AVR ONE!

Quick-start Guide

EVK1100 + Windows®

Section 1

Introduction

1.1 General

This document contains a quick-start guide describing how to get up and running using the AVR® ONE! debugger with AVR32 Studio. In addition to the AVR ONE! debugger, you need the following items:

- AVR32 Studio 2.5 software
- AVR32 GNU Toolchain 2.4
- EVK110x Evaluation board

Software and documents can be found at www.atmel.com/avrone

1.2 Requirements

This example was created on a PC running Microsoft® Windows® XP Professional. For other versions of Windows, the behaviour when installing software and drivers may be slightly different.

Please read the AVR32 Studio 2.5 release notes for information about support for other versions of Windows.

Section 2

Quick-start guide (short version)

2.1 Install Hardware and software

- Install the MICTOR38 connector on the EVK1100 board.
- Download and install avr32-gnu-toolchain-2.4.x and AVR32Studio-2.5.x.
- Connect AVR ONE! to power and USB and turn it on.
- Install AVR ONE! USB driver.
- Connect AVR ONE! to the EVK1100 using the MICTOR38 connector.
- Connect the EVK1100 to power and turn it on.
- Start AVR32 Studio.
- Select a suitable workspace folder to contain your projects.
- Exit from the welcome screen to workbench.
- Right-click in the *AVR32 Targets* view and select **Scan Targets**.
- Select the AVR ONE! and click on the *Properties*-tab.
- Select *Details*-tab. Set MCU to **UC3A0512** or **UC3A0512ES**, depending on what MCU is mounted on your EVK1100 and Board to **EVK1100**,.
- Right-click on the AVR ONE! in the *AVR32 Target* view and select **Chip Erase**. This operation is only needed one time (when the EVK1100 is new).

2.2 Create a demonstration project

- *Select File>New>Example*.
- Select *EVK1100>Components>DIP204 example*, then **Next**.
- Enter a name for the project, and click **Finish**.
- Right-click on the project in *Project Explorer* view and select **Build Project** (or use Ctrl+B).

2.3 Configure target MCU for a debug session using trace

- When the build process is finished, right-click on the project in the *Project Explorer*-view and select *Debug As>Debug Configurations*.
- In the *Debug Configurations*-view, select **AVR32 Application** and click **New**. A new launch configuration will be created and default values will be filled into all fields.
- Select the *Trace*-tab and click **Enable Trace**.
- Select the preferred trace method. In this case we want **Buffered AUX Trace**.
- Select the preferred action when buffer is full. In this case we choose **Break, read out and halt**.
- Select Buffer Size. We use **16kB** for a quick test.

- Select Debugger tab and tick *Stop on startup at: main*.

2.4 Start the debug session and configure AVR32 Studio 2.5 for trace

- Click the **Debug**-button. Now the program will be loaded into the target, and run until `main()`.
- When the program halts, add at least a trace start-point (Right-click to the left of the source code line in the source code view).

2.5 Start the trace debug session

- Click **Resume** (green *Play* button in Debug view) and wait until the program halts.
- You can now look at the trace data in the *Trace*-view.

Section 3

Hardware preparation

In case you have an evaluation kit without the MICTOR38 connector, you need to install one. In case the connector is already mounted, you can skip this chapter.

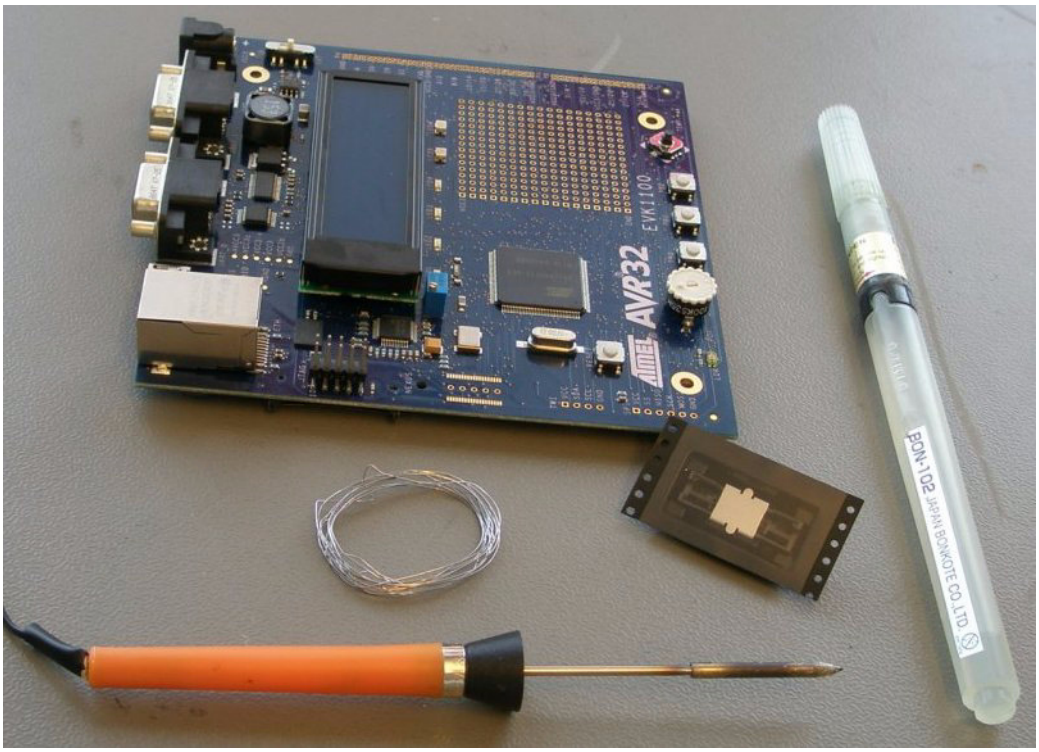
To be able to connect to the evaluation board AUX port, you need to solder a connector to the board. The AVR ONE! Kit contains one MICTOR38 connector for this purpose. If you need more connectors for other kits, or your own designs, you can buy more connectors from Atmel, or Tyco Electronics/AMP.

The Tyco Electronics/AMP Part number is 2-5767004-2.

To install the MICTOR38 connector, you only need a fine-tipped soldering iron, a small piece of fine solder (0,3mm is OK), and some extra flux. Also remember to provide proper ventilation to prevent inhaling the fumes from the flux.

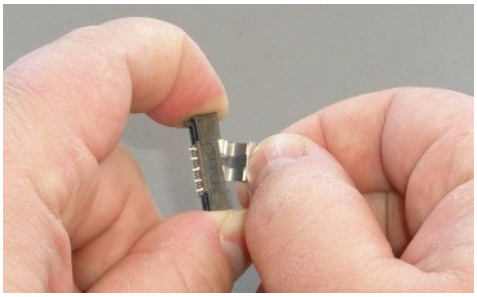
The soldering guide shows the EVK1100, but is applicable for all other kits that needs a MICTOR38 connector (like the EVK1101).

Figure 3-1. Required hardware and tools for installation of Mictor38



Unpack the Mictor38 connector and remove the pick-and-place pad

Figure 3-2. Remove the pick-and-place pad

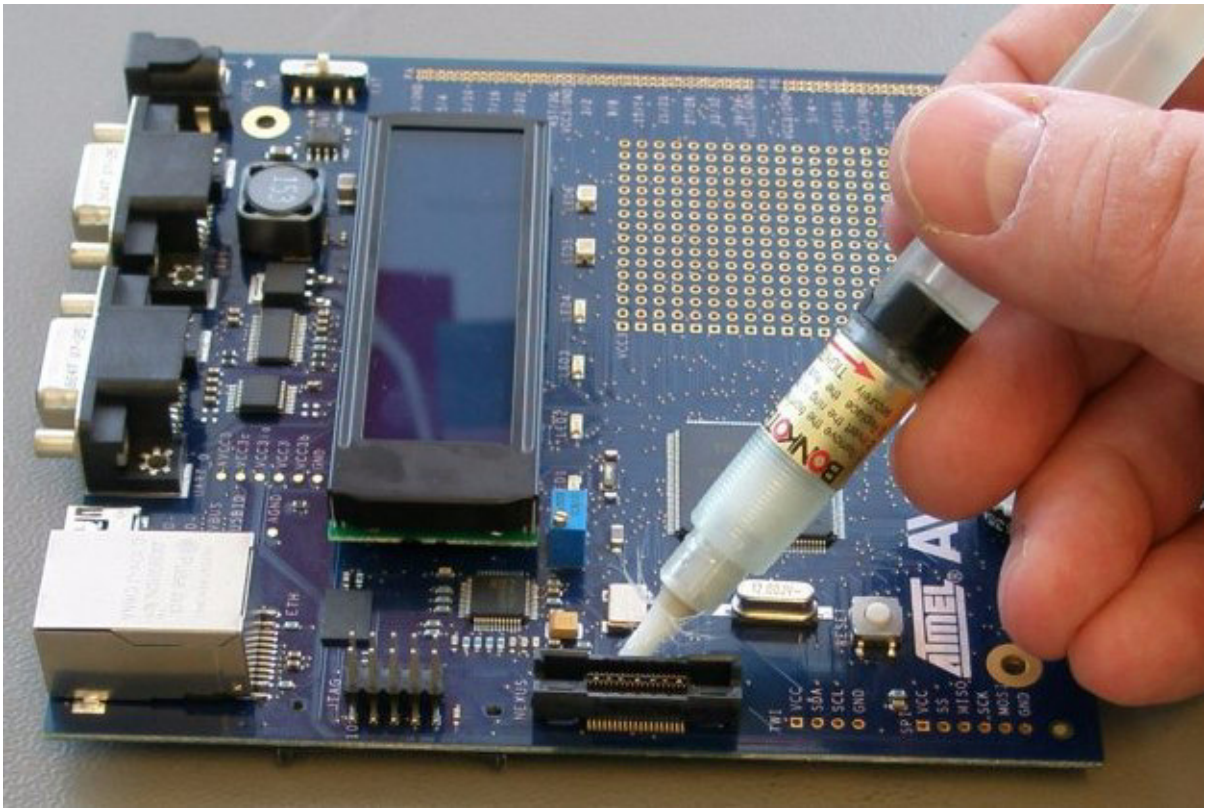


Place the connector onto the footprint on the evaluation board. Make sure that the guide tab beneath the connector fits into the guide hole in the PCB.

Add a fair amount of flux. The extra flux is very important for a good result. It is also very important to keep the tip of the soldering iron clean while mounting the connector.

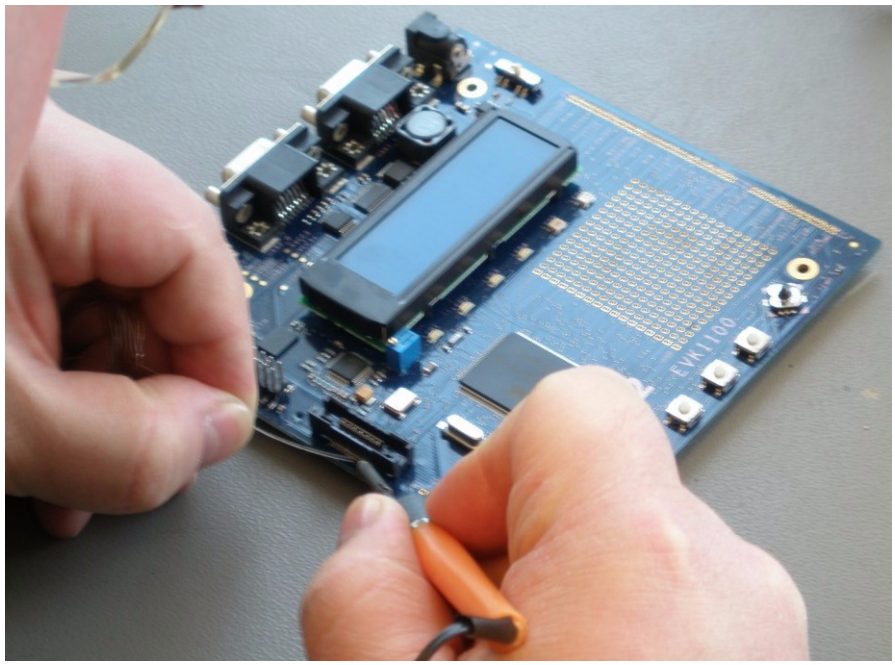
Too thick solder, too little flux or solder-blobs on the tip of the soldering iron will give bad connections or short circuits.

Figure 3-3. Place Mictor38 and apply flux



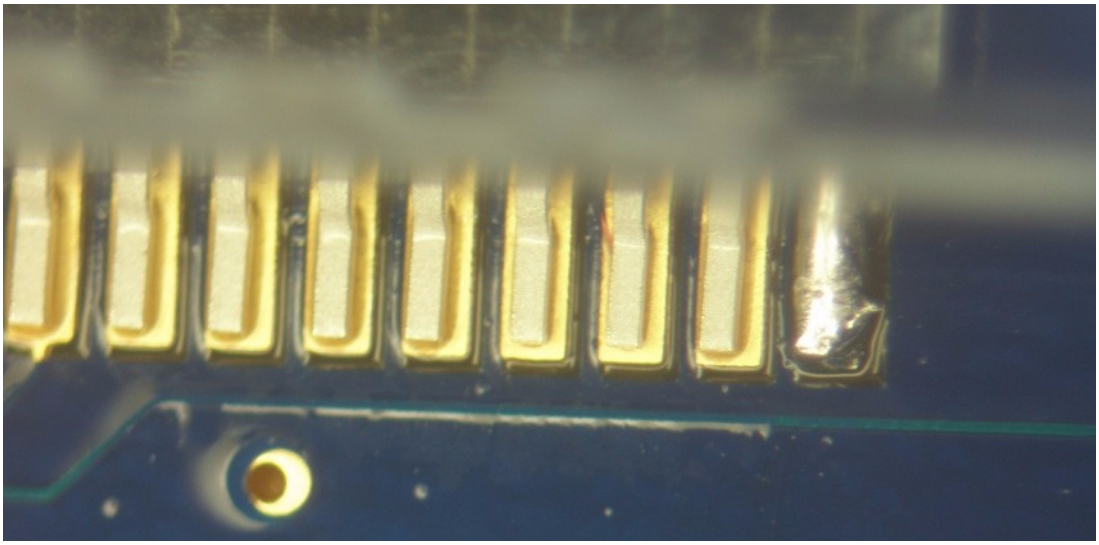
Make sure that the connector is firmly seated on the footprint, and start by soldering the corners.

Figure 3-4. Soldering the corners



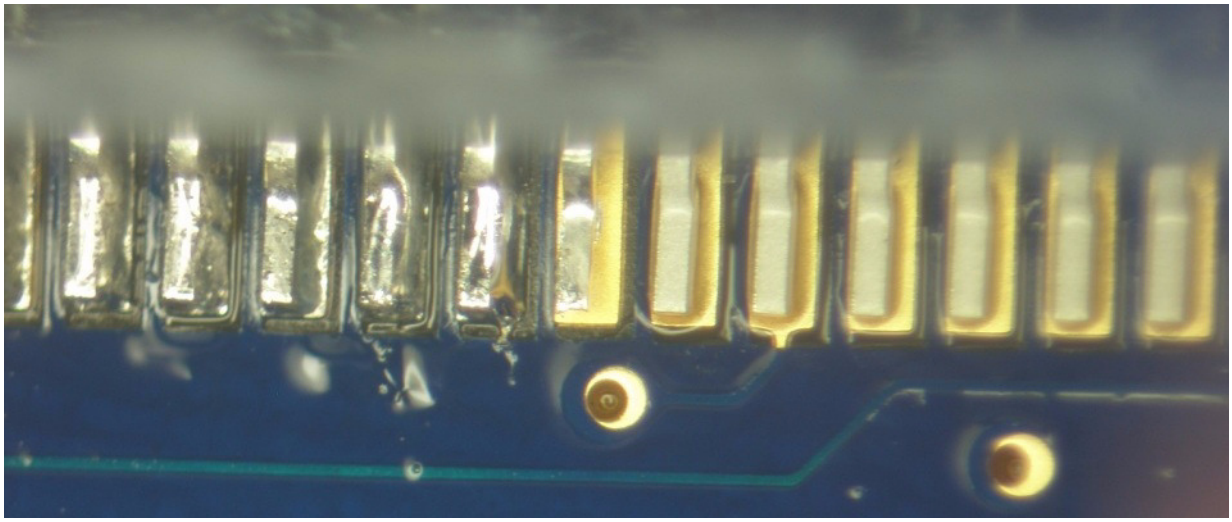
When all corners are soldered, check that connector is still firmly seated. It is still possible to push the connector down and re-heat corner pins if you need to adjust a bit.

