# AVR ONE!

# Quick-start Guide

EVK1101 + Windows®

***Section 6***

# 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

- 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 EVK1101 using the 10pin JTAG connector.
- Connect the EVK1101 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 *Board*-tab. Set Board to **EVK1101**, MCU to **UC3B0256** or **UC3B256ES**, depending on what MCU is mounted on your EVK1101.
- Right-click on the AVR ONE! in the *AVR32 Target* view and select **Chip Erase**. This operation is only needed one time (when the EVK1101 is new).

## 2.2 Create a demonstration project

- *Select File>New>Example*.
- Select *EVK1101>Components>Accelerometer 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*-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 **Nano Trace**.
- Select the preferred action when buffer is full. In this case we choose **Break, read out and halt**.
- Deselect the option **Break on application buffer access**
- Set Buffer Size. Select **Specify size and location**, then click **Detect**.

## 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

# Software Installation

## 3.1 Download the software

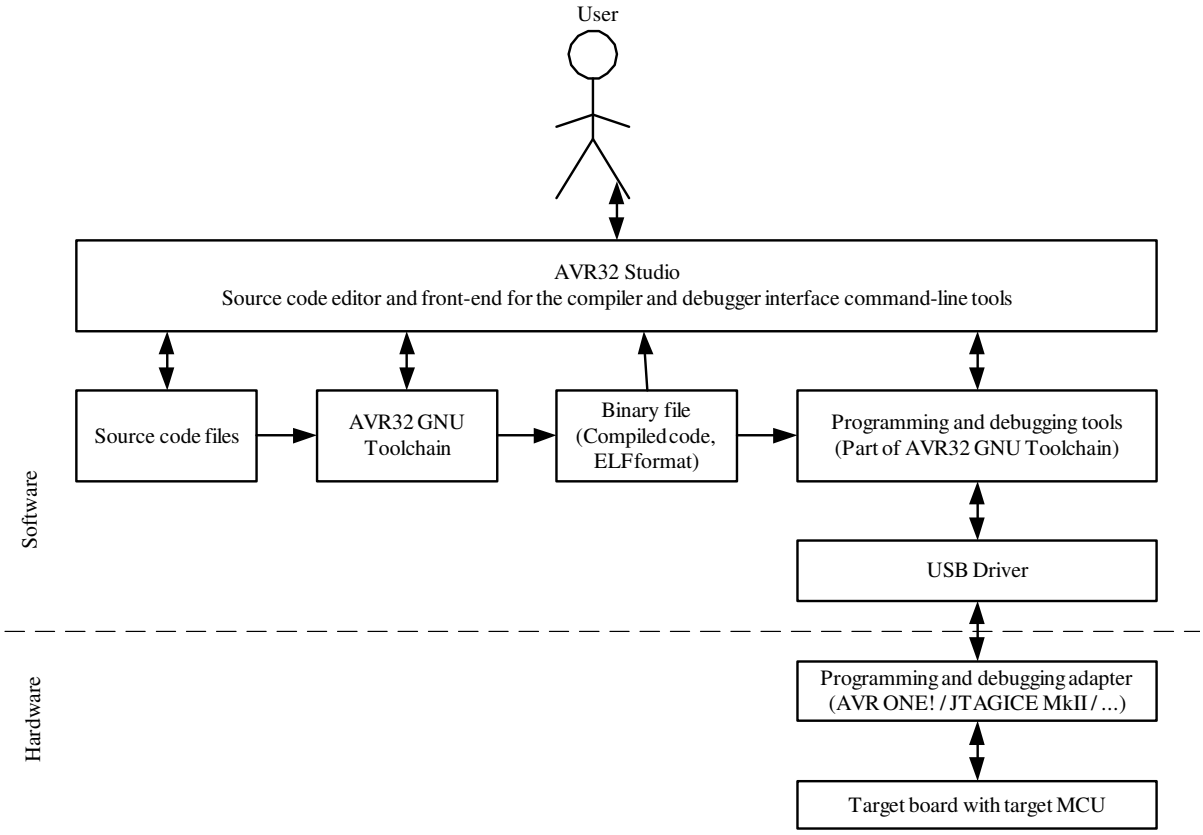
To use the AVR ONE!, you must download and install two software packages:

