8051 IAR C-SPY Hardware Debugger Systems

User Guide

for the

MCS-51 Microcontroller Family

COPYRIGHT NOTICE

DISCLAIMER

TRADEMARKS

EDITION NOTICE

Second edition: October 2005

Part number: CS8051R-2

This guide applies to 8051 IAR Embedded Workbench™ version 7.x.

Contents

Tables

Figures

Preface

Welcome to the *8051 IAR C-SPY Hardware Debugger Systems User Guide*. The purpose of this guide is to provide you with detailed reference information that can help you use the features in the 8051 IAR C-SPY® Hardware Debugger Systems.

Who should read this guide

You should read this guide if you want to debug your application with an existing debug solution or if you want to create your own hardware debug monitor for your target board. In addition, you should have a working knowledge of:

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

This guide also assumes that you already have a 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; it does not describe the general features available in the IAR C-SPY debuggger 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 avaialable 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 *8051 IAR C-SPY Hardware Debugger Systems User Guide* was printed. The release notes `cs8051r.htm`, `cs8051si.htm`, `cs8051cc.htm`, and `cs8051ad.htm` contain 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 systems to be used for the target systems. The chapter briefly shows the difference in functionality provided by the different C-SPY drivers.
- *Hardware-specific debugging* describes the additional options, menus, and features provided by these debugger systems.

Other documentation

The complete set of IAR development tools for the MCS-51 microcontroller are described in a series of guides. For information about:

- Using the IAR Embedded Workbench® IDE with the C-SPY® Debugger, refer to the *IAR Embedded Workbench® IDE User Guide*
- Programming for the 8051 IAR C/C++ Compiler, refer to the *8051 IAR C/C++ Compiler Reference Guide*
- Programming for the 8051 IAR Assembler, refer to the *8051 IAR Assembler Reference Guide*
- 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*
- Using the IAR DLIB Library, refer to the *DLIB Library Reference information*, available in the 8051 IAR Embedded Workbench IDE online help system.
- Using the IAR CLIB Library, refer to the *IAR C Library Functions Reference Guide*, available in the 8051 IAR Embedded Workbench IDE online help system.
- Porting application code and projects created with a previous version of the 8051 IAR Embedded Workbench IDE, refer to *8051 IAR Embedded Workbench Migration Guide.*

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 IAR Systems web site, **www.iar.com**, holds application notes and other product information.
- The Analog Devices website, **www.analog.com**, contains information about the ADu8xx microcontrollers.
- The Silicon Laboratories website, **www.silabs.com,** contains information about the C8051Fxxx microcontrollers.
- The Chipcon website, **www.chipcon.com**, contains information about the Chipcon 8051 microcontrollers.

- Finally, the Embedded C++ Technical Committee web site, **www.caravan.net/ec2plus**, contains information about the Embedded C++ standard.

Document conventions

This book uses the following typographic conventions:

Style	Used for
`computer`	Text that you type or that appears on the screen.
`parameter`	A label representing the actual value you should type as part of a command.
`[option]`	An optional 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.
reference	A cross-reference within this guide or to another guide.
…	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.

Table 1: Typographic conventions used in this guide

Introduction to C-SPY hardware debugger systems

This guide introduces you to the 8051 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*.

The IAR C-SPY hardware debugger systems

The IAR 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, windows, and dialog boxes—to the functions provided by the target system, for instance special breakpoints. This driver is automatically installed during the installation of the IAR Embedded Workbench.

At the time of writing this guide, the IAR C-SPY Debugger for the MCS-51-compatible microcontrollers is available with drivers for the following target systems:

- Simulator
- Generic and target-specific ROM-monitors
- Analog Devices' ADu84x protocol
- Silicon Laboratories' C8051Fxxx protocol

For information about the availability of other drivers, see the IAR Systems website, **www.iar.com**. For specific technical details of a certain driver, see the readme file for this driver.