Figure 3-5. Soldered corner pin



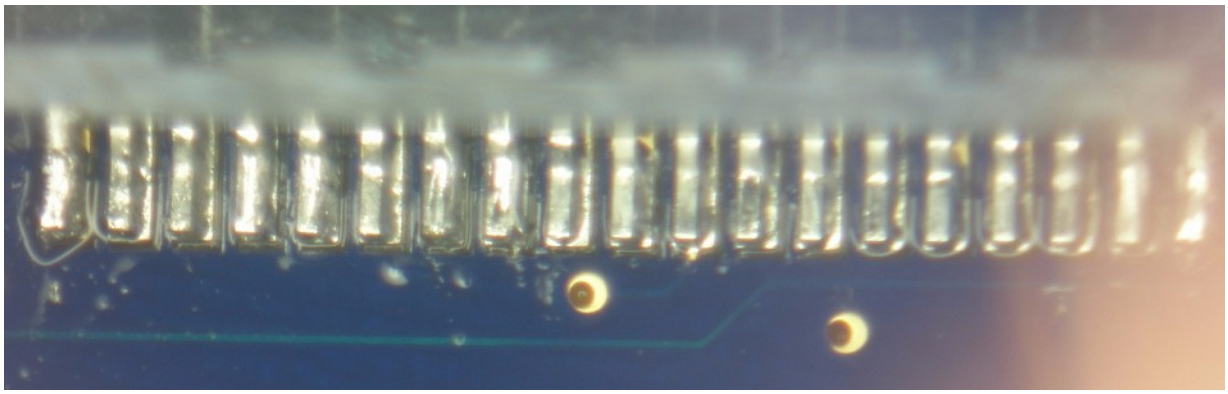
Solder the remaining pins.

Figure 3-6. Continuing with the remaining pins



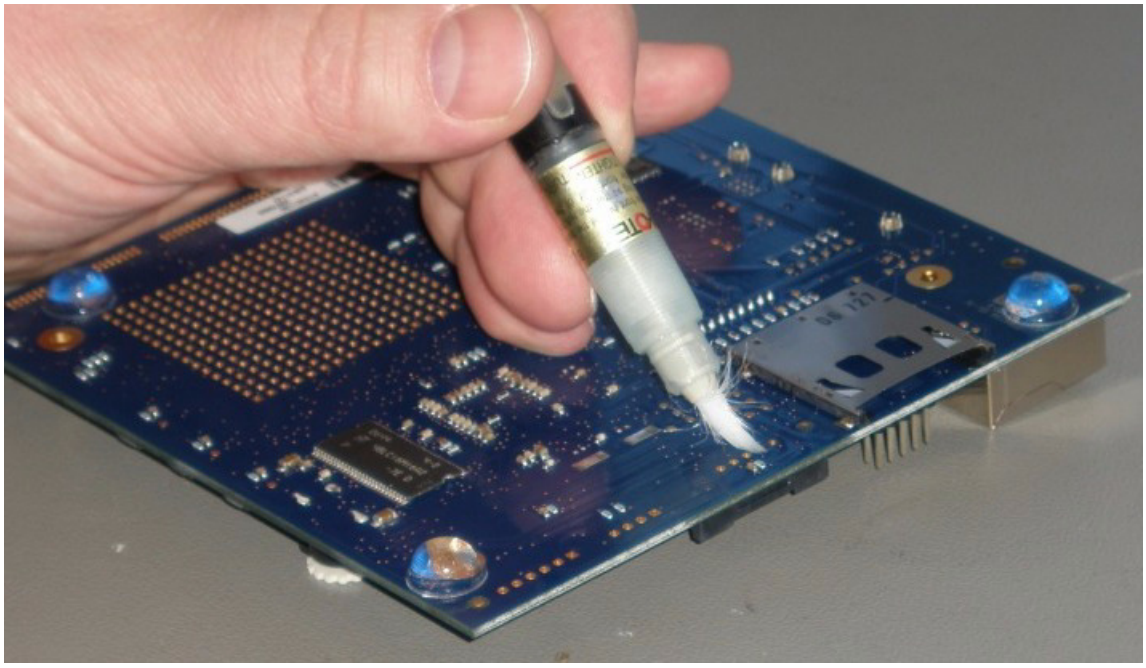
After soldering, you should make sure that there are no shorts circuits between pins.

Figure 3-7. All pins finished



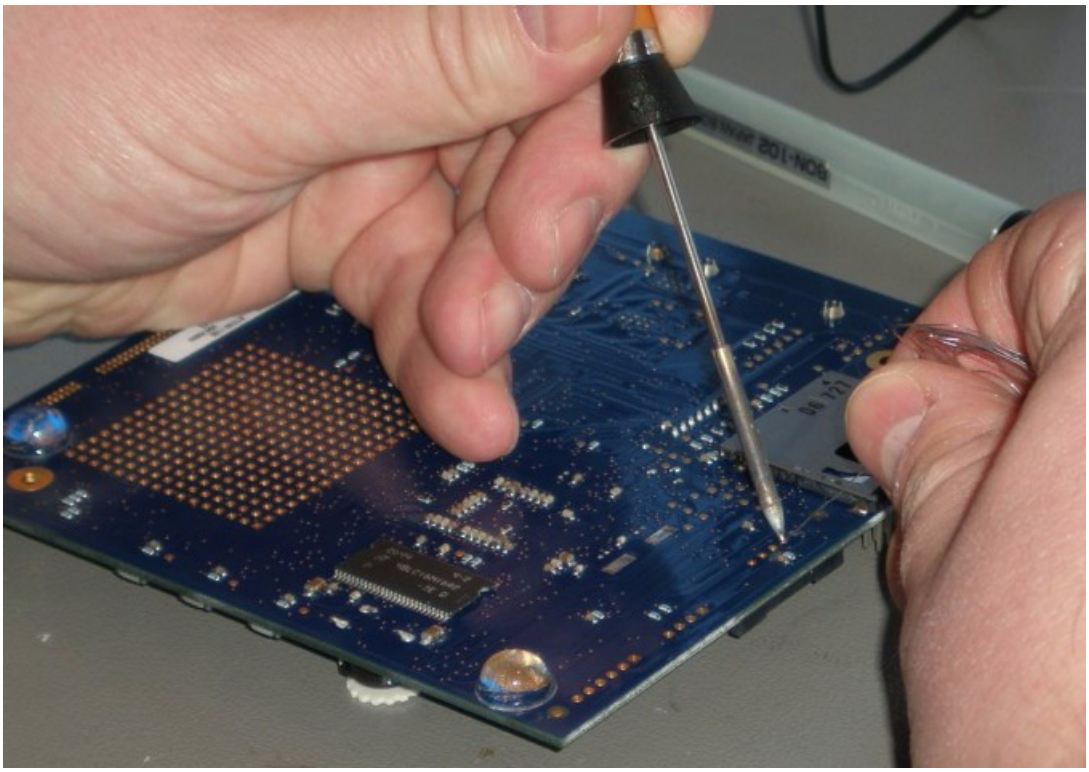
Turn the board and apply flux on the ground pins.

Figure 3-8. Apply flux on ground pins



Solder the five ground pins.

Figure 3-9. Solder ground pins



Section 4

Software Installation

4.1 Download the software

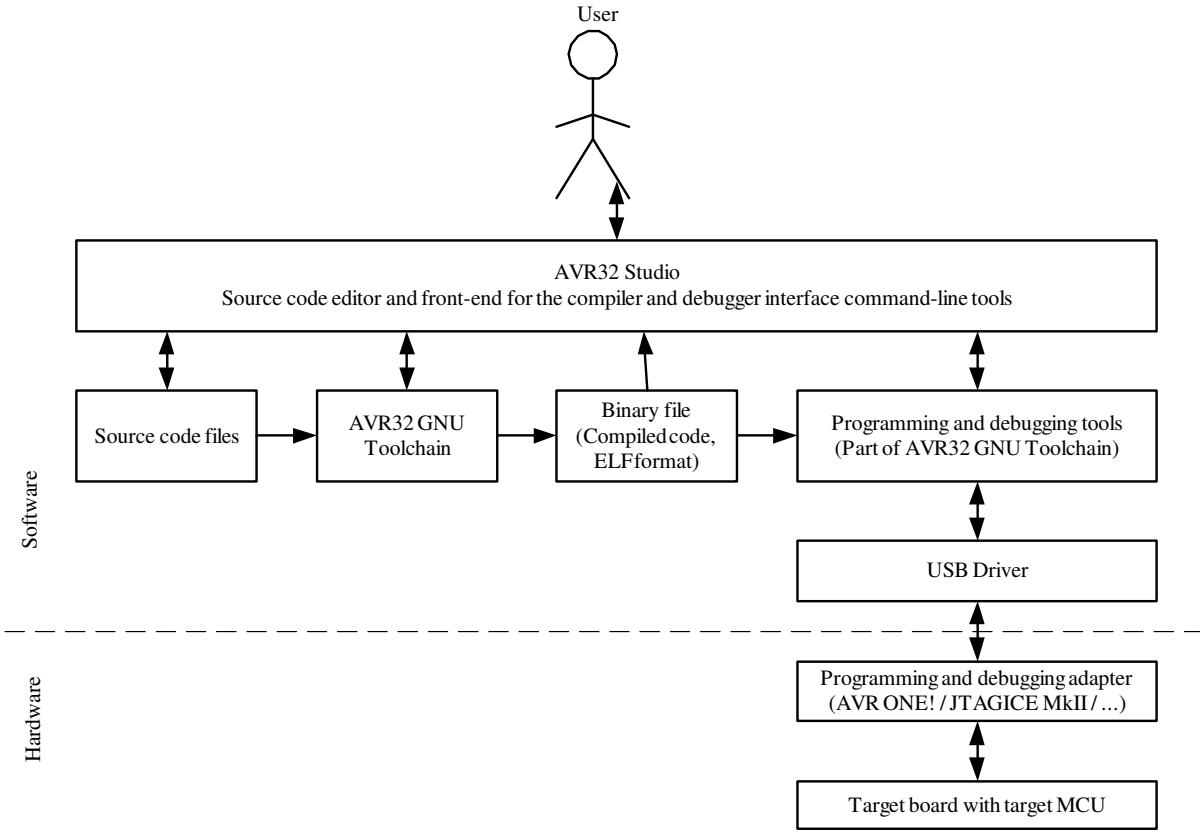
To use the AVR ONE!, you must download and install two software packages:

- avr32-gnu-toolchain-2.4.x.exe
- AVR32Studio-2.5.x.exe

The AVR32 Toolchain is a collection of tools that are required to be able to work with the AVR ONE! It contains command-line tools for controlling the AVR ONE!, and tools to compile code for the AVR32 MCUs.

AVR32 Studio is the front end that uses the AVR32 GNU Toolchain to generate binary code for the target, program the target, and control the debug sessions.

Figure 4-1. Tools structure



4.2 Download the two installation files to your disk.

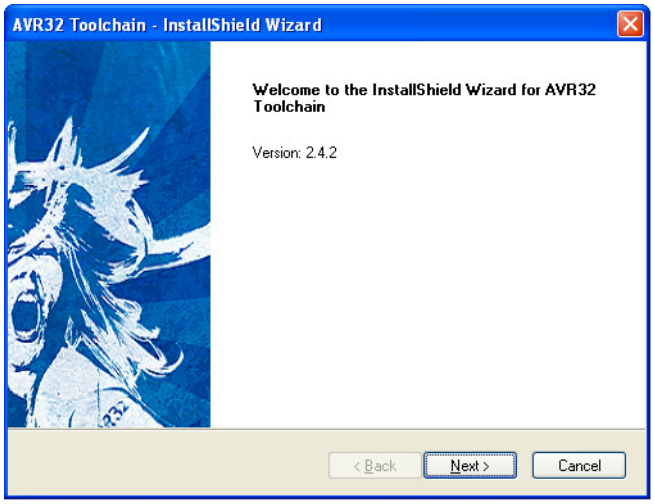
The installation files can be found at this location: **www.atmel.com/avrone**

4.3 Install AVR32 GNU Toolchain

If you have any AVR tools connected to the USB hub, turn them off now. Otherwise the USB driver installation may fail.

Double-click on avr32-gnu-toolchain-2.4.x to start the installation process.

Figure 4-2. AVR32 GNU Toolchain installation welcome

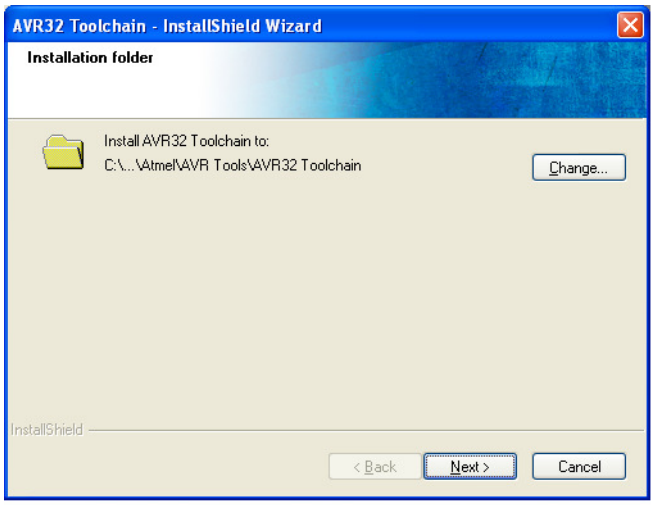


Click **Next**.

Figure 4-3. AVR32 GNU Toolchain License Agreement form

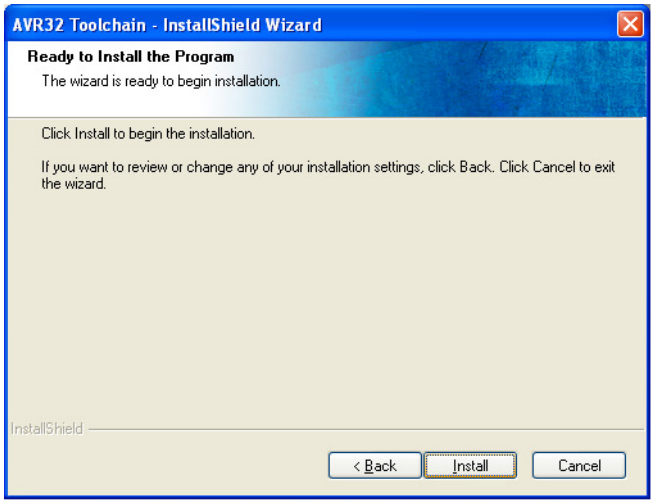
Select **I accept the terms of the licence agreement**, then click **Next**.

Figure 4-4. AVR32 GNU Toolchain installation folder select



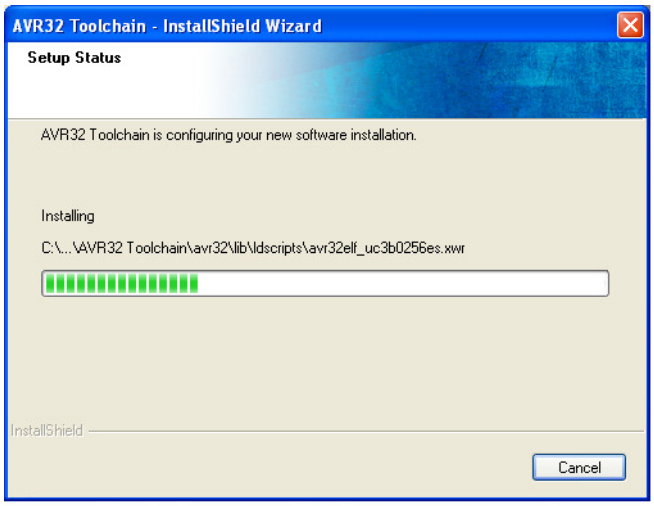
Check that the installation folder is correct and click **Next**.

Figure 4-5. AVR32 GNU Toolchain installer configuration finished



Click **Install**.

Figure 4-6. AVR32 GNU Toolchain installation progress indicator



The AVR32 GNU Toolchain is now being installed. As a part of the installation process, USB drivers for all supported programming and debugging adapters are installed.

