

IAR C-SPY® BDM Debugger

User Guide

for Freescale's
HCS12 Microprocessor Family



CSHCS12B-2

 IAR
SYSTEMS

COPYRIGHT NOTICE

Copyright © 1997–2010 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 contained 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.

Freescale is a registered trademark of Freescale Inc.

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

EDITION NOTICE

Second edition: January 2010

Part number: CSHCS12B-2

This guide applies to version 3.x of IAR Embedded Workbench® for HCS12.

Internal reference: R10, 5.6.x, 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	viii
Naming conventions	viii
Introduction to C-SPY® BDM Debugger	1
The C-SPY BDM debugger	1
Differences between the C-SPY drivers	2
The C-SPY BDM debugger driver	2
Hardware installation	3
Getting started	3
Running the demo program	4
BDM-specific debugging	7
Debugger options for debugging using hardware systems	7
Setup	8
Download	9
Communication	11
Extra Options page	12
BDM debugger menu	12
Using breakpoints	13
Available breakpoints	13
Breakpoint Usage dialog box	14
Resolving problems	14
Using flash loaders	17
The flash loader	17
Setting up the flash loader(s)	17

The flash loading mechanism	18
Build considerations	18
Flash Loader Overview dialog box	18
Flash Loader Configuration dialog box	20
Index	23

Preface

Welcome to the *IAR C-SPY® BDM Debugger User Guide*. The purpose of this guide is to provide you with detailed reference information that can help you use the features in the IAR C-SPY® Hardware Debugger Systems for HCS12.

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 or C++ programming language
- Application development for embedded systems
- The architecture and instruction set of the HCS12 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; this guide does not describe 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 might have been added to the software after the *IAR C-SPY® BDM Debugger User Guide* was printed. The IAR Information Center guides you to the latest information.

What this guide contains

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

- *Introduction to C-SPY® BDM Debugger* introduces you to the C-SPY driver for the BDM debugger. The chapter briefly shows the difference in functionality provided by the BDM debugger and the C-SPY simulator.
- *BDM-specific debugging* describes the additional options, menus, and features provided by this debugger system.

Other documentation

The complete set of IAR Systems development tools for the HCS12 microcontroller are described in a series of guides. For information about:

- Programming for the IAR C/C++ Compiler, refer to the *IAR C/C++ Compiler Reference Guide for HCS12*
- Programming for the IAR Assembler, refer to the *IAR Assembler Reference Guide for HCS12*
- Using the IAR XLINK Linker, the IAR XAR Library Builder, and the IAR XLIB Librarian, refer to the *IAR Linker and Library Tools Reference Guide*
- Porting application code and projects created with a previous version of the IAR Embedded Workbench for HCS12, refer to the *IAR Embedded Workbench Migration Guide for HCS12*.

These guides can be found in the `hcs12\doc` directory or reached from the **Help** menu.

All of these guides are delivered in hypertext PDF or HTML format on the installation media.

Recommended web sites:

- The Freescale web site, www.freescale.com, contains information and news about the HCS12 microcontrollers.
- The IAR Systems web site, www.iar.com, holds application notes and other product information.

Document conventions

When, in this text, we refer to the programming language C, the text also applies to C++, unless otherwise stated.

When referring to a directory in your product installation, for example `hcs12\doc`, the full path to the location is assumed, for example `c:\Program Files\IAR Systems\Embedded Workbench 5.n\hcs12\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.
<code>parameter</code>	A placeholder for an actual value used as a parameter, for example <code>filename.h</code> where <code>filename</code> represents the name of the file.
<code>[option]</code>	An optional part of a command.
<code>a b c</code>	Alternatives in a command.
<code>{a b c}</code>	A mandatory part of a command with alternatives.
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

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 HCS12	IAR Embedded Workbench®
IAR Embedded Workbench® IDE for HCS12	the IDE
IAR C-SPY® Debugger for HCS12	C-SPY, the debugger
IAR C-SPY® Simulator	the simulator
IAR C/C++ Compiler™ for HCS12	the compiler

Table 2: Naming conventions used in this guide

Brand name	Generic term
IAR Assembler™ for HCS12	the assembler
IAR XLINK™ Linker	XLINK, the linker
IAR XAR Library builder™	the library builder
IAR XLIB Librarian™	the librarian
IAR DLIB Library™	the DLIB library

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

Introduction to C-SPY® BDM Debugger

This chapter introduces you to the C-SPY BDM Debugger system and to how it differs from the C-SPY Simulator.