When a debugging session is started, the application code is automatically downloaded into writable code memory. This feature can be disabled, if necessary.

Note: In addition to the drivers supplied with IAR Embedded Workbench, it is also possible to load debugger drivers supplied by a third-party vendor; see *Debugging using a third-party driver*, page 10.

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

DIFFERENCES BETWEEN THE HARDWARE DEBUG SYSTEMS AND THE SIMULATOR

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

Feature	Simulator	Analog Devices	Silabs	ROM-monitor
Code breakpoints	Unlimited	Unlimited	4	Target-dependent
Data breakpoints	Yes	No	No	Target-dependent
Execution in real time	No	Yes	Yes	Yes
Zero memory footprint	Yes	Yes	No	No
Simulated interrupts	Yes	No	No	No
Real interrupts	No	Yes	Yes	Yes
Cycle counter	Yes	No	No	No
Code coverage	Yes	No	No	No
Data coverage	Yes	No	No	No
Profiling	Yes	Yes	Yes	Yes, depending on the number of code breakpoints

Table 1: Driver differences

Below are general descriptions of the different drivers.

The Analog Devices driver

The Analog Devices driver is automatically installed during the installation of IAR Embedded Workbench. This driver works as an interface to the ADu84x protocol from Analog Devices. Because the ROM-monitor is located within the boot loader, no ordinary ROM-monitor program or extra specific hardware is needed to make the debugging work.

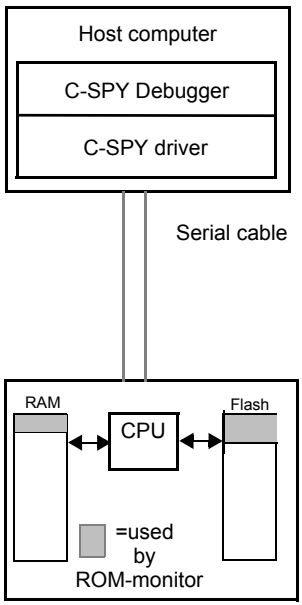

Figure 1: Analog Devices driver communication overview

For further information, refer to the documentation supplied with the target system.

This driver supports all ADu84x microcontrollers from Analog Devices. For information about availability, see their website **www.analog.com**.

The Silabs driver

The Silabs driver is automatically installed during the installation of IAR Embedded Workbench. The Silabs driver works as an interface to the C8051Fxxx protocol that is used to communicate with the C8051Fxxx microcontrollers from Silicon Laboratories. For information about availability, see their website **www.silabs.com**.

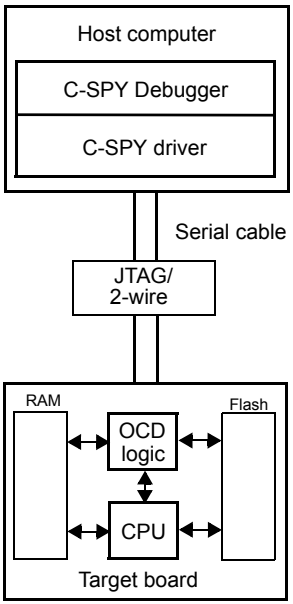

Figure 2: Silabs driver communication overview

The C-SPY ROM-monitor driver

The IAR C-SPY ROM-monitor driver is automatically installed during the installation of IAR Embedded Workbench. Using the C-SPY ROM-monitor driver, C-SPY can connect to a ROM-monitor located on the development board.

The ROM-monitor driver uses the IAR ROM-monitor protocol over the RS232 port to communicate with the ROM-monitor that is installed on a target system.

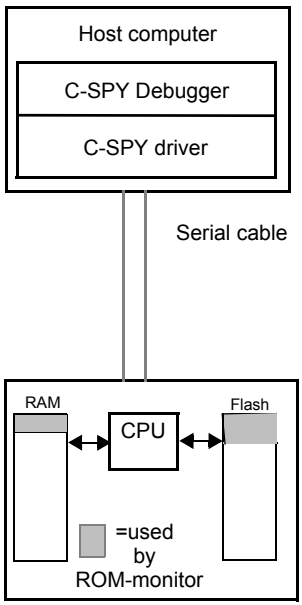

Figure 3: IAR ROM-monitor driver communication overview

For further information, refer to the documentation supplied with the system target.