Figure 4-7. USB Drivers installation start

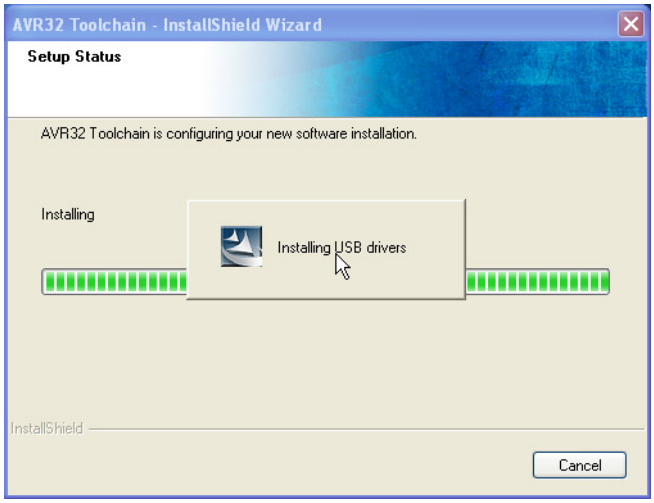
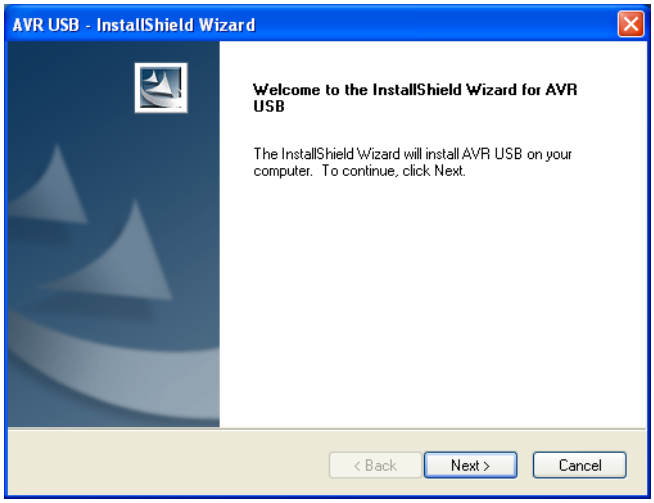
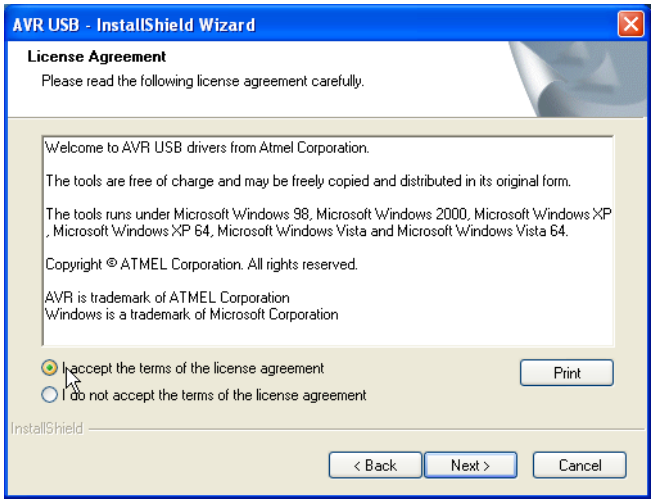


Figure 4-8. USB Driver installer welcome



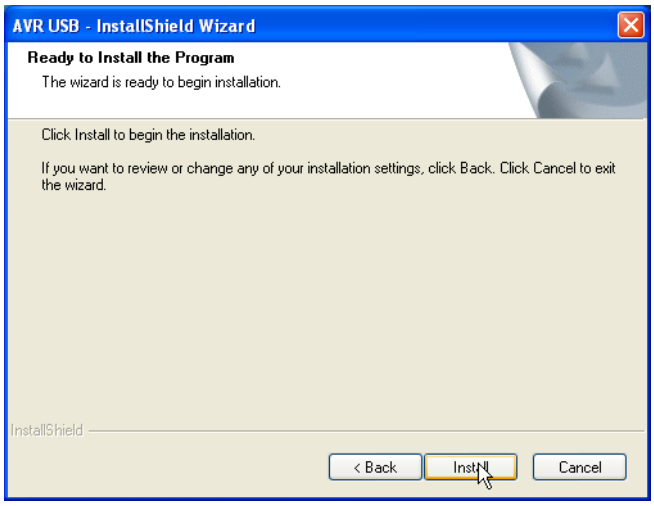
Click **Next**.

Figure 4-9. USB Drivers licence agreement form



Select **I accept the terms of the licence agreement**, then click **Next**.

Figure 4-10. USB drivers installer configuration finished



Click **Install**.

Figure 4-11. USB Drivers installation progress indicator

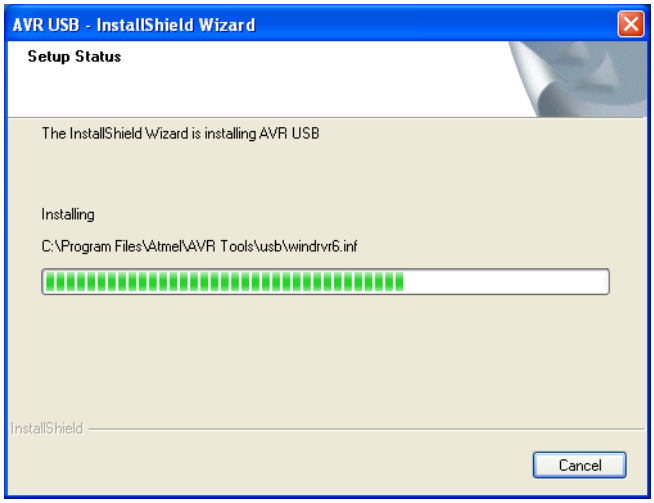
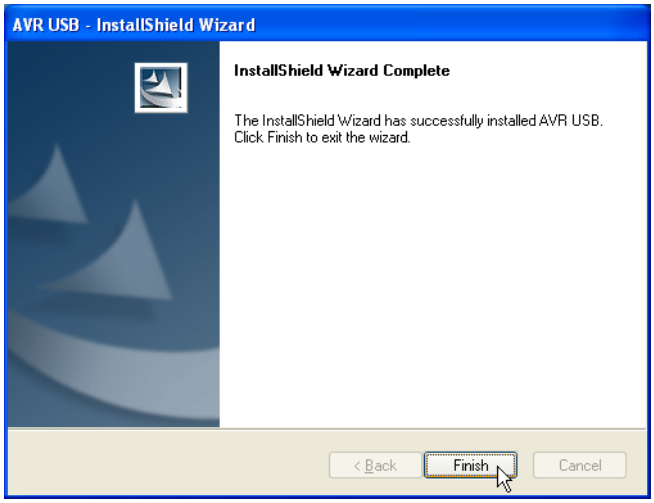


Figure 4-12. USB Drivers installation complete



Click **Finish**.

Figure 4-13. AVR32 GNU Toolchain installation complete

Click **Finish** to complete the AVR32 Toolchain installation process.

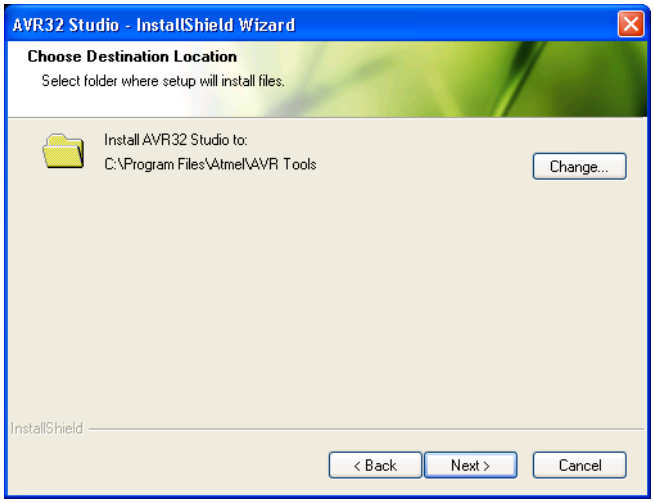
4.4 Install AVR32 Studio 2.5

Double-click on the AVR32Studio-2.5.x.exe file to start the installation process.

Figure 4-14. AVR32 Studio 2.5 installer welcome

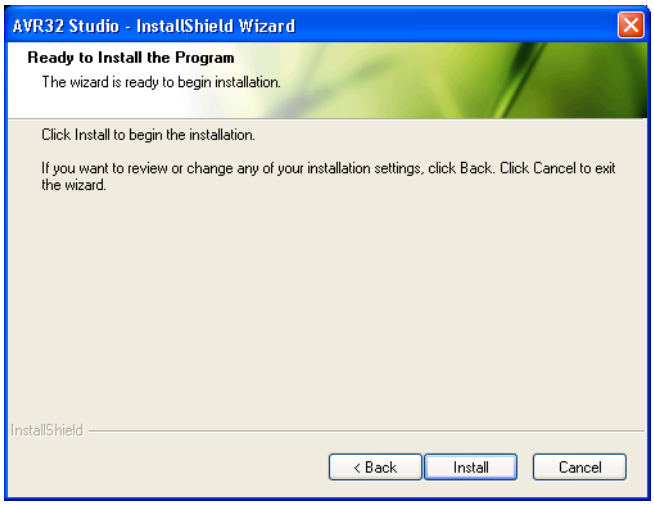
Click **Next**.

Figure 4-15. AVR32 Studio installation folder select



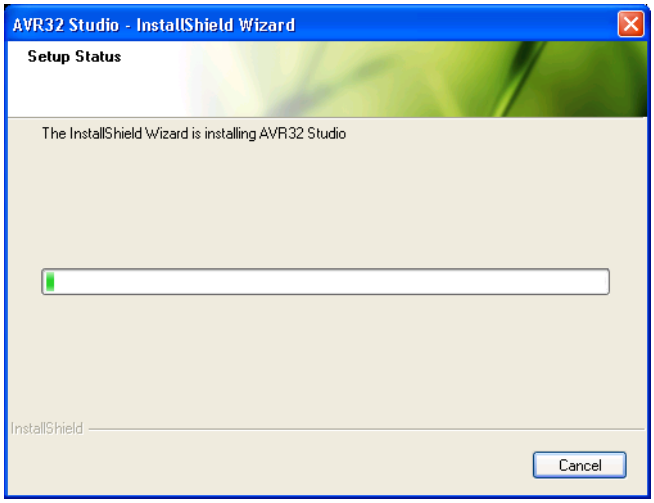
Check that the installation folder is correct and click **Next**.

Figure 4-16. AVR32 Studio installer configuration finished



Click **Install** to start the installation.

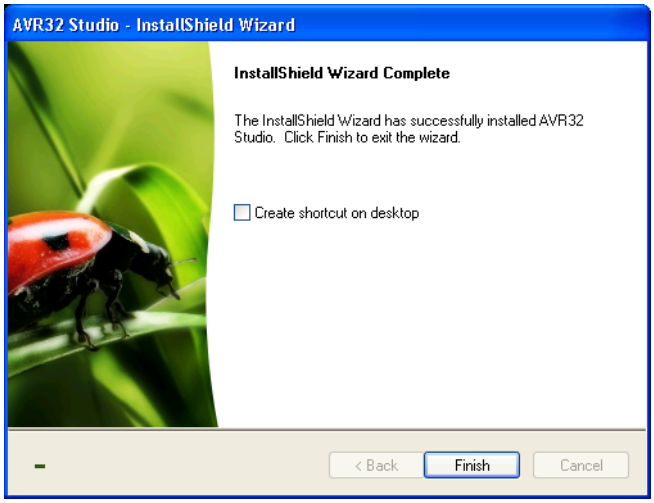
Figure 4-17. AVR32 Studio installation progress indicator



Wait for the installation process to complete.

If a suitable Java™ runtime is not installed, a Java installer wizard will guide you through the installation procedure.

Figure 4-18. AVR32 Studio installation process complete



Tick **Create shortcut on desktop** if you want a shortcut to be created. Then click **Finish**.

4.5 Connect the AVR ONE! to power and USB host

- Connect the AVR ONE! to power using the supplied power supply.
- Connect the AVR ONE! to the USB host (PC) using the supplied USB cable
- Turn on the AVR ONE! using the power switch next to the power connector

Figure 4-19. AVR ONE! connected to power and USB

4.6 Install AVR ONE! Driver

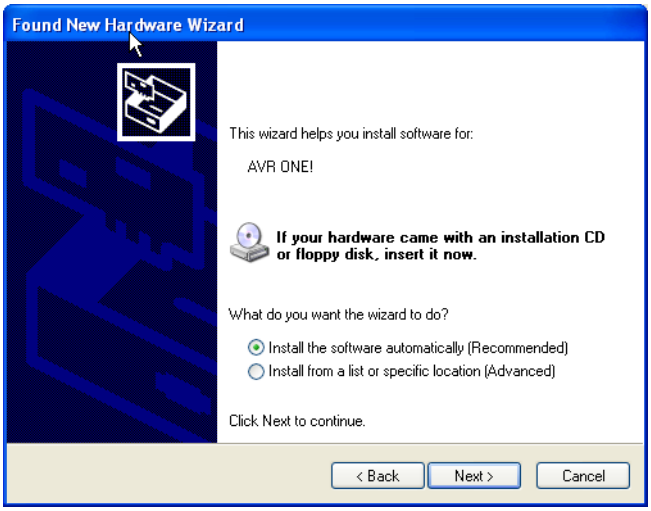
When the AVR ONE! is powered up and connected to the PC for the first time, the proper USB driver must be installed. Since the PC is keeping track of the serial number of each USB device, this will happen every time a new AVR ONE! is connected to the PC, even if the driver is the same as for all other AVR ONE!s that have been connected previously. This is a property of the operating system, and is not controlled by any Atmel software installed.

Figure 4-20. "New hardware" notification pop-up

Figure 4-21. AVR ONE! Hardware installation wizard

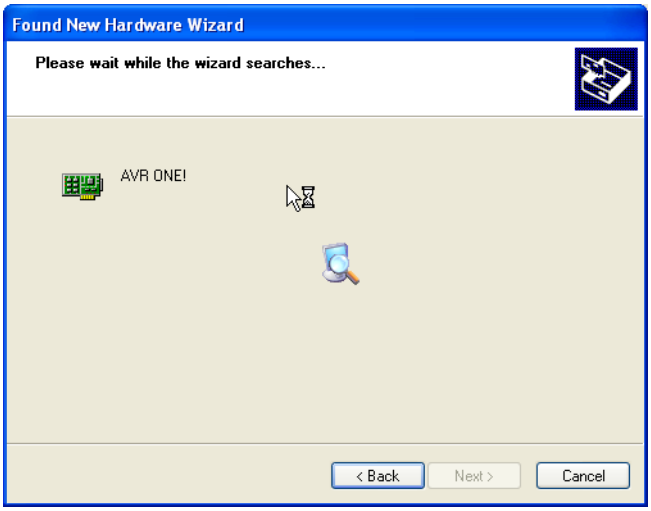
When the hardware installation wizard pops up, select **No, not this time** and click **Next**.

Figure 4-22. Hardware installation wizard configuration



Select **Install the software automatically** and click **Next**.

Figure 4-23. Hardware installation in progress



Wait for the installation process to complete.

Figure 4-24. Hardware installation wizard complete

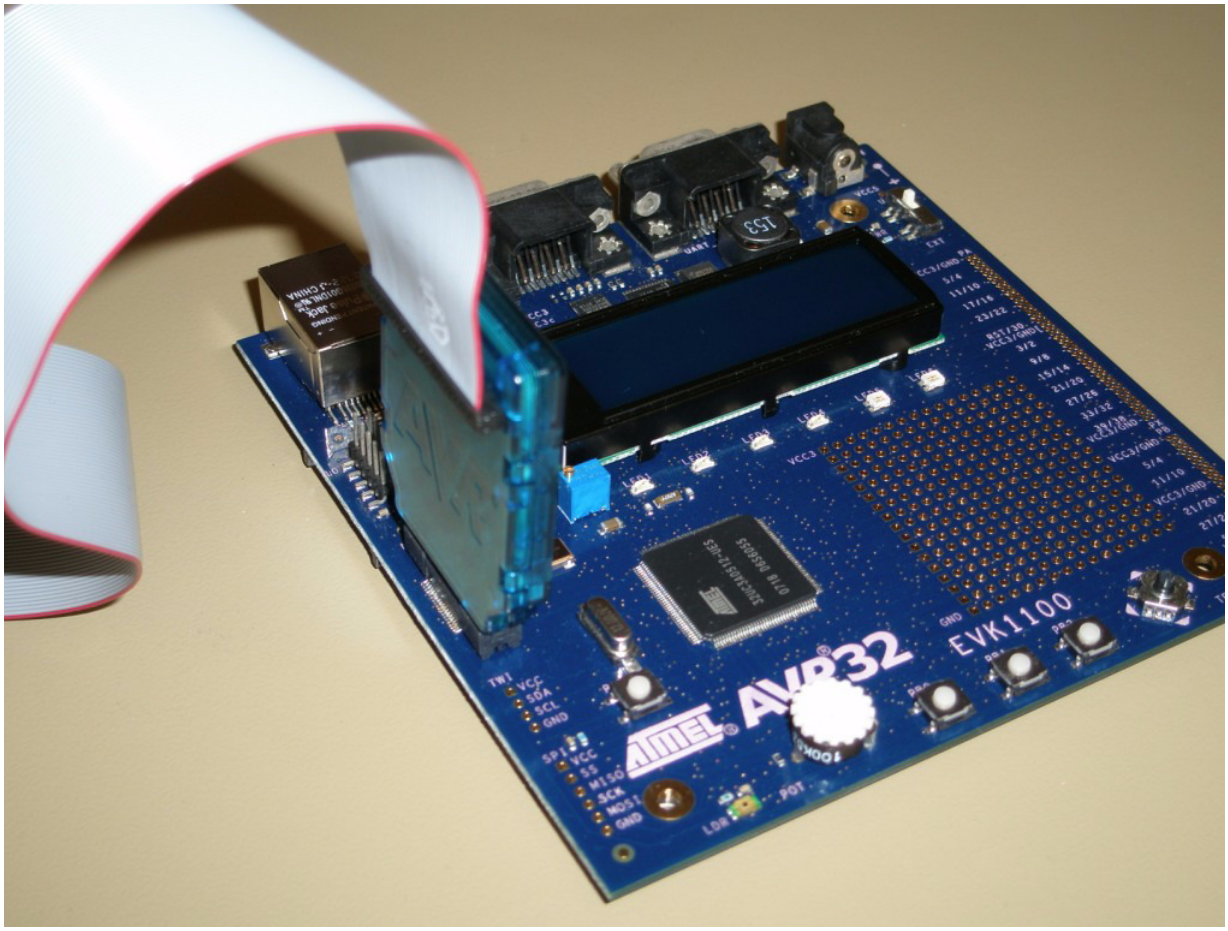
Click **Finish**.