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

The C-SPY BDM debugger

C-SPY consists of both a general 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, windows, and dialog boxes—to the functions provided by the target system. This driver is automatically installed during the installation of IAR Embedded Workbench.

At the time of writing this guide, the IAR C-SPY Debugger for the HCS12 microcontrollers is available with these drivers:

- BDM debugger
- Simulator.

The C-SPY BDM Debugger 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 C-SPY BDM Debugger driver can connect to these interfaces:

P&E Microcomputer Systems	Freescale	SofTec
CABLE12	Motorola SDI	inDART-HCS12
CABLE12HS		inDART-One
Parallel BDM Multilink		
USB BDM Multilink		
Cyclone PRO		

Table 3: BDM debugger interfaces

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

DIFFERENCES BETWEEN THE C-SPY DRIVERS

This table summarizes the key differences between the C-SPY drivers:

Feature	Simulator	BDM debugger
Code breakpoints (OP-fetch)	Unlimited	x ¹
Data breakpoints	x	--
Execution in real time	--	x
Zero memory footprint	x	x
Simulated interrupts	x	--
Real interrupts	--	x
Interrupt logging	x	--
Live watch	--	x
Cycle counter	x	--
Code coverage	x	--
Data coverage	x	--
Function/instruction profiler	x	--
Profiling	x	x ^{2, 3}
Trace	x	--

Table 4: Driver differences

1 For detailed information about available breakpoints, see Table 8, *Hardware and software breakpoints*, page 13.

2 Cycle counter statistics are not available.

3 Profiling works provided that enough breakpoints are available. That is, the application is executed in RAM.

The C-SPY BDM debugger driver

The C-SPY BDM debugger driver is automatically installed during the installation of IAR Embedded Workbench. Using the C-SPY BDM debugger driver, C-SPY can connect to a BDM debugger. Many of the HCS12 microcontrollers have built-in, on-chip debug support. Because the hardware debugger logic is built into the microcontroller, no ordinary ROM-monitor program or extra specific hardware is needed to make the debugging work.

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

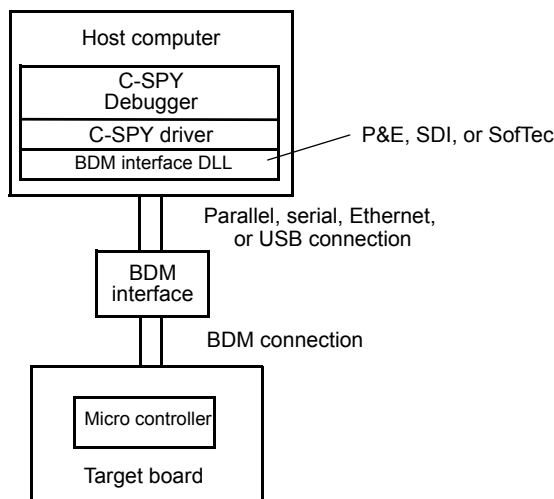


Figure 1: C-SPY BDM debugger communication overview

For further information, refer to the documentation supplied with the BDM debugger.

When a debugging session is started, your application is automatically downloaded and programmed into flash memory. You can disable this feature, if necessary.

HARDWARE INSTALLATION

For information about the hardware installation, see the documentation supplied with the BDM debugger from Freescale. The following power-up sequence is recommended to ensure proper communication between the target board, BDM debugger, and C-SPY:

- 1 Power up the target board.
- 2 Power up the BDM debugger.
- 3 Start the C-SPY debugging session.

Getting started

IAR Embedded Workbench comes with example applications. You can use these examples to get started using the development tools from IAR Systems or simply to

verify that contact has been established with your target board. You can also use the examples as a starting point for your application project.

You can find the examples from the IAR Information Center. Click the button **EXAMPLE PROJECTS**. The examples are ready to be used as is. They are supplied with ready-made workspace files, together with source code files and linker command files.

RUNNING THE DEMO PROGRAM

In this example, the demo project is set up for the Motorola M68KIT 9S12DP256B board, where the LEDs are connected to `PORTB`. When you run the demo program, the LEDs will blink.

The procedure can be applied to other evaluation boards as well.