When a debugging session is started, your application is automatically downloaded and programmed into writable CODE memory. This feature can be disabled, if necessary.

Building a new ROM-monitor

There is a large number of 8051 devices on the market. The variety makes it impossible to have one ROM-monitor firmware to support them all. Therefore a full ROM-monitor project is included in the product package to enable customizing for a specific target. Follow this procedure to build the ROM-monitor:

1 Open IAR Embedded Workbench. Choose **Create new project** and select **ROM-monitor**. Click **OK**.

A project containing the monitor files is now created. The project contains common monitor files and files that need to be edited. The generic files are located in `src\rom\common_src` and are not copied, while the files that need to be edited are copied automatically to the project directory.

2 Build the project. A number of error messages will be generated. Each error is located close to a function that needs to be edited in order to make the ROM-monitor work. Take some time to look in the source files to get a feeling of what needs to be done. You can easily move from error to error with the F4 key.

3 Edit the `chip_layout.h` file to set important settings, such as breakpoints, bus width, special SFR registers, etc. This is the most important file in the project as it is used to control how to build the ROM-monitor and contains all defines. For information about some of the settings, see *Setting up chiplayout.h* and the header file.

4 It is highly recommended to start with reading the information about the UART in the chip manual. Put together a small program that echoes characters received on the UART. Use a standard terminal window on the host machine to verify the UART functionality. Once this program is running, you can move the UART setup to the ROM-monitor source code.

5 Set up the UART in the function `uart_init()`. All communication starts at 9600 baud. The ROM-monitor driver will, at a later stage, call the function `set_baudrate()` to init a baud rate change to speed up the communication. The baud rates supported by the target is defined in the file `chip_layout.h`.

6 Set up the target-specific details, such as enabling memory, initializing clocks, etc. This is done in `low_level_init()` and `high_level_init()`. `low_level_init()` is called after a hard reset from the `cstartup` code before initializing variables while `high_level_init()` is called each time the ROM-monitor is entered.

7 Most memory access methods are the same for all 8051 derivatives, except for the method for writing code. Edit the function `byte_write_code()`. Depending on your target you might also have to edit the file `chip_layout.h` according to the instructions in the file.

8 Download the generated ROM-monitor using the appropriate dowload tool. Create a simple test project, choose ROM-monitor as the driver and set baudrate. Writing code is usually one of the parts that can be hard to get to work. Therefore it is recommended to select **Suppress download**. Make sure you deselect **Verify download**. This way you can start with verifying that the ROM-monitor's basic functions work. Once this is done, you can deselect **Suppress download** and download a test program.

SETTING UP CHIPLAYOUT.H

Even though the ROM-monitor consists of a large number of files, very few usually need to be edited to make the ROM-monitor run. One of the files that need to be edited is `chip_layout.h`. This is one of the most important files in the project, as it sets up the conditions under which the ROM-monitor will work. Some of the sections of this file is described here, while there is more information in the header file.

Breakpoints

There are two types of beakpoints: software and hardware. The ROM-monitor uses the `LCALL` instruction as a generic software breakpoint. There is also support for target-specific breakpoint instructions, such as `0xA5`. Set `SW_BP_TYPE` to either `BP_OF_LCALL_TYPE` or `BP_OF_A5_TYPE`. If no software breakpoints are to be used, set `SW_BP_TYPE` to `NO_SW_BP`.

Application bus width

The application bus width controls the addressable memory area. It is either 16 or 24 bit depending on the target or the location of the application on the target.