Section 5

5.1 Connect the AVR ONE! to the EVK1100

Connect the AVR ONE! debugger to the EVK1100 evaluation board using the MICTOR38 connector.

Figure 5-1. AVR ONE! connected to the EVK1100

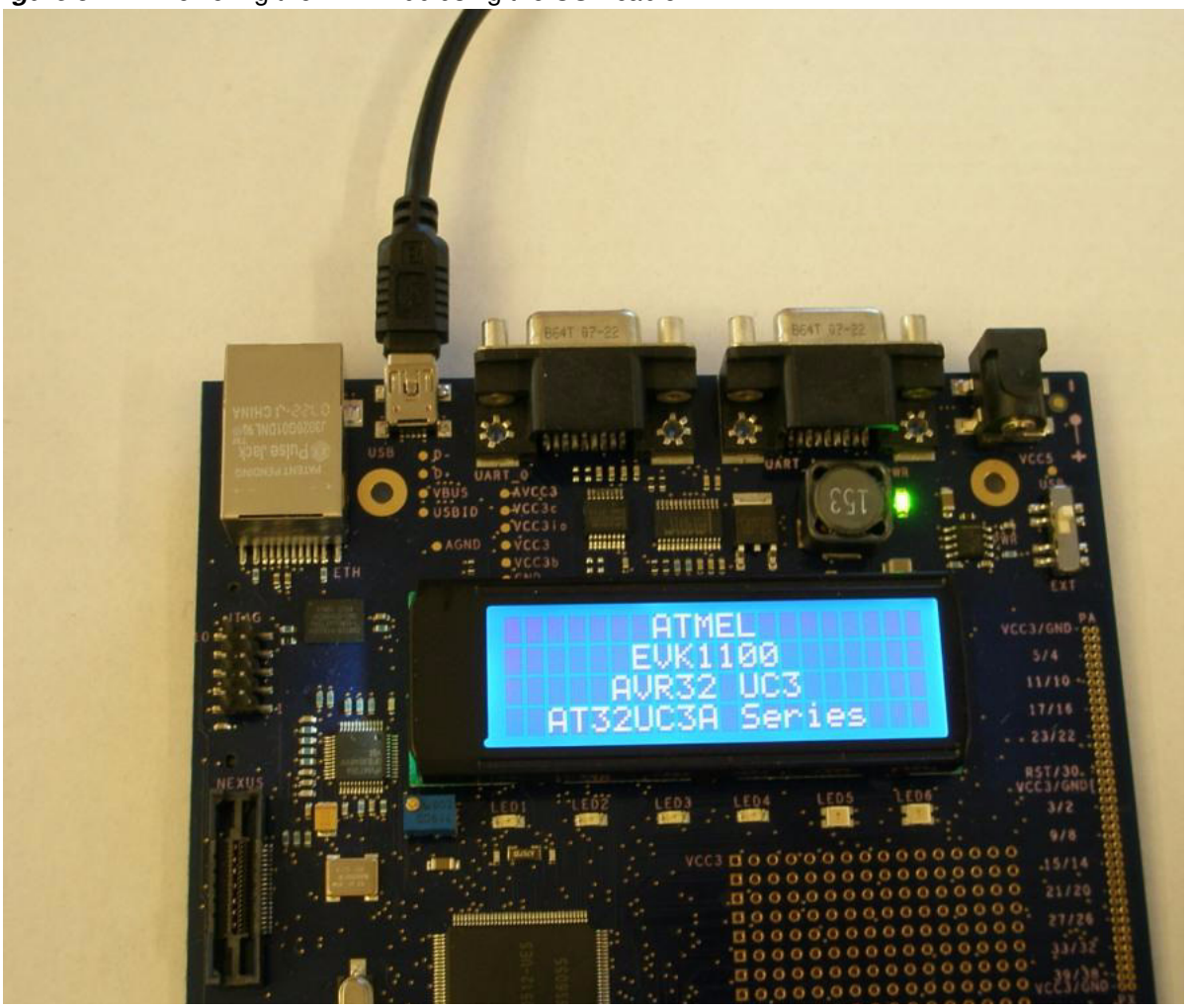


5.2 Connect the EVK1100 to power

Connect the EVK1100 to power and turn it on. The easiest way to provide power is to use the supplied USB cable.

Switch it on by setting the power switch to **USB**.

Figure 5-2. Powering the EVK1100 using the USB cable



Section 6

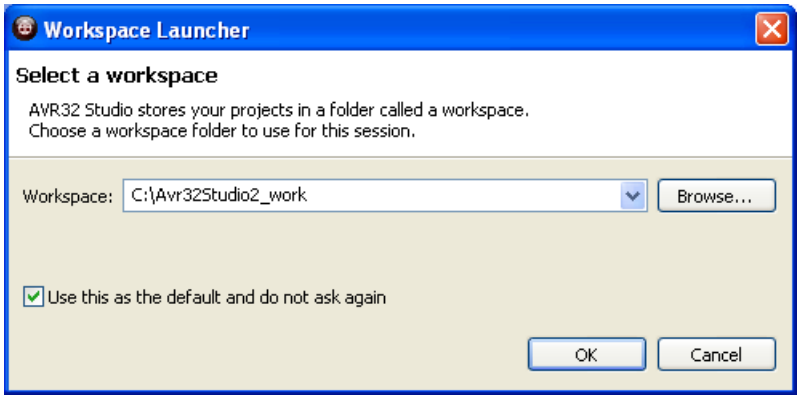
Create demo application

6.1 Start AVR32 Studio

Start AVR32 Studio. Start-up may take a while (because of all the Java libraries being loaded).

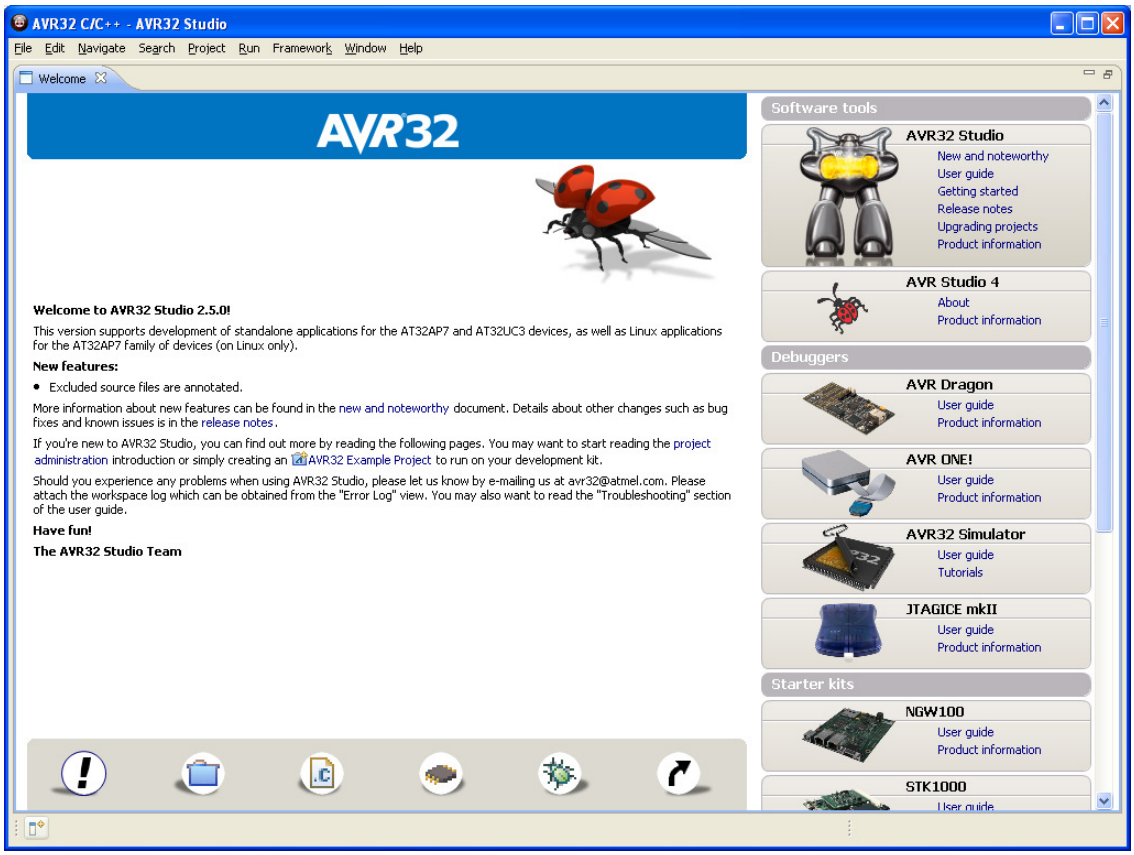
Figure 6-1. AVR32 Studio splash screen

Figure 6-2. AVR32 Studio workspace selection



Select a suitable workspace folder for your project files. If you want to use the same folder for your workspace every time you start AVR32 Studio, you should tick the box before clicking **OK**.

Figure 6-3. AVR32 Studio Welcome view



Exit from the welcome screen to the workbench by clicking on the **Close Page** icon (Arrow).

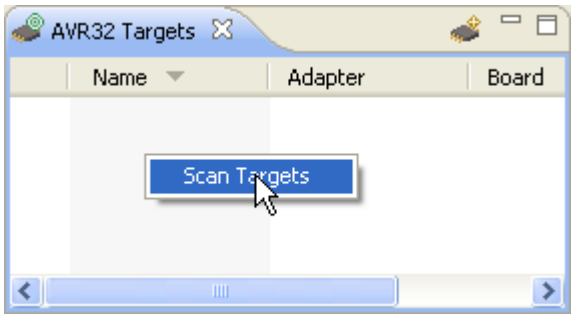
6.2 Configure adapter and target

Before you can use the AVR ONE! and the EVK1100, you have to tell AVR32 Studio what type of equipment is connected to your PC.

“Target” refers to the MCU on the EVK1100 evaluation board, and “Adapter” refers to the tool connecting the target to the PC (in this case, the AVR ONE!).

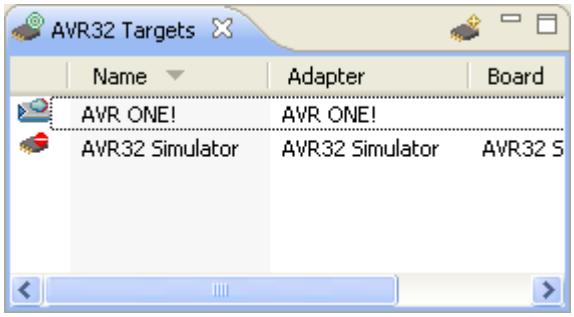
6.2.1 Add and configure the adapter (AVR ONE!)

Figure 6-4. Scan Targets



Right-click in the **AVR32 Target**-view and select **Scan Targets**.

Figure 6-5. Available targets



Select the AVR ONE!

Figure 6-6. AVR ONE! Selected

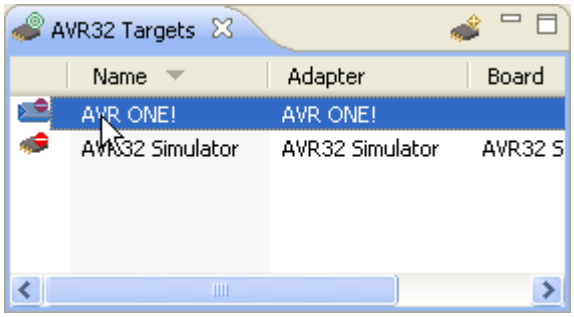
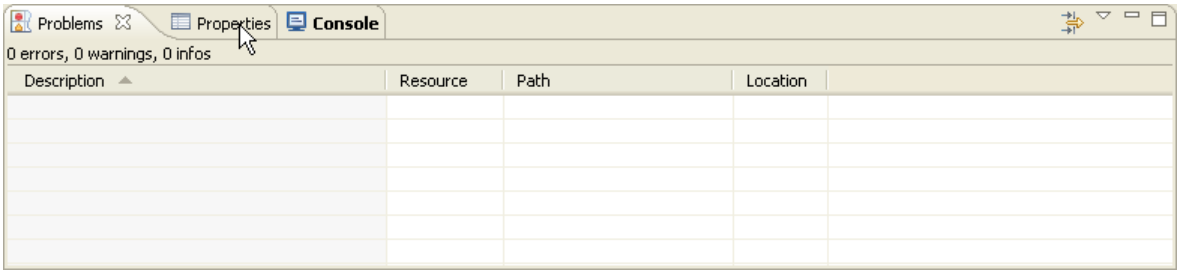
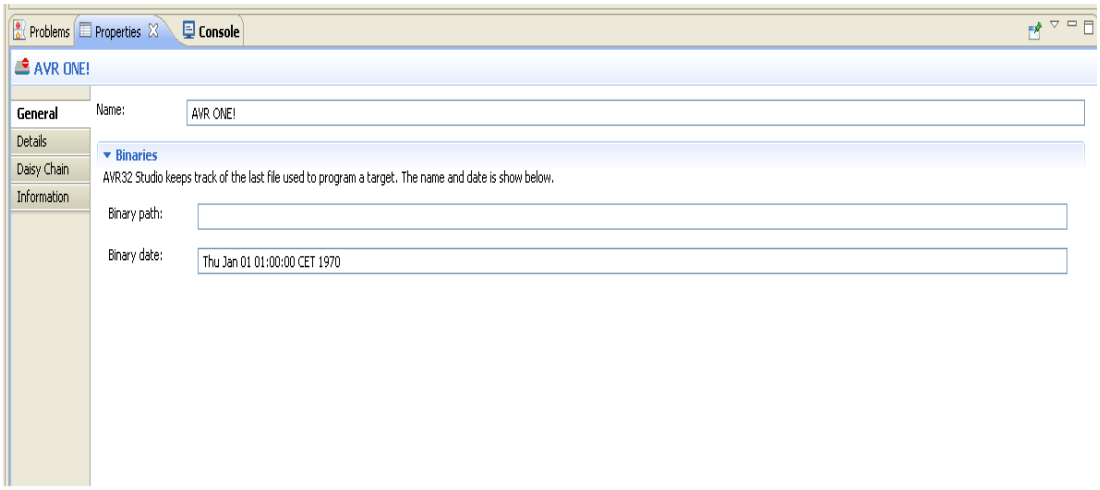


Figure 6-7. Selecting the properties view



Click on the **Properties** tab.

Figure 6-8. Properties view

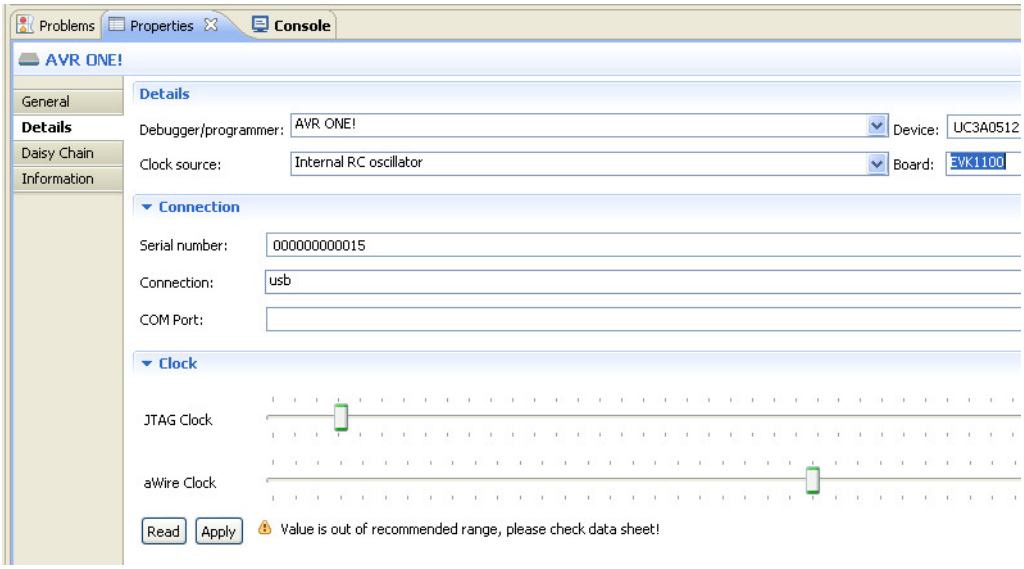


If you have several adapters connected at the same time, this is the place where you can give them unique names. Just type the name you want to use in the **Name** field.

6.2.2 Configure target board and MCU

Select the **Details** tab.

Figure 6-9. Setting the board and device type



Set MCU to **UC3A0512** or **UC3A0512ES**, depending on what MCU is installed on your EVK1100.