- 1 To use an example application, choose **Help>Information Center** and click **EXAMPLE PROJECTS**. Choose an example project and then click **Open project**. For example, you can choose the example application **BDM_demo**.
- 2 In the dialog box that appears, choose a destination folder for your project location. Click **Select** to confirm your choice.
- 3 The available example projects are displayed in the workspace window. Select one of the projects, and if it is not the active project (highlighted in bold), right-click it and choose **Set As Active** from the context menu.
- 4 To view the project settings, select the project and choose **Options** from the context menu. Make sure these options are selected:

Category	Page	Option/Setting
General Options	Target	Device: MC9S12DP256B ¹⁾
Debugger	Setup	Driver: BDM debugger
BDM Debugger	Communication	Communication type: P&E USB BDM Multilink ²⁾

Table 5: Project options for C-SPY BDM debugger example

1) A default linker command file (`xc1`) and device description file (`ddf`) will automatically be used depending on your choice of device.

2) If you use a different communication type, you might have to specify a port, the crystal frequency, and baud rate, accordingly.

As for other settings, the project is set up to suit the target system you selected.

For further details about the C-SPY options for the hardware target system and how to configure C-SPY to interact with the target board, see *Debugger options for debugging using hardware systems*, page 7.

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

- 5 To compile and link the application, choose **Project>Make** or click the **Make** button.

- 6** To start C-SPY, choose **Project>Debug** or click the **Debug** button. If C-SPY fails to establish contact with the target system, see *Resolving problems*, page 14.
- 7** If C-SPY should fail to establish contact with the target hardware, see *Resolving problems*, page 14.
- 8** Set a breakpoint on this line:

```
PORTB = -arr[i++];
```

Note: The number of physical hardware breakpoints used when setting breakpoints is limited. To read more about this, see *Using breakpoints*, page 13.

- 9** Click the **Go** button a few times to see how the LEDs on the board change as different values are written to `PORTB`.
- 10** To see the LEDs blink continuously, remove the breakpoint and click **Go**.
Click the **Stop** button to stop execution.

BDM-specific debugging

This chapter describes the additional options, menus, and features provided by the C-SPY® BDM debugger systems. The chapter contains the following sections:

- Debugger options for debugging using hardware systems
- BDM debugger menu
- Using breakpoints
- Resolving problems.

Debugger options for debugging using hardware systems

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

- 1 To open the **Options** dialog box, choose **Project>Options**.
- 2 To set C-SPY generic options and select a C-SPY driver:
 - Select **Debugger** from the **Category** list
 - On the **Setup** page, select the appropriate C-SPY driver from the **Driver** drop-down list, in this case the **BDM debugger**.

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 device description file and linker command file is automatically selected depending on your selection of a 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.

- 3 To set the driver-specific options, select the appropriate driver from the **Category** list. Note that the options required for BDM debugging are only available when you have selected **BDM debugger** from the **Driver** list on the **Debugger>Setup** page.

For details about each page, see:

- *Setup*, page 8
 - *Download*, page 9
 - *Communication*, page 11
 - *Extra Options page*, page 12.
- 4 To set the option for communication, open the **Communication** page. For details about these options, see *Communication*, page 11.
 - 5 When you have set all the required options, click **OK** in the **Options** dialog box.

SETUP

The **Setup** page contains the options for choosing operating mode.

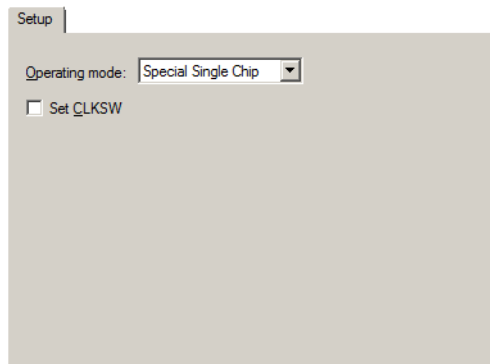


Figure 2: Setup page

Operating mode

This option is used for choosing the operating mode. You can choose between:

- Normal Expanded Narrow
- Normal Expanded Wide
- Normal Single Chip
- Special Expanded Narrow
- Special Expanded Wide
- Special Single Chip.

For more information about the different operating modes, refer to the hardware documentation from Freescale Inc.

Set CLKSW

Use this option to set the CLKSW bit in the BDM status register. Due to hardware errors in some Freescale devices, C-SPY can lose the BDM connection to these devices, in particular when changing the clock speed using Phase-Locked Loop (PLL). Setting the CLKSW bit will eliminate some of these problems.