Remapped IDATA

The ROM-monitor will need some working memory. As a default this is `0x00-0x7F` in DATA. To free this memory when the user application is running, this memory will be copied into PDATA(XDTATA)/IDATA. The define `MON_REMAP_IDATA_TO_MEM` controls the type of memory used for this copy.

Special SFR registers

Some SFR registers are needed by the ROM-monitor and would be overwritten if shared with the application. To avoid this these registers are stored/restored by the monitor. For this purpose an SFR info struct is used. In this struct special SFRs that need special care can be added.

Flash page

The flash page section defines the flash page properties such as page size and total memory size. When writing to memory C-SPY will send the bytes to write at full speed over the UART. Depending on the flash memory the bytes might come to fast. For this purpose `FLASH_WRITE_DELAY` can be defined to insert a delay between each byte.

Debugging the ROM-monitor

Debugging a ROM-monitor can sometimes be really hard, due to the number of subsystems involved (PC, host debugger, ROM-monitor firmware, hardware and user application). One way is to use the I/O signals on the target chip lightning LEDs or writing displays that let you know where you are in the program and what part you never reach.

Another way is to use the simulator. This can easily be done by letting C-SPY simulate the UART and reading the communication data from a file. There is a pre-made macro file `serialData.mac` included with the product that will set up the necessary interrupts and feed the simulated UART with data. There is also a UART data file `SerialData.txt` included, which contains some basic driver commands. When the macro is used, the data from the file will be sent as input to the monitor triggering the different actions/calls. The files are located at `8051\src\rom`.

Hardware-specific debugging

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

- C-SPY options for debugging using hardware systems
- Debugging using a third-party driver
- Chipcon driver-specific debugging
- ROM-monitor driver-specific debugging
- Analog Devices driver-specific debugging
- Silabs driver-specific debugging
- Using breakpoints

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 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 debugger-generic options and select a debugger driver:
   - Select **Debugger** from the **Category** list
   - On the **Setup** page, select the appropriate debugger driver from the **Driver** list.

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

   Note that a default device description file is automatically selected depending on your selection of a device on the **General Options>Target** page.

3 To set the driver-specific options, select the appropriate driver from the **Category** list. Depending on which debugger driver you are using, the corresponding driver options will be enabled. When you have set all the required options, click **OK** in the **Options** dialog box.

Debugging using a third-party driver

It is possible to load other debugger drivers than those supplied with the IAR Embedded Workbench.

THIRD-PARTY DRIVER OPTIONS

The **Third-Party Driver** options are used for loading any driver plugin provided by a third-party vendor. These drivers must be compatible with the C-SPY debugger driver specification.

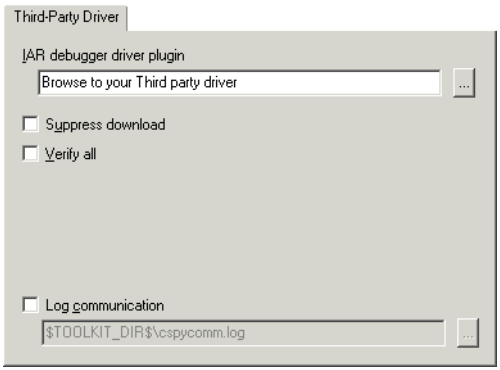

Figure 4: C-SPY Third party driver options

IAR debugger driver plugin

Enter the file path to the third-party driver plugin DLL file, in this text box, or browse to the driver DLL file using the browse button.

Log communication

Use this option to log the communication between C-SPY and the target system to a file. To interpret the result, a detailed knowledge of the interface is required. This option can be used if is supported by the third-party driver.

Chipcon driver-specific debugging

In this section you can read about the features specific for the Chipcon emulator driver:

DOWNLOAD

On the **Download** page you can control the download.