Figure 6-10. MCU Markings

To check which type of MCU is mounted on your EVK1100 evaluation board, you can read the part number printed on the MCU. The picture shows the part number printed on an -ES part (-UES suffix).

Set **Board** to **EVK1100**.

Set **MCU Clock source** to **Crystal**.

Adjust the JTAG Clock to a suitable value (Usually 33MHz or less. Max speed depends on target board signal quality). Click **Apply**.

The target and adapter configuration process is now complete.

6.2.3 Target MCU Chip erase

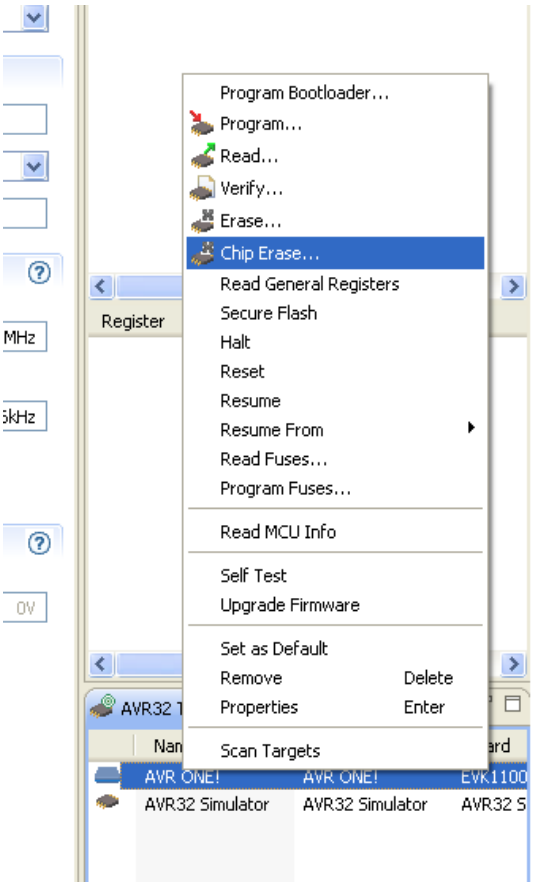
If the EVK1100 evaluation board is brand new, or if it still contains the original demo application (Control Panel Demo), the FLASH lock-bits need to be cleared. Right-click on the AVR ONE! In the *AVR32 Target* view and select **Chip Erase**.

WARNING! This process will erase the original demo application programmed at the factory. After this operation the EVK1100 evaluation board will be completely empty. If you need to keep the original application, you should not perform this operation.

If you would like to use your EVK1100 for this example, it is not difficult to restore the original “Control Panel Demo application”. All you have to do is to build the “Control Panel Demo example” enclosed with AVR32 Studio.

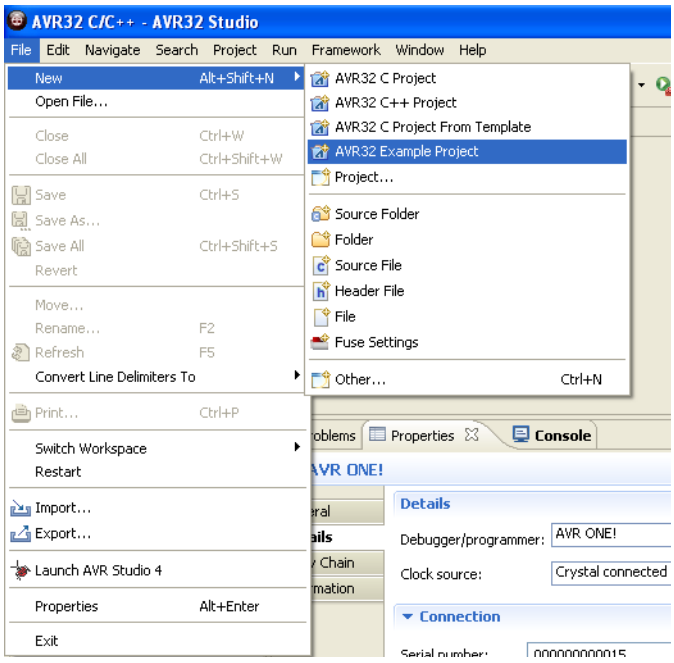
You should now perform the **Chip Erase** operation.

Figure 6-11. Chip erase operation



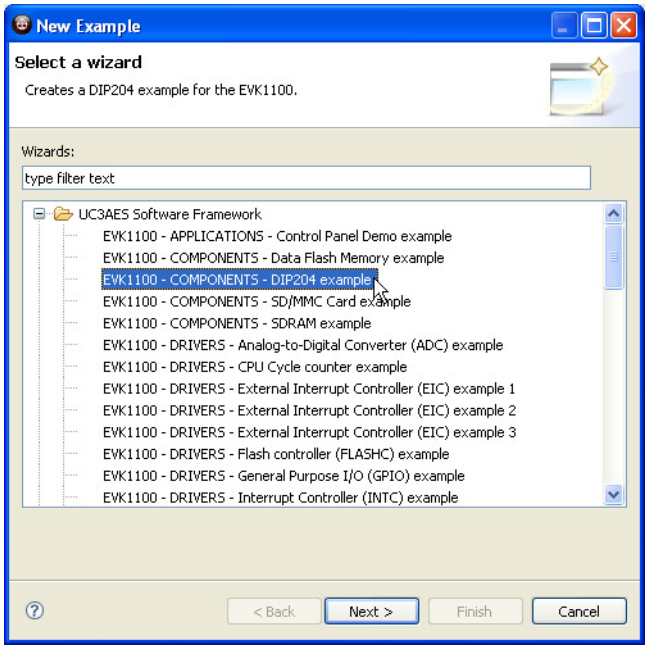
6.3 Create a demonstration project

Figure 6-12. Create new project



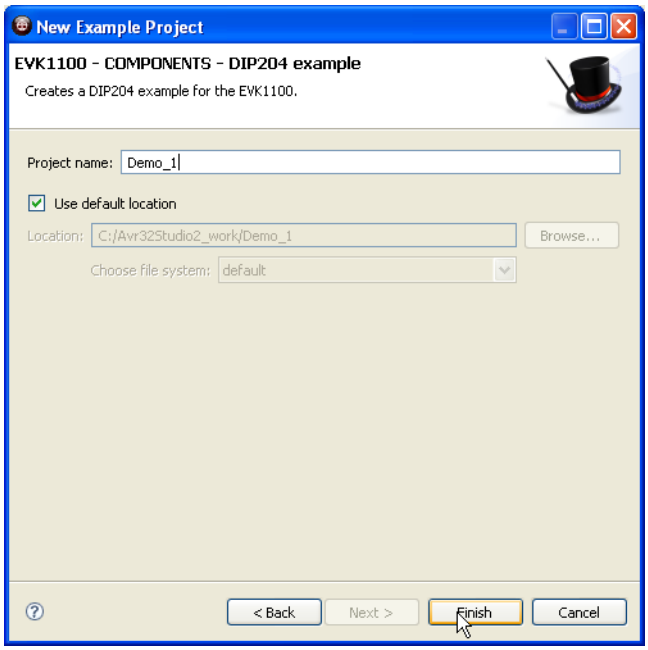
Create a new project by clicking *File>New>AVR32 Example Project.*

Figure 6-13. Select project example



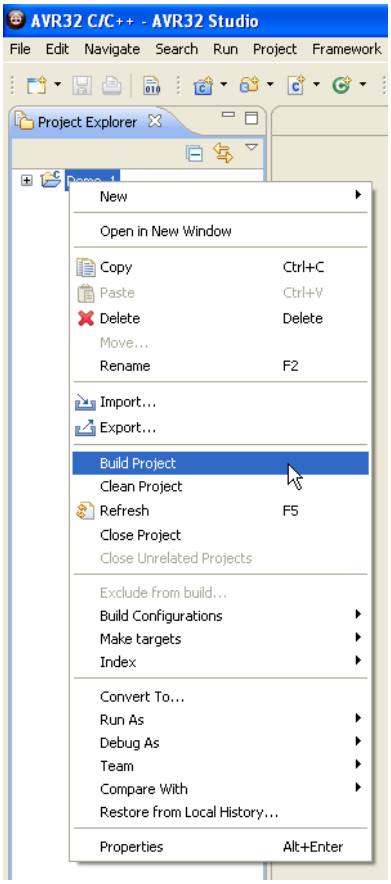
Select **EVK1100 – Components - DIP204 example**, then **Next**

Figure 6-14. New project name



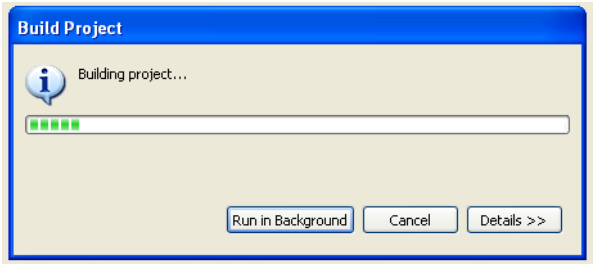
Enter a name for the project, and click **Finish**.

Figure 6-15. Build project



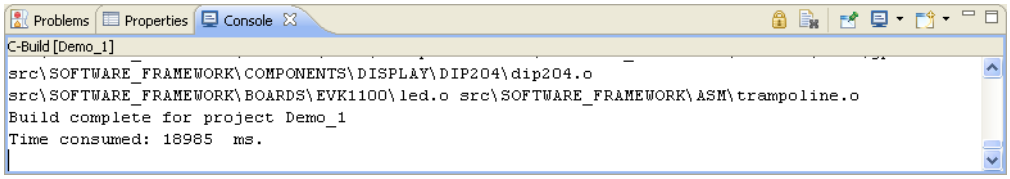
Right-click on the project in **Project Explorer**-view and select **Build Project** (or press CTRL+B).

Figure 6-16. Project build progress



Wait for the project build process to finish.

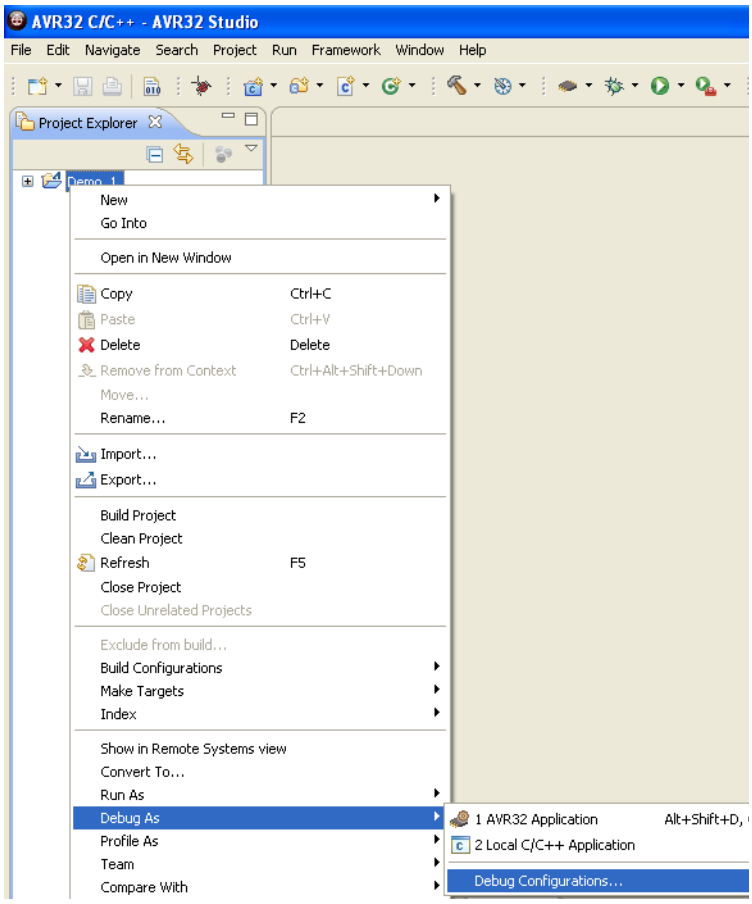
Figure 6-17. Console view



The console shows output from the compiler. Make sure that this ends with a “Build complete ...” message (Except for the “Time consumed” message). If something is not working, you will see error messages in this view.

6.4 Configure AVR32 Studio for a debug session using trace

Figure 6-18. Open Debug Dialog



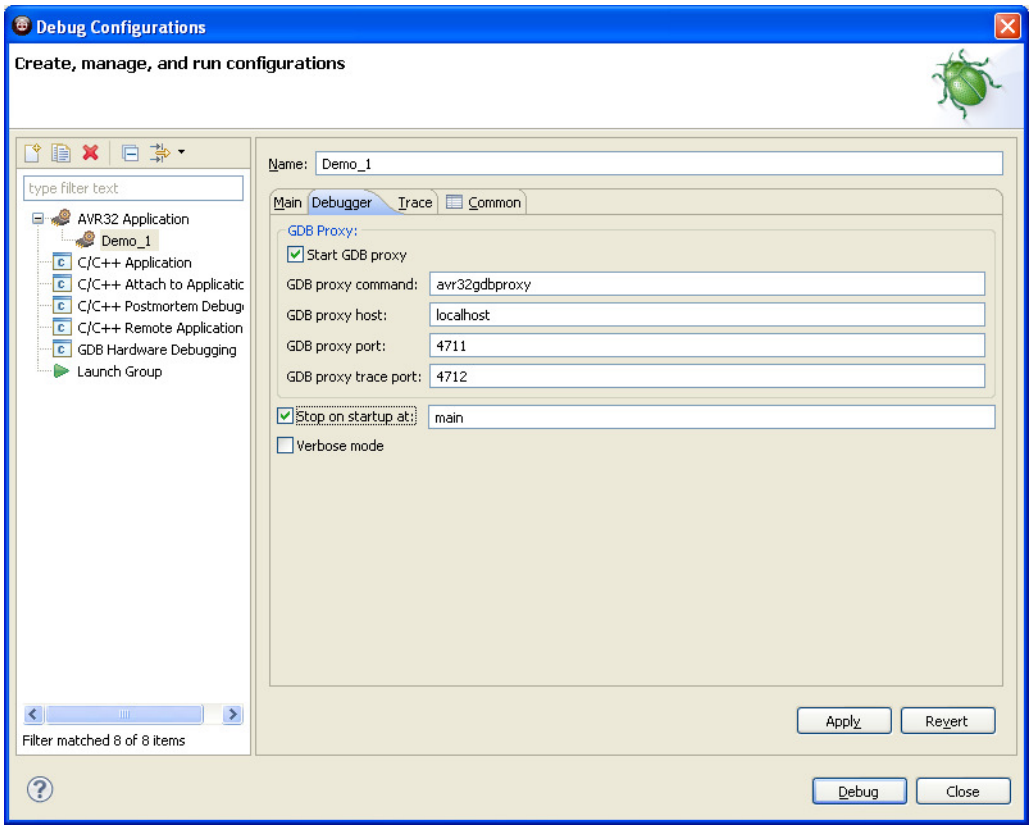
When the build process is finished, right-click on the project in the *Project Explorer* view and select *Debug As>Debug Configurations.*

6.4.1 Create a new debug launch configuration

In the *Debug Configurations* view, select **AVR32 Application** and right click and select **New**. A new launch configuration will be created and default values will be filled into all applicable fields.

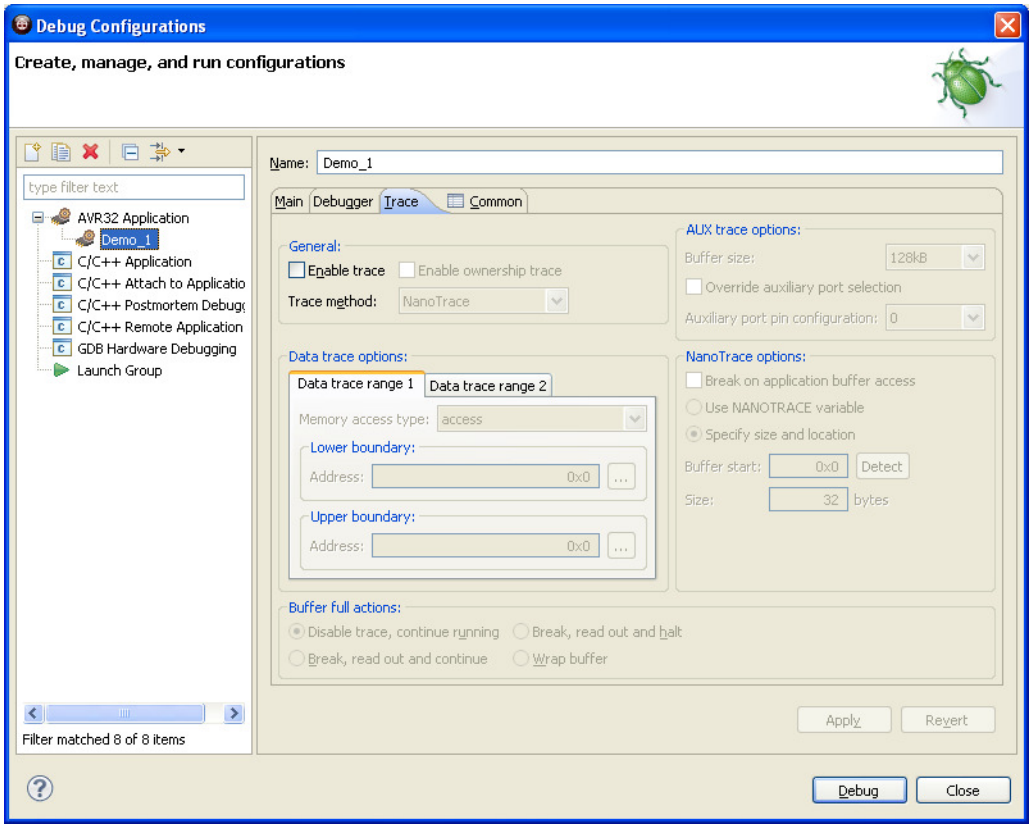
Select the *Debugger* tab and tick the **Stop on startup at: main** option.