- 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 3-1.*** Tools structure



## 3.2 Download the two installation files to your disk.

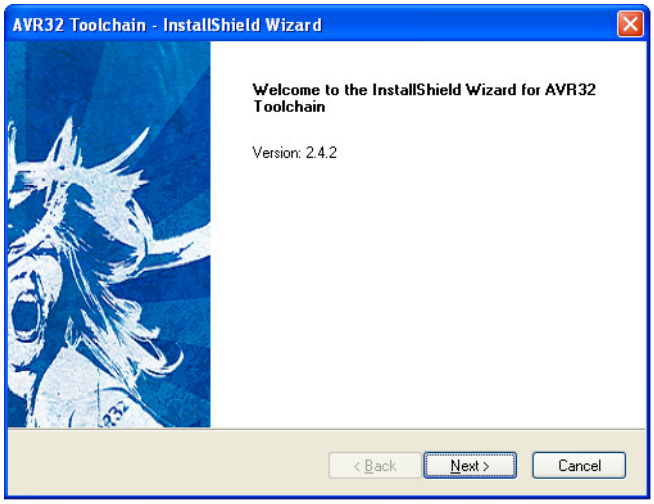
The installation files can be found at this location: **www.atmel.com/avrone**

## 3.3 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 3-2.*** AVR32 GNU Toolchain installation welcome

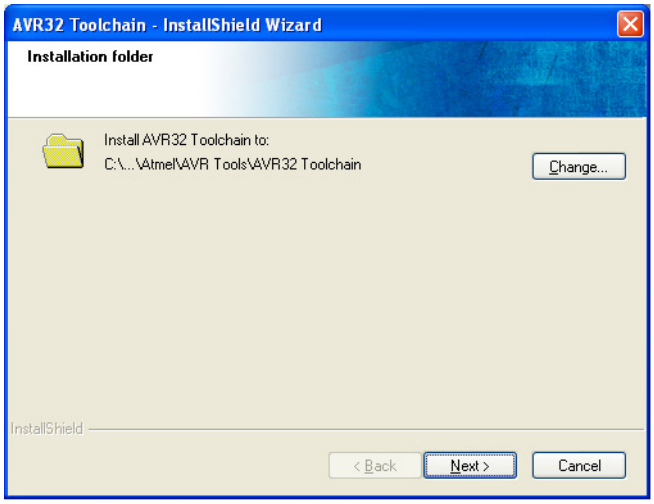



Click **Next**.

***Figure 3-3.*** AVR32 GNU Toolchain License Agreement form

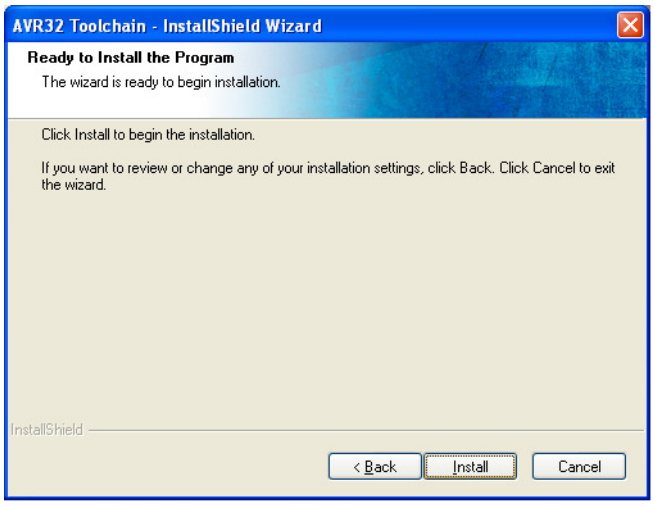

Select **I accept the terms of the licence agreement**, then click **Next**.

***Figure 3-4.*** AVR32 GNU Toolchain installation folder select