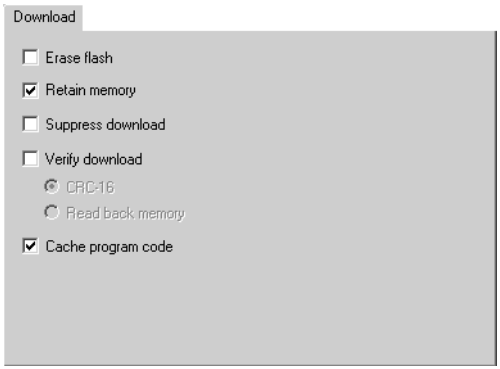

Figure 5: Chipcon download options

Erase flash

The option **Erase flash** erases all flash before download.

Retain memory

With the option **Retain memory** only the changed, new or updated pages will be downloaded to flash, saving flash cycles/life time.

Suppress download

If you already have your application in flash memory, select **Suppress download**. It is highly recommended that you also select **Verify download** if you select **Suppress download**.

Verify download

The option **Verify download** verifies that the program data has been correctly transferred from the driver to the device. **Select CRC-16** to verify a target using on-chip page CRC16, and select **Read back memory** to verify a target by reading back memory. This verification increases the programming sequence time.

Cache program code

This option enables program code cache while debugging.

TARGET

The **Target** page contains options interface speed, stack overflow warnings, number of banks and communication logs for the Chipcon driver:

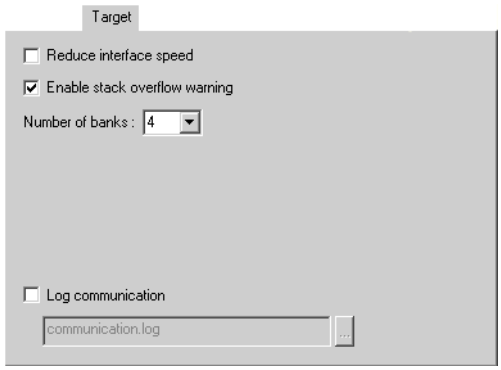

Figure 6: Chipcon driver serial port options

Reduce interface speed

With this option you can slow down interface speed, which can be very useful if you use a long cable or encounter communication problems or interference.

Enable stack overflow warning

This option enables stack overflow warnings. This is not a runtime check, but is done at the next stop, which means that it will not stop the execution if a stack overflow is encountered.

Number of banks

With this option you can set the number of banks.

Log communication

With this option you can choose a log file for your communication log.

THE CHIPCON MENU

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

The following commands are available on the **Chipcon** menu:

Menu command	Description
Breakpoint Usage	Displays the Breakpoint Usage dialog box which lists all active breakpoints; see *Breakpoint Usage dialog box*, page 21.
Stop timers on halt	Stops the timers when the execution is stopped

Table 2: Commands on the Chipcon menu

ROM-monitor driver-specific debugging

In this section you can read about the features specific for the generic ROM-monitor driver:

DOWNLOAD

On the **Download** page you can control the download.

Figure 7: Download options for the generic ROM-monitor driver

Verify download

The option **Verify download** verifies that the program data has been correctly transferred from the driver to the device. This verification does increase the programming sequence time.

Suppress download

If you already have your application in flash memory, select **Suppress download**. It is highly recommended that you also select **Verify download** if you select **Suppress download**.

SERIAL PORT

The **Serial Port** page contains all the communication options for the generic ROM-monitor driver:

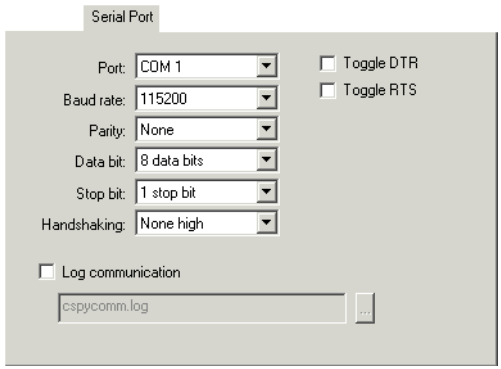

Figure 8: Serial port options for the generic ROM-monitor driver

Option	Setting
Port	One of the supported ports: **COM1, COM2, COM3, COM4**
Baud rate	**38400**
Parity	Only **None** is supported
Data bit	Only **8 data bits** is supported
Stop bit	Only **2 stop bits** is supported
Handshaking	**None high** and **None low**

Table 3: Serial Port options for the generic ROM-monitor driver

C-SPY tries to connect at 9600 baud and then changes to the baud rate of the selected serial port when making the first contact with the evaluation board. If these options have not been specified, C-SPY will try using the COM1 port.