Figure 6-19. **Debugger** tab



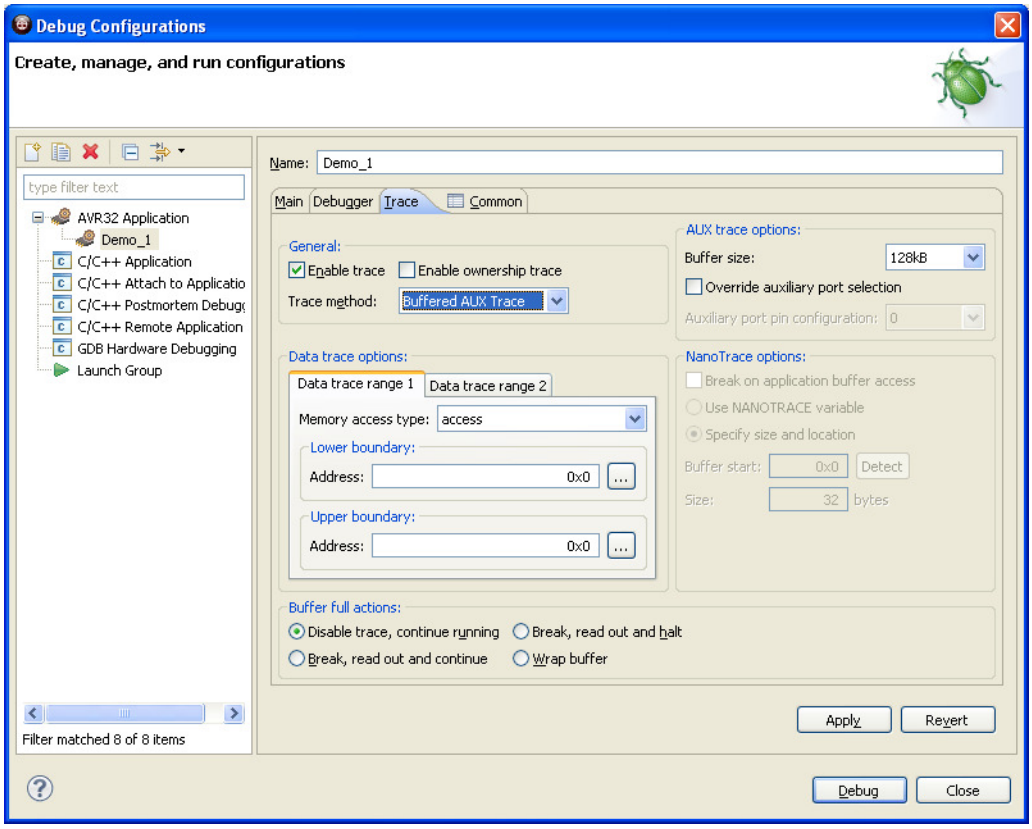
6.4.2 Configure the target trace module for program trace

Figure 6-20. Debug configurations, **Trace** tab



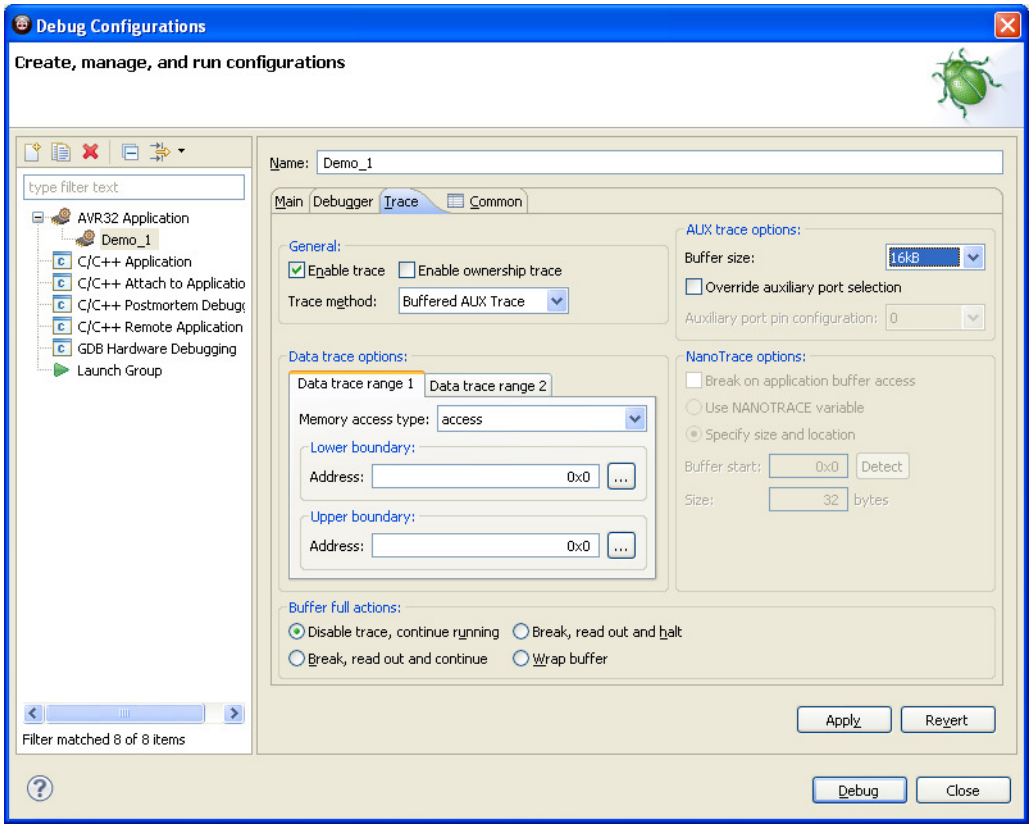
Select the **Trace** tab and click **Enable Trace**.

Figure 6-21. Preferred Trace method



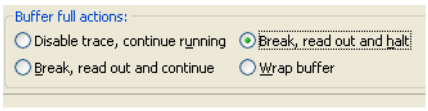
Select the preferred trace method. In this case we want **Buffered AUX Trace**.

Figure 6-22. Trace buffer size



Select Buffer Size. We select **16kB** for a quick test.

Figure 6-23. Buffer full action

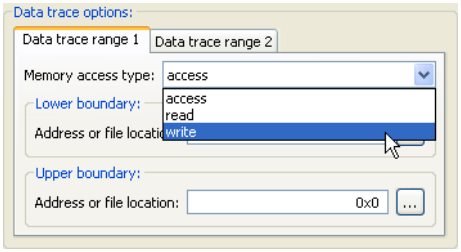


Selected the preferred action when buffer is full. In this case we choose **Break, read out and halt**.

6.4.3 Configure the target trace module for data trace

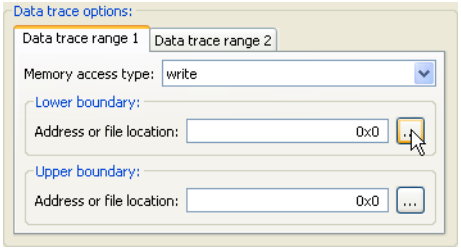
We would like to keep an eye on one of our variables. To do this, we configure a data trace range. In our case, we want a trace message each time the program writes to a variable called display.

Figure 6-24. Memory access type



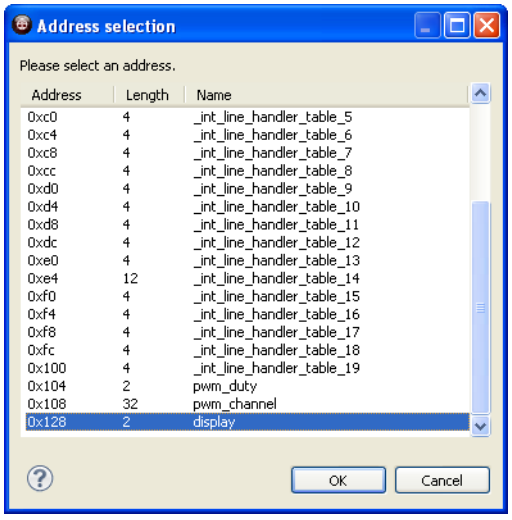
Set Memory access type to **write**.

Figure 6-25. Data trace boundaries



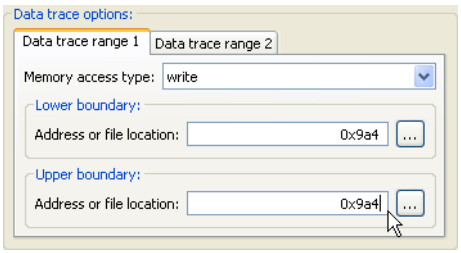
Select memory location for lower and upper boundaries.

Figure 6-26. Variable address selection dialogue



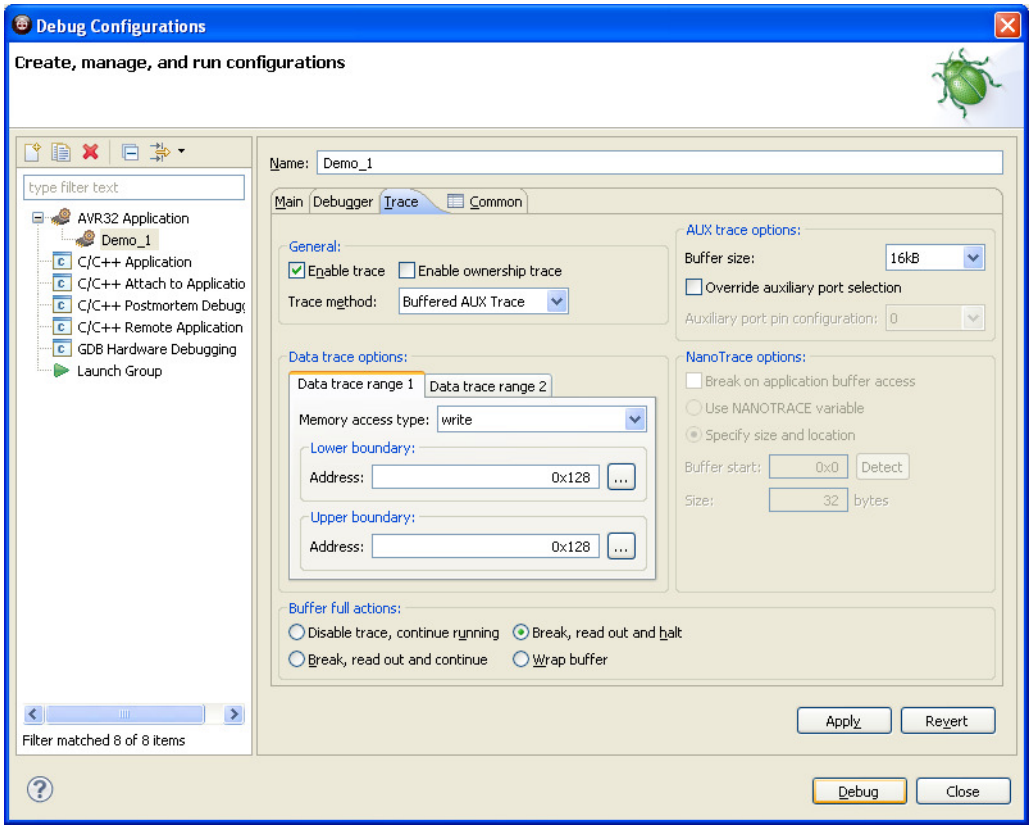
Select the start and stop addresses for the data range. Use the Address selection dialogue, or type the addresses.

Figure 6-27. Configured data trace range



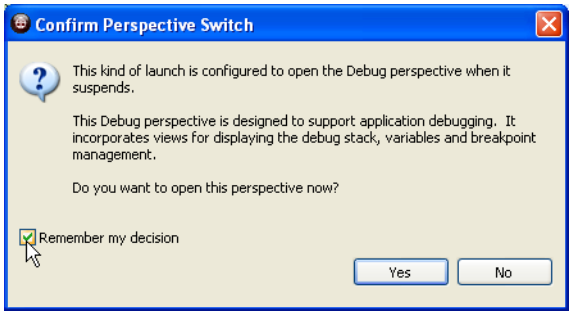
6.5 Start a debug session and configure the debugger for trace

Figure 6-28. Starting a debug session



Click the **Debug** button. Now the program will be loaded into the target, and run until main().

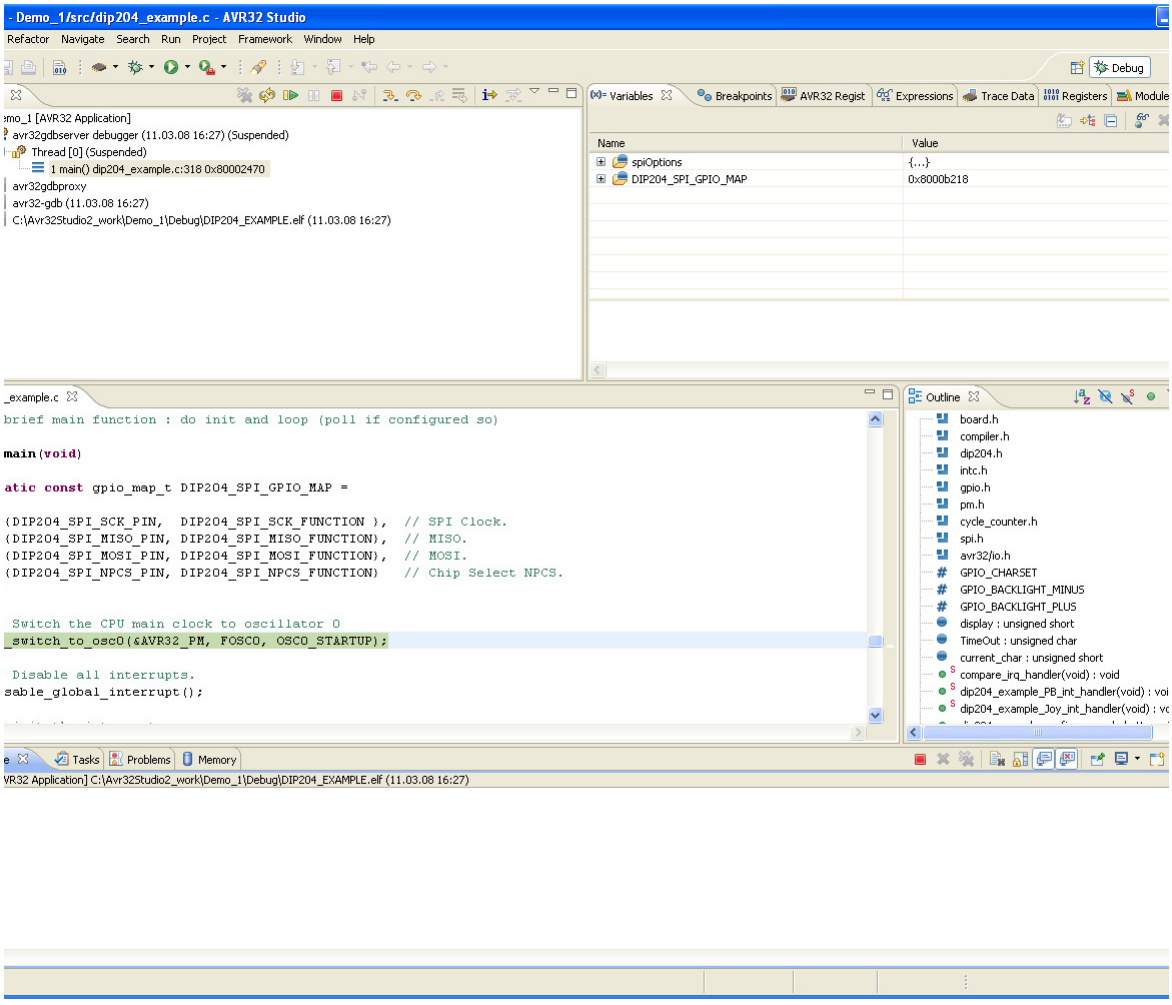
Figure 6-29. Switching perspective



When the debug session starts, AVR32 Studio 2.5 will change to the *Debug* perspective (desktop layout designed for use during debug sessions). You should click **Yes**. To avoid being asked every time you start a debug session, you should also click the **Remember my decision** box before answering **Yes**.