DOWNLOAD

By default, C-SPY downloads the application to RAM or flash when a debug session starts. The **Download** options let you modify the behavior of the download.

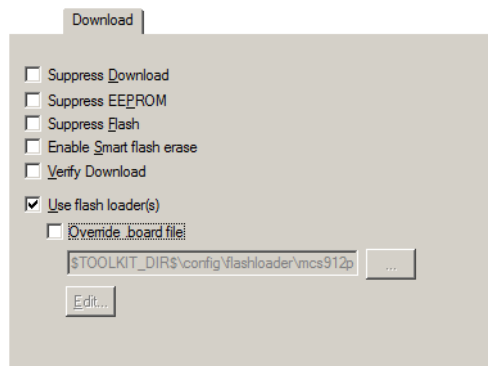


Figure 3: C-SPY Download options

Suppress Download

Use this option to debug an application that already resides in target memory. When this option is selected, the code download is disabled, while preserving the present content of the flash.

If this option is combined with the **Verify download** option, the debugger will read back the code image from non-volatile memory and verify that it is identical to the debugged program.

Suppress EEPROM

This option suppresses EEPROM download.

Suppress Flash

This option suppresses flash download.

Enable Smart flash erase

This option erases a flash array only if the bytes to be downloaded are different from the bytes already present.

Verify Download

Use this option to verify that the downloaded code image can be read back from target memory with the correct contents.

Use flash loader(s)

Use this option to use one or several flash loaders for downloading your application to flash memory. If a flash loader is available for the selected chip, it is used by default. Press the **Edit** button to open the **Flash Loader Overview** dialog box.

To read more about flash loaders, see *The flash loader*, page 17.

Note: If there is a built-in flash loader for your microcontroller, this option is not available.

Override .board file

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 .board file** and specify the path to the flash loader you want to use. A browse button is available for your convenience. Click **Edit** to open the **Flash Loader Overview** dialog box. For more information, see *Flash Loader Overview dialog box*, page 18.

COMMUNICATION

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

The screenshot shows a window titled "Communication" with the following settings:

- Communication type: Motorola SDI
- Port: COM1
- Crystal frequency (KHz): 16000
- Baud rate: 19200

Figure 4: Communication page

These settings are available:

Setting	Description
Communication type	Selects one of the supported communication types: <ul style="list-style-type: none"> • Motorola SDI • P&E Cable I2 • P&E Cable I2HS • P&E Parallel BDM Multilink • P&E USB BDM Multilink • Cyclone Pro Serial • Cyclone Pro USB • Cyclone Pro Ethernet • inDART-HCS12 • inDART-One
Port	Selects one of the supported ports. Choose either between COM1–4 or LPT1–3. ^{*)}
Crystal frequency (kHz)	Specifies the crystal frequency. ^{*)}
Baud rate	Selects the speed of the COM port. ^{*)}

Table 6: Communication options

***) This option depends on your communication type. If the option is not enabled, it is either not needed or automatically detected by C-SPY.**

EXTRA OPTIONS PAGE

The **Extra Options** page provides you with a command line interface to C-SPY.



Figure 5: Extra Options page for C-SPY command line options

Use command line options

Additional command line arguments (not supported by the GUI) for C-SPY can be specified here.

BDM debugger menu

When you are using the C-SPY BDM debugger driver, the **BDM debugger** menu appears in C-SPY.

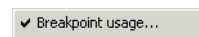


Figure 6: The BDM debugger menu

This command is available on the **BDM debugger** menu:

Menu command	Description
Breakpoint Usage	Displays the Breakpoint Usage dialog box which lists all active breakpoints; see <i>Breakpoint Usage dialog box</i> , page 14.

Table 7: Commands on the BDM debugger menu

Using breakpoints

This section provides an overview of the available breakpoints for the C-SPY hardware debugger systems. This is described:

- *Available breakpoints*, page 13
- *Breakpoint Usage dialog box*, page 14.

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

AVAILABLE BREAKPOINTS

Using the C-SPY drivers for hardware debugger systems you can set *code breakpoints*. The physical breakpoint used on the target can be one of two types—*hardware* and *software*.

This table summarizes the characteristics of breakpoints for the different target systems:

Type	Number	Execution overhead
Hardware	Normally two	No
Software (application located in RAM memory)	Unlimited	Yes

Table 8: Hardware and software breakpoints

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.