Toggle DTR

This option will toggle DTR signals to reset the target.

Toggle RTR

This option will toggle RTR signals to reset the target.

Log communication

For troubleshooting purposes, there is a possibility to log all characters sent between C-SPY and the ROM-monitor program to a file. If you select the **Log communication** option, the file `cspycomm.log` will be used in the current working directory. You can use the browse button to choose another file or location.

THE GENERIC ROM-MONITOR MENU

When you are using the Generic ROM-monitor driver, the **Generic ROM-monitor** menu appears in C-SPY.

The following commands are available on the **Generic ROM-monitor** menu:

Menu command	Description
Breakpoint Usage	Displays the Breakpoint Usage dialog box which lists all active breakpoints; see *Breakpoint Usage dialog box*, page 21.

Table 4: Commands on the generic ROM-monitor menu

Analog Devices driver-specific debugging

In this section you can read about the features specific for the Analog Devices driver:

DOWNLOAD

On the **Download** page you can control the download.

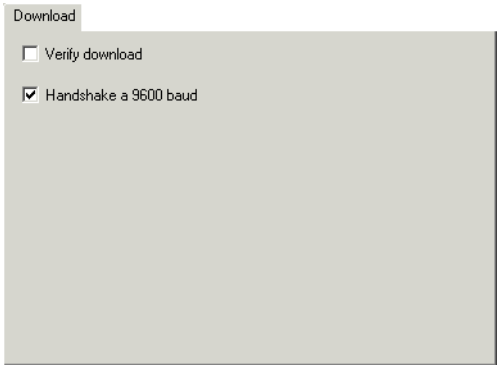

Figure 9: Analog Devices driver download options

Verify download

The option **Verify download** verifies that the program data has been correctly transferred from the driver to the device. This verification does increase the programming sequence time.

Handshake at 9600 baud

This option forces the driver to handshake at 9600 baud before changing to the baud rate selected on the **Serial Port** option page.

Note: There is no option for suppressing download. To initialize the debug session using the ADu84x protocol the memory must be erased. This is a security mechanism protecting user program code.

SERIAL PORT

The **Serial Port** page contains the communication options for the Analog Devices driver:

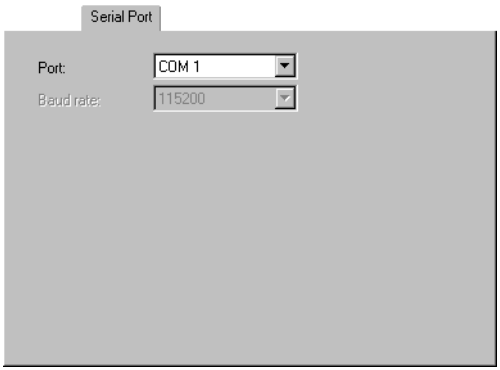

Figure 10: Analog Devices driver serial port options

Option	Setting
Port	One of the supported ports: `COM1, COM2, COM3, COM4`
Baud rate	`2400-115200`. If the option **Handshake at 9600 baud** is deselected the only available baud rate is `115200`.

Table 5: Analog Devices driver serial port options

C-SPY tries to connect at 9600 baud and then changes to the baud rate of the selected serial port when making the first contact with the evaluation board. If these options have not been specified, C-SPY will try using the COM1 port.

THE ANALOG DEVICES MENU

When you are using the Analog Devices driver, the **Analog Devices** menu appears in C-SPY.