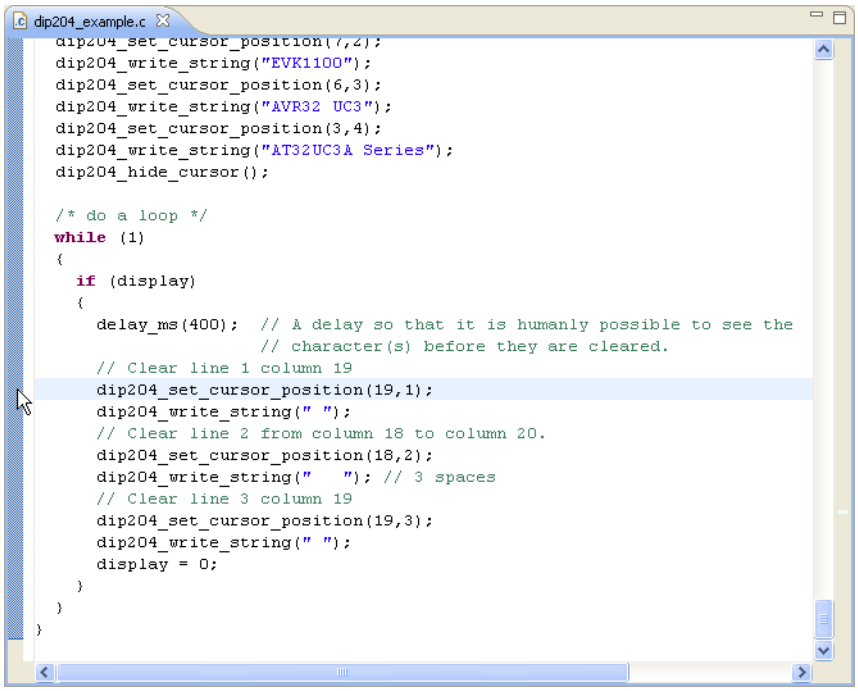
Wait until the target has stopped at the first instruction in the main() routine.

Figure 6-30. Program halted at main()



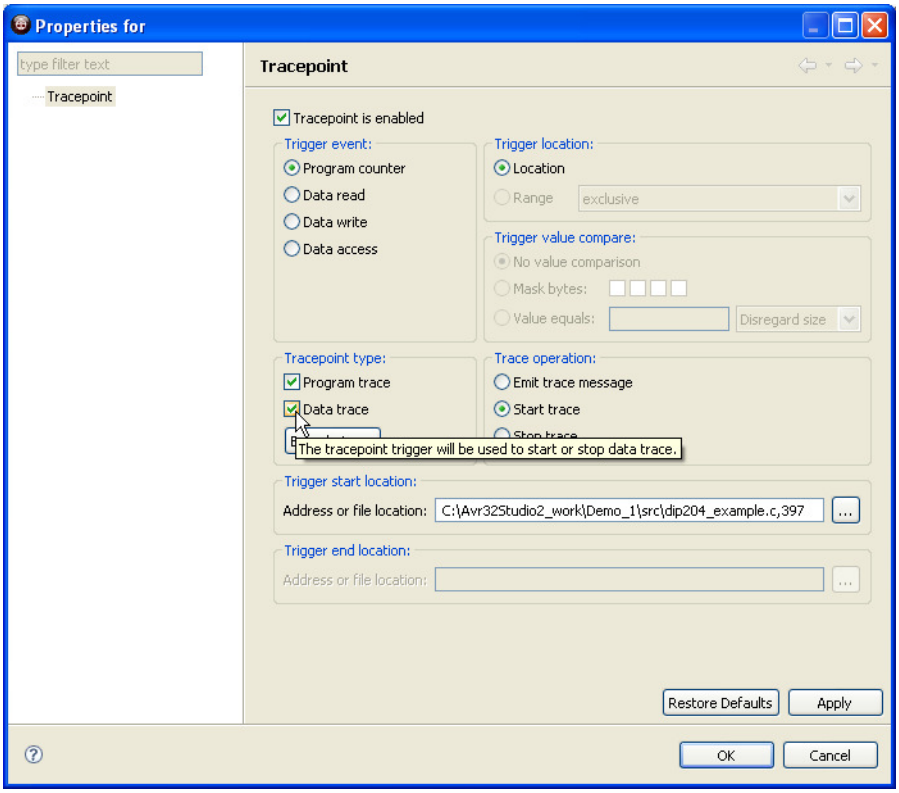
6.5.1 Add start and stop trace-points

Figure 6-31. Source code editor



Scroll down to and select line 356 in the file DIP204_Example.c and then select *Run>Toggle Trace Point*.

Figure 6-32. Tracepoint (Start)



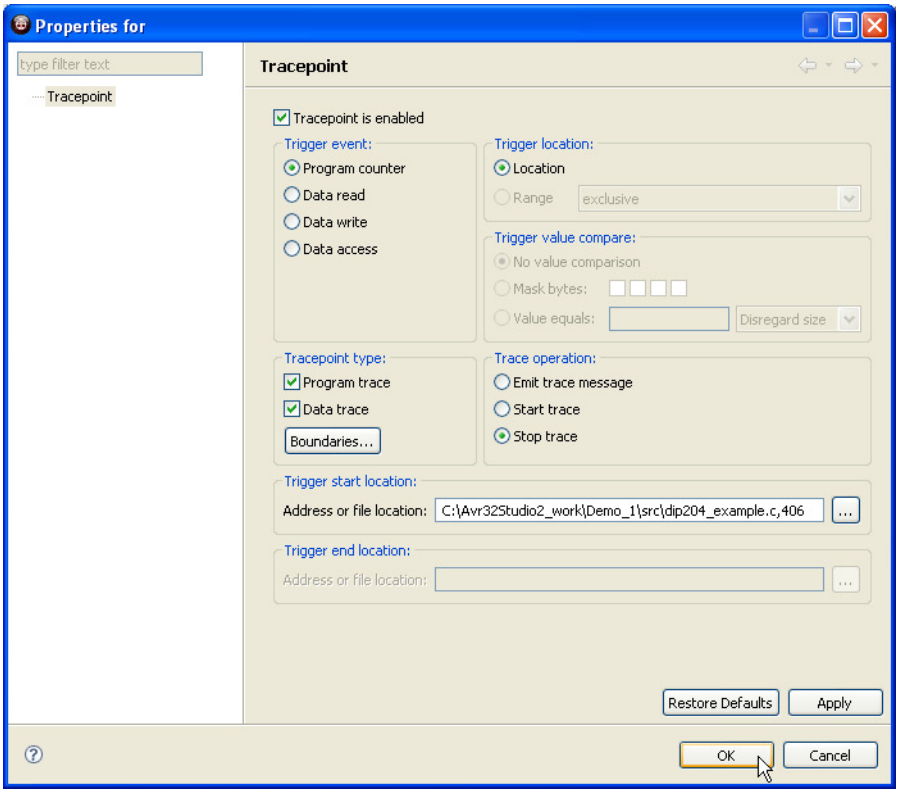
Set Tracepoint Configuration values:

- Set *Trigger Event* to *Program Counter*
- Set *Trace Operation* to *Start Trace*
- Set *Tracepoint type* to both *Program trace* and *Data trace*
- Click **OK**

This will create a tracepoint that starts both program and data trace when the program counter hits this code line.

Scroll down to and select line 364 in the file DIP204_Example.c and then select *Run>Toggle Tracepoint*.

Figure 6-33. Tracepoint (Stop)

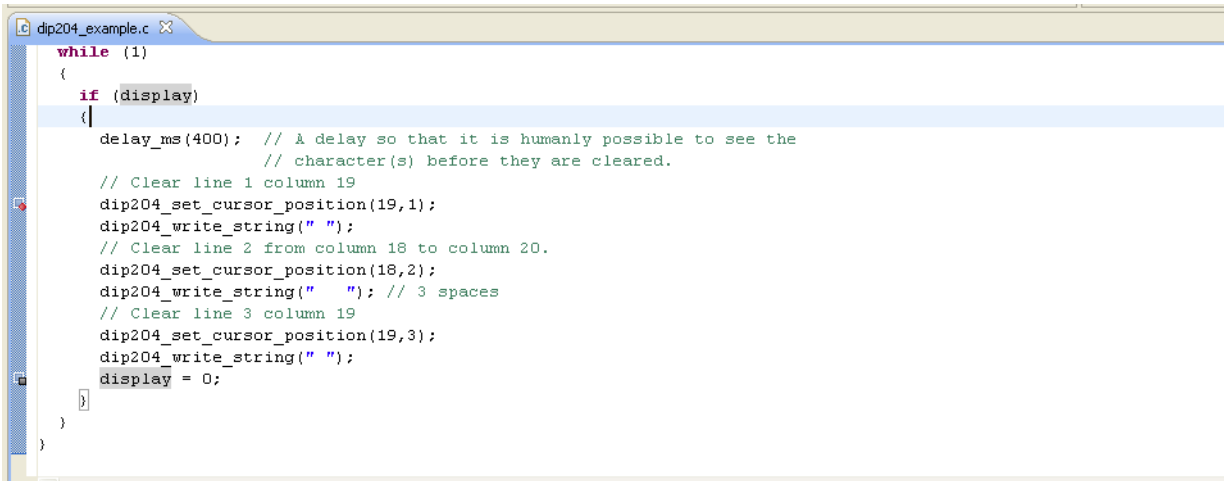


Set Tracepoint Configuration values:

- Set *Trigger Event* to *Program Counter*
- Set *Trace Operation* to *Stop Trace*
- Set *Tracepoint type* to both *Program trace* and *Data trace*
- Click **OK**

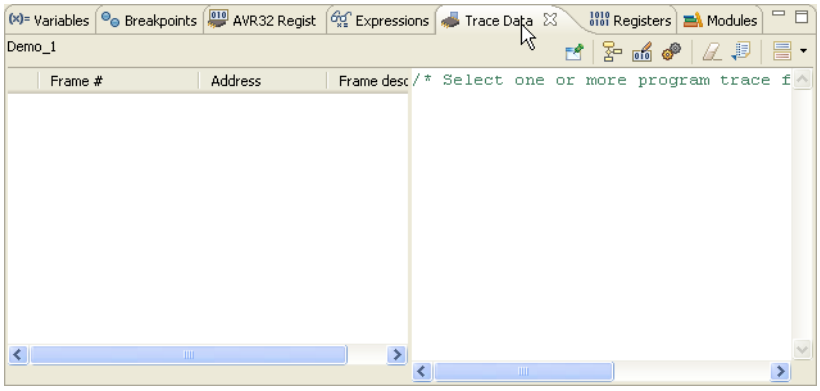
This will create a tracepoint that stops both program and data trace when the program counter hits this code line.

Figure 6-34. Source editor with tracepoint indicators



The source editor now has two tracepoint indicators next to the respective code lines.

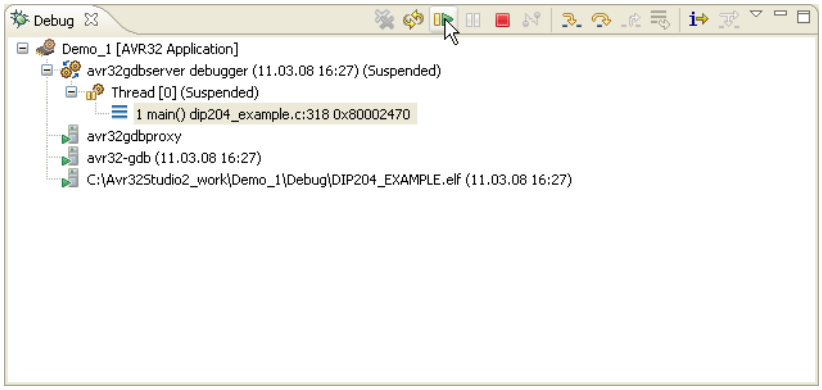
Figure 6-35. Trace data view (empty)



Click on the **Trace Data** tab to bring the trace data view to the front.

6.6 Start the trace debug session

Figure 6-36. Resume debug session



Make sure that the main() process is still selected in the *Debug* view before pressing the **Resume** button.

Figure 6-37. LCD Display showing original message

The display should look like this.

Push the joystick button on the EVK1100 evaluation board a few times, until the trace buffer is full and the target stops (6-7 button operations should be enough).

Figure 6-38. Target stopped because trace buffer full

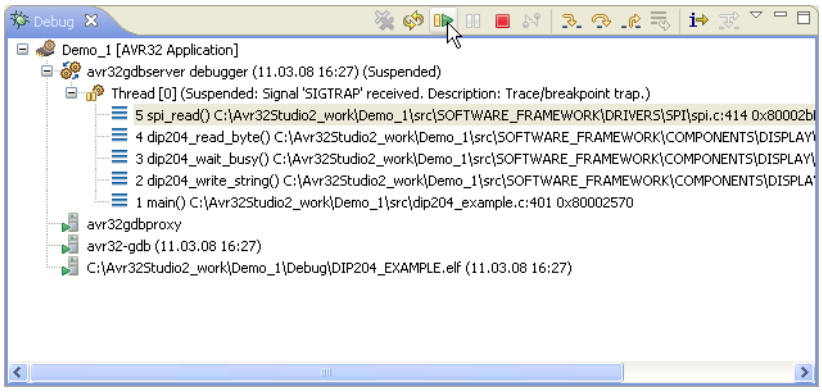
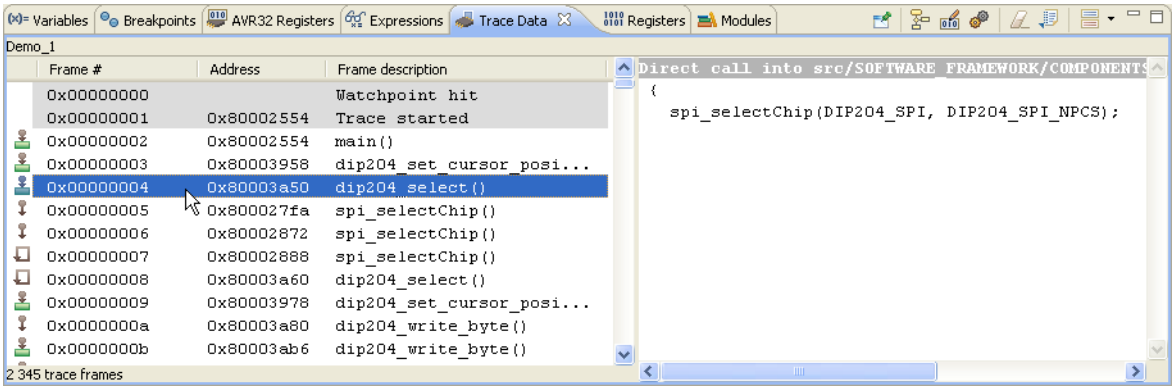
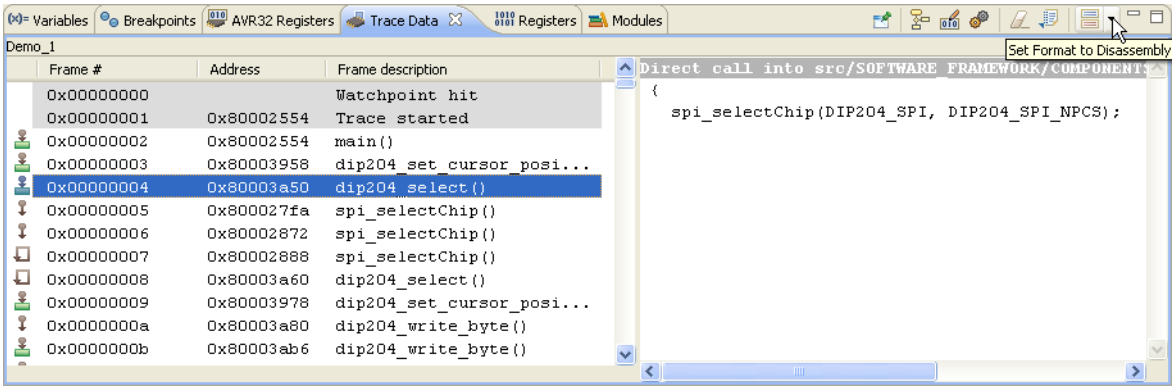


Figure 6-39. Trace data view (not empty)



Have a look at the trace data collected by clicking on a trace frame.

Figure 6-40. Changing trace view format



Change the format of the code view by opening the trace format menu (click the small arrow).