Hardware breakpoints

Hardware breakpoints are available in the microcontroller. The number of breakpoints is normally two.

Software breakpoints

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 your application, or parts of it, must be temporarily located in read/write memory. Software breakpoints are implemented by a temporary substitution of 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.

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.

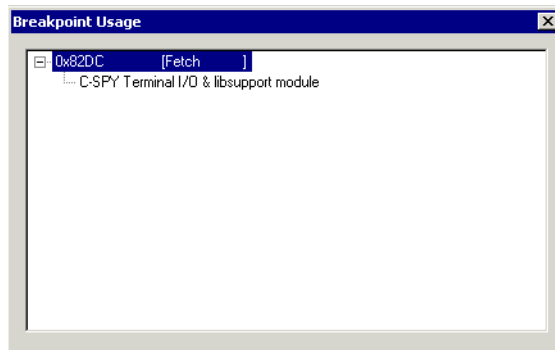


Figure 7: Breakpoint Usage dialog box

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.

For more information, see the *IAR Embedded Workbench® IDE User Guide*.

Resolving problems

Debugging using the C-SPY hardware debugger systems requires interaction between many systems, independent from each other. For this reason, setting up this debug system can be a complex task. 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 with the C-SPY hardware debugger systems.

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 most common is that your application has been incorrectly linked:

- Check the contents of your linker command file and make sure that your application has not been linked to the wrong address
- Check that you are using correct linker command file.



If you are using the IAR Embedded Workbench, the linker command file is 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 list.



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** area.

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*, page 11

Examine the linker command file to make sure that your 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 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.

A set of flash loaders for various microcontrollers is provided with IAR Embedded Workbench for HCS12. The flash loader API, documentation, and several implementation examples 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 **BDM Debugger** category and click the **Download** tab.
- 3 Select the **Use Flash loader(s)** option. A default flash loader configured for the device you have specified will be used. The configuration is specified in a preconfigured `board` file.
- 4 To override the default flash loader or to modify the behavior of the default flash loader to suit your board, select the **Override default .board file** option, and **Edit** to open the **Flash Loader Configuration** dialog box. A copy of the `*.board` file will be created in your project directory and the path to the `*.board` file will be updated accordingly.
- 5 The **Flash Loader Overview** dialog box lists all currently configured flash loaders; see *Flash Loader Overview dialog box*, page 18. 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 various flash loader options, see *Flash Loader Configuration dialog box*, page 20.

THE FLASH LOADING MECHANISM

When the **Use flash loader(s)** option is selected and one or several flash loaders have been configured, these steps are 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 programs the application code into flash memory.
- 4 The flash loader terminates.
- 5 C-SPY switches context to the user application.

Steps 2 to 4 are performed for each memory range of the application.

The steps 1 to 4 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 (d12) 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. 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.

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.

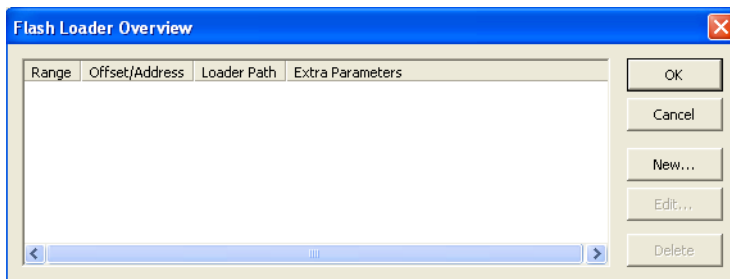


Figure 8: Flash Loader Overview dialog box

The display area

Each row in the display area shows how you have set up one flash loader for flashing a specific part of memory:

Column	Description
Range	The part of your application to be programmed by the selected flash loader.
Offset/Address	The start of the memory where your application will be flashed. If the address is preceded with <code>a</code> , the address is absolute. Otherwise, it is a relative offset to the start of the memory.
Loader Path	The path to the flash loader <code>*.flash</code> file to be used (<code>*.out</code> for old-style flash loaders).
Extra Parameters	List of extra parameters that will be passed to the flash loader.

Table 9: Flash Loader Overview columns

Click on the column headers to sort the list by name, location, or full name.

Function buttons

These function buttons are available:

Button	Description
OK	The selected flash loader(s) will be used for downloading your application to memory.
Cancel	Standard cancel.