The following commands are available on the **Analog Devices** menu:

Menu command	Description
Breakpoint Usage	Displays the Breakpoint Usage dialog box which lists all active breakpoints; see *Breakpoint Usage dialog box*, page 21.

Table 6: Commands on the Analog Devices menu

Silabs driver-specific debugging

In this section you can read about the features specific for the Silabs ROM-monitor driver:

DOWNLOAD

On the **Download** page you can control the download, the 2-wire interface that is used with C8051F3xx interfaces, and the flash page size.

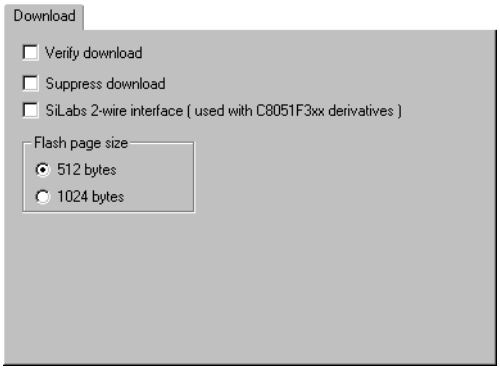

Figure 11: Silabs download options

Verify download

The option **Verify download** verifies that the program data has been correctly transferred from the driver to the device. This verification does increase the programming sequence time.

Suppress download

If you already have your application in flash memory, select **Suppress download**. It is highly recommended that you also select **Verify download** if you select **Suppress download**.

SiLabs 2-wire interface

The SiLabs C8051F3xx family uses the Cygnal 2-wire debugging interface (C2). You must select this option to be able to connect to a SiLabs C8051F3xx target device using an RS232 to 2-wire adapter between the host and the board.

Flash page size

With the **Flash page size** option you can select the size of the flash page, which can be either 512 or 1024 bytes.

SERIAL PORT

The **Serial Port** page contains the communication options for the Silabs driver:

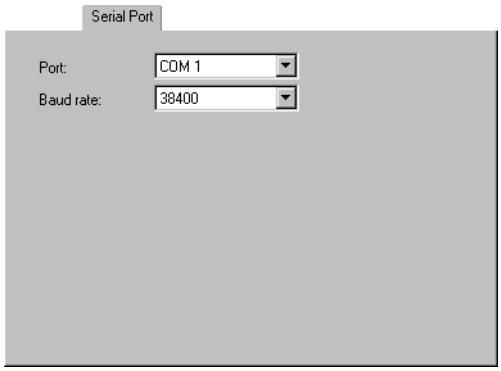

Figure 12: Silabs driver serial port options

Option	Setting
Port	One of the supported ports: `COM1, COM2, COM3, COM4`
Baud rate	`2400-115200`

Table 7: Serial Port options for the Silabs driver

C-SPY tries to connect with the selected serial port when making the first contact with the evaluation board. If these options have not been specified, C-SPY will try using the COM1 port.

THE SILABS MENU

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

The following commands are available on the **Silabs** menu:

Menu command	Description
Breakpoint Usage	Displays the Breakpoint Usage dialog box which lists all active breakpoints; see *Breakpoint Usage dialog box*, page 21.
Cache code memory	Enables memory changes in code memory during execution/debugging. This speeds up debugging and only needs to be turned off if you want to verify code written in code memory.

Table 8: Commands on the Silabs menu

Using breakpoints

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

- *Available breakpoints*, page 19
- *Breakpoint Usage dialog box*, page 21.

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

AVAILABLE BREAKPOINTS

Using the C-SPY drivers for hardware debugger systems you can set *code breakpoints* and for some drivers you can also set *data breakpoints*. The amount of breakpoints you can set in C-SPY heavily depends on the target system you are using. For some target systems, the number of breakpoints you can set depends on the number of available *hardware breakpoints*. A hardware breakpoint is set using hardware support on the chip. This will not affect the program code. The disadvantage is that the number of breakpoints are usually very few.