Figure 6-41. Set trace view format to *Mixed source and Disassembly*

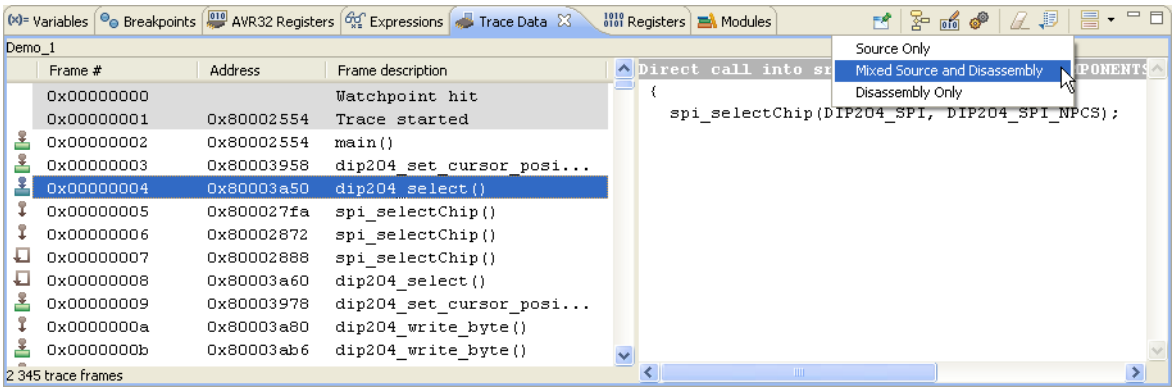
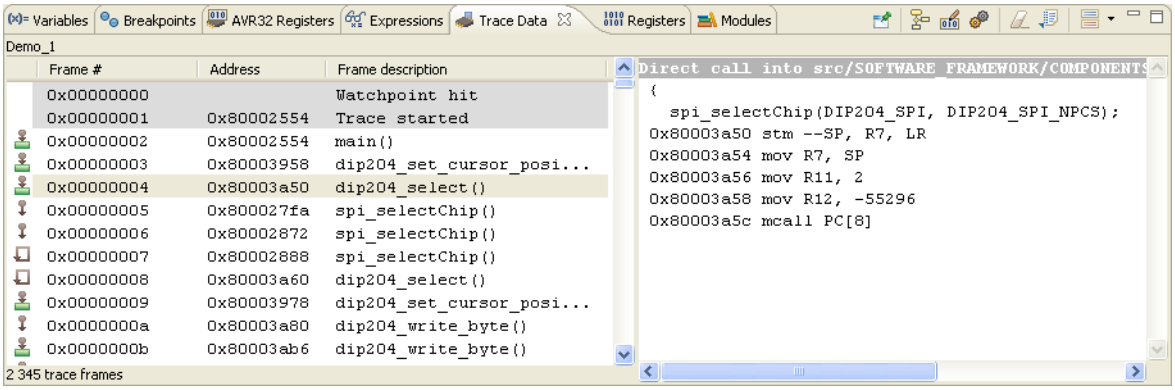
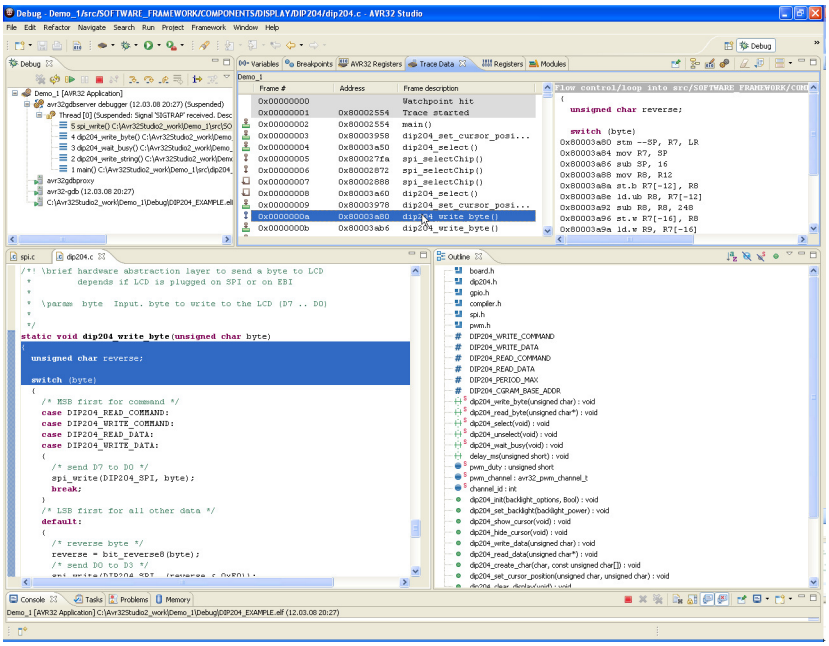


Figure 6-42. Viewing Mixed source and disassembly trace data



Double-click on a trace frame to highlight source code in the source editor.

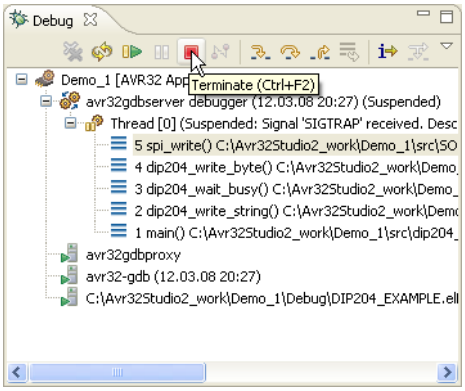
Figure 6-43. Trace frame highlighting source code in the editor



6.7 Modify the code and restart the debug session

If we want to make changes to our code, we must stop the debug session, edit, rebuild and start the debug session again.

Figure 6-44. Terminating the debug session



Edit the source code. This example changes the cursor position in line 342 from (6,3) to (1,3), then the text in line 343.

Figure 6-45. Editing source code line 343

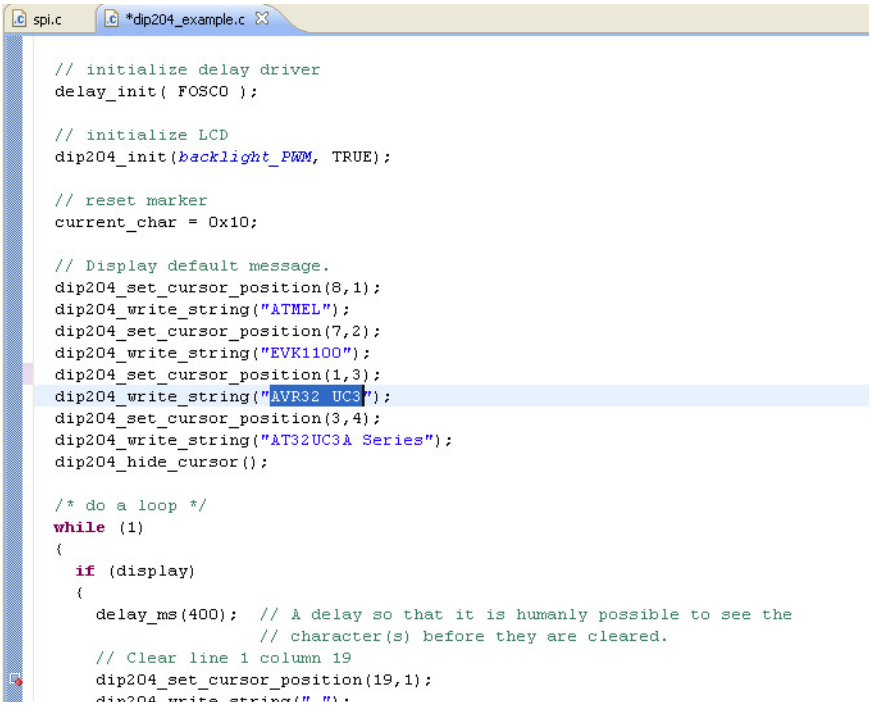
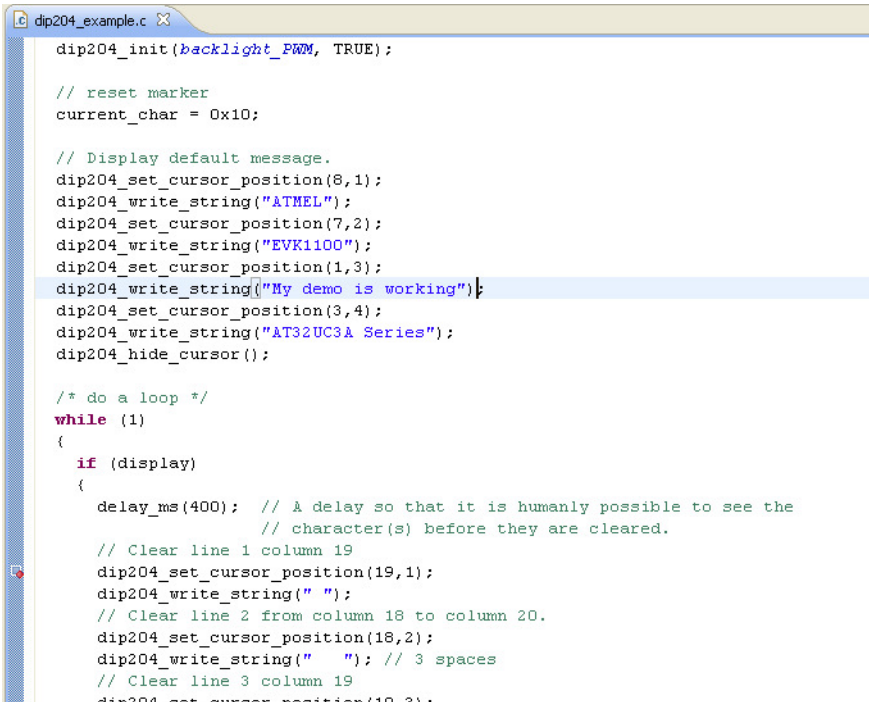


Figure 6-46. Source code edit finished



Start a new debug session. AVR32 Studio uses the previous Launch Configuration if you just press the Debug button.

Figure 6-47. The Debug button

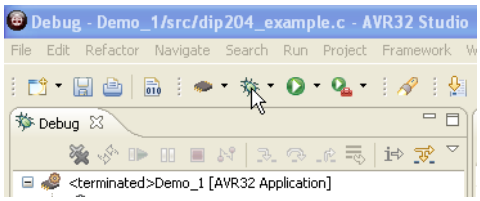
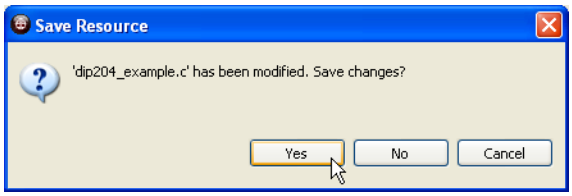
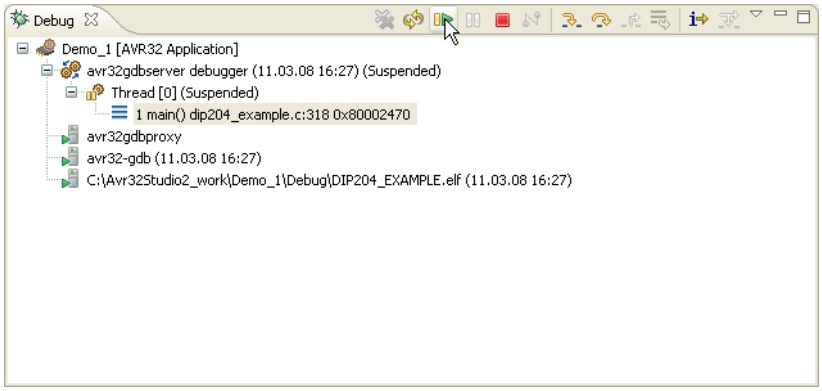


Figure 6-48. Save edited source code dialog



Confirm saving the edited source code file. AVR32 Studio2.5 will now rebuild the project and program the target MCU FLASH. The code will run from start to main() and halt.

Figure 6-49. Resume button



Click “Resume” to start the application.

Figure 6-50. LCD Display showing edited message

The LCD display should now contain the edited message.

Congratulations! You have now created your first AVR32 application and collected real time trace data from the target MCU running your program using the AVR ONE!

Section 7

Firmware Upgrade

7.1 Firmware upgrade overview

The tools (adapters) used to provide the physical connection between PC and target MCU contains firmware. This firmware needs to be compatible with the gnu toolchain and AVR32 Studio installed on the PC.

When AVR32 Studio is started, or when a new adapter is detected, AVR32 Studio will perform a firmware version check to determine if the adapter firmware needs to be upgraded.

If AVR32 Studio contains a newer firmware than present in the adapter, the adapter will be upgraded.

7.2 Firmware version test and upgrade

When AVR32 Studio is testing the firmware version of connected adapters, you can see a progress indicator in the status line.

Figure 7-1. Firmware version test

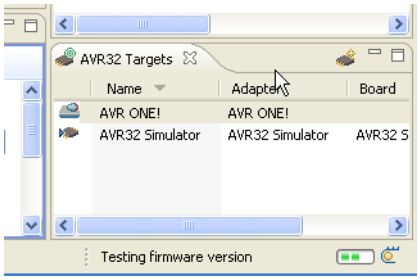
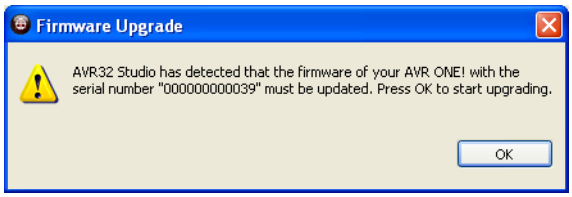


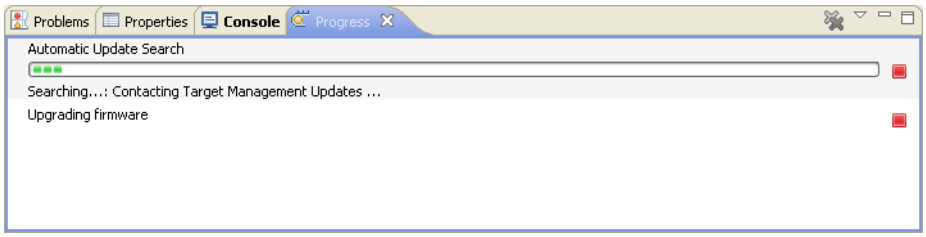
Figure 7-2. Firmware upgrade message



If the adapter firmware must be upgraded, you will be notified by a pop-up. Click **OK** to continue.

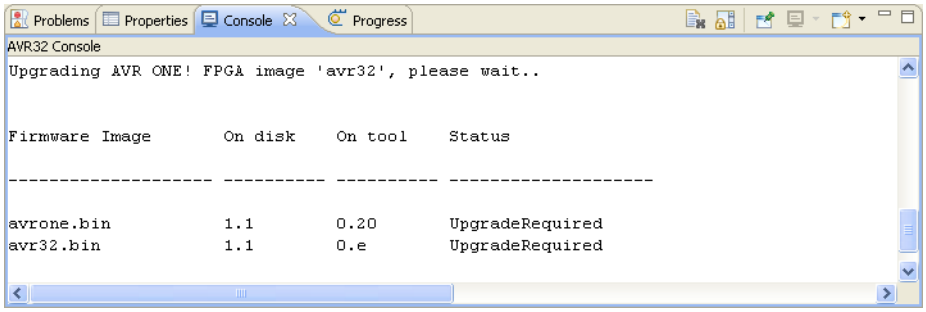
Firmware upgrade progress can be monitored by activating the *Progress* view.

Figure 7-3. Firmware upgrade progress



A firmware upgrade report can be found in the *Console* view.

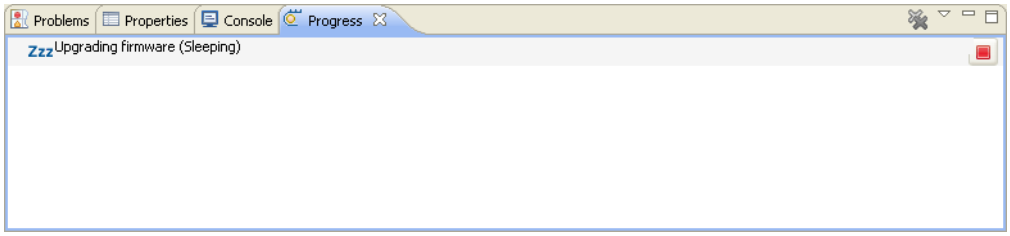
Figure 7-4. Firmware upgrade report



7.3 Adapter in use

The firmware version test is a process that is running in the background. This may cause a situation where the adapter is busy (debug session active) when AVR32 Studio determines that the firmware should be upgraded. In this case, the firmware upgrade process will wait until the adapter is not busy anymore (debug session terminated).

Figure 7-5. Firmware upgrade process waiting for adapter



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com/avrone

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Processor Accessories ***category:***

Click to view products by Microchip ***manufacturer:***

Other Similar products are found below :

2447 451651 SPB204-AL-1 90001265-88 32316 MIKROE-2147 20-101-0431 28148 T050000 28961 28960 Basic US PI3 Kit 82634DSARPLTVIK 1557 20-101-0502 28109 EP-CHUPCNETPLUS 1019 4466 AD-FMC-SDCARD ATATMEL-ICE-CABLE ATATMEL-ICE-PCBA ATAVR-MICTOR38 Basic INTL PI3 Kit 20-101-0495 BK0006 BK0007 LG624 DAISY CHAIN-1 MIKROE-2094 MIKROE-2148 MIKROE-2149 130-35000 28106 28152 30055 30078 32333 555-28188 572-28132 CWH-CTP-STC-YE 910-28015B ATABOT 700-00056 FXX-3006-JES 28114 28985 868 1744 ARX-DSP B000003