Table 10: Function buttons in the Flash Loader Overview dialog box

Button	Description
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 20.
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 20.
Delete	Deletes the selected flash loader configuration.

Table 10: Function buttons in the Flash Loader Overview dialog box (Continued)

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 to suit your board. A copy of the default `board` file will be created in your project directory.

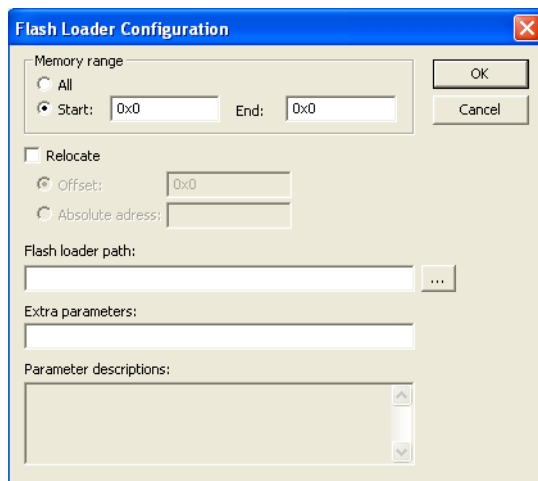


Figure 9: Flash Loader Configuration dialog box

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, that is relocate the location of the application in memory. This means that you can flash your application to a different location from where it was linked. Choose between:

- Offset** A numeric value for a relative offset. This offset will be added to the addresses in the application file.
- Absolute address** A numeric value for an absolute base address where the application will be flashed. The lowest address in the application will be placed on this address. Note that you can only use one flash loader for your application when you specify an absolute address.

You can use these numeric formats:

- 123456 Decimal numbers
- 0x123456 Hexadecimal numbers
- 0123456 Octal numbers

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.

Flash loader path

Use the text box to specify the path to the flash loader file (`*.flash`) to be used by your board configuration.

Extra parameters

Some flash loaders define their own set of specific options. Use this text box to specify options to control the flash loader. For information about available flash loader options, see the **Parameter descriptions** field.

Parameter descriptions

The **Parameter descriptions** field displays a description of the extra parameters specified in the **Extra parameters** text box.

A

assumptions, programming experience v

B

Baud rate (C-SPY option) 11

BDM debugger (C-SPY driver) 2

 hardware installation 3

bold style, in this guide viii

Breakpoint Usage dialog box (Simulator menu) 14

breakpoints, hardware (BDM debugger) 13

C

command line options, typographic convention viii

command prompt icon, in this guide viii

Communication type (C-SPY option) 11

Communication (C-SPY options) 11

computer style, typographic convention viii

conventions, used in this guide vi

copyright notice ii

Crystal frequency (kHz) (C-SPY option) 11

C-SPY

 BDM debugger 1

 differences between drivers 2

C-SPY drivers, BDM debugger 2

C-SPY options

 Communication 11

 Download 9

 Setup 8

C++ terminology vi

D

debugger drivers, BDM debugger 2

disclaimer ii

document conventions vi

documentation

 other documentation vi

 this guide v

Download (C-SPY options) 9

E

edition, of this guide ii

Enable Smart flash erase (C-SPY option) 10

Extra Options, for C-SPY hardware driver 12

F

Flash Loader Overview dialog box 18

Flash loader path, specifying the path to a flash loader 21

flash loader, using 17

H

hardware breakpoints (BDM debugger) 13

I

icons, in this guide viii

italic style, in this guide viii

L

lightbulb icon, in this guide viii

N

naming conventions viii

O

Operating mode (C-SPY option) 8

options
 hardware debugger systems 7

P

parameters
 list of passed to the flash loader 19
 specify to control flash loader 22
parameters, typographic convention viii
part number, of this guide ii
Port (C-SPY option) 11
prerequisites, programming experience. v
programming experience. v
publication date, of this guide ii

R

reference information, typographic convention. viii
registered trademarks ii

S

Set CLKSW (C-SPY option) 9
Setup (C-SPY options) 8
Suppress Download (C-SPY option). 9
Suppress EEPROM (C-SPY option) 9
Suppress Flash (C-SPY option) 9

T

terminology. vi
tools icon, in this guide viii
trademarks ii
typographic conventions viii

U

Use flash loader (C-SPY option). 10

V

Verify Download (C-SPY option) 10
version, IAR Embedded Workbench. ii

W

warnings icon, in this guide viii