For information about the breakpoints supported in the different drivers, see *Table 1, Driver differences*, page 2.

A software breakpoint instruction temporarily replaces the application code with an instruction that handles the execution over to the ROM-monitor. There are two common ways to do this:

- Inserting an `LCALL #monitor` instruction
- Using the opcode `0xA5` to switch from the executing application to the ROM-monitor.

Padding for safe insertion of breakpoint instruction(s)

When using `LCALL` as code breakpoints extra memory space might be needed to avoid overwriting application memory. In an assembler program this has to be done manually. In C programs you can use the compiler option `__rom_mon_bp_padding`.

To set the equivalent option in the IAR Embedded Workbench IDE, choose the compiler option **Code>Padding for ROM-monitor breakpoints**.

Using this option makes it possible to set a breakpoint on every C statement.

Breakpoints in flash memory

When you set a software breakpoint in flash memory the driver will need to flash the page(s) containing the breakpoint instruction byte(s) once.

If you set a conditional breakpoint the driver has to flash the page(s) every time the breakpoint is not taken to check if the condition is met.

Every step you take on C-level will force the driver to temporarily set breakpoints on each possible endup statement.

Note: The Analog Devices driver will cache breakpoints, and it will not flash the page until the execution has started.

Periodically monitoring data

Breakpoints can only be set to occur during an instruction fetch. However, C-SPY provides a non-realtime data breakpoint mechanism, which lets you periodically monitor data without using data breakpoints. For a description of the data breakpoint mechanism, see the chapter *Using breakpoints* in the *IAR Embedded Workbench® IDE User Guide*.

BREAKPOINTS AND INTERRUPTS

If an interrupt becomes active when C-SPY is processing a breakpoint, C-SPY will stop at the first instruction of the interrupt service routine.

BREAKPOINT USAGE DIALOG BOX

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

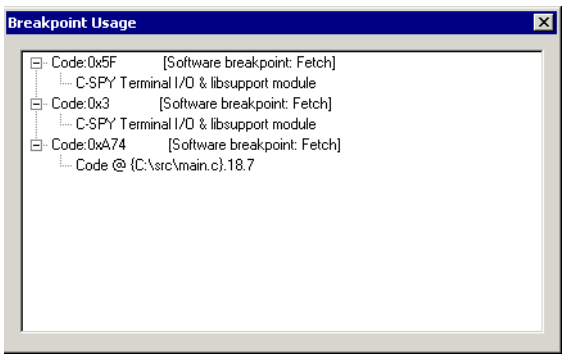

Figure 13: Breakpoint Usage dialog box

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*.

Troubleshooting

This section includes suggestions for resolving the most common problems that can occur when debugging with the IAR C-SPY Debugger in conjunction with a ROM-monitor program.

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

MONITOR WORKS, BUT APPLICATION WILL NOT RUN

Your application is probably linked to some illegal code area (like the interrupt table). Examine the linker command file and verify that the start addresses of `CODE` and `DATA` segments are correct.

Make sure you disable the watchdog timer if it is not used. Typically this should be done in the `__low_level_init` routine. Otherwise the application program will restart, which would lead to unexpected behavior.

NO CONTACT WITH THE TARGET HARDWARE

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

- Check the communication devices on your host computer
- Verify that the cable is properly plugged in and not damaged or of the wrong type
- Make sure that the evaluation board is supplied with sufficient power
- Check that the correct options for communication have been specified in IAR Embedded Workbench; see the *Serial Port* section for the driver you are using.

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

NO CONTACT WITH THE MONITOR

There are several possible reasons for C-SPY can fail to establish contact with the ROM-monitor program.

- Try a lower baud rate.
- A protocol error could have occurred. Try resetting your evaluation board and restart C-SPY.
- Check that the correct options for serial communication have been specified in the IAR Embedded Workbench. See the corresponding sections for the appropriate driver.
- Verify that the serial cable is properly plugged in and not damaged or of the wrong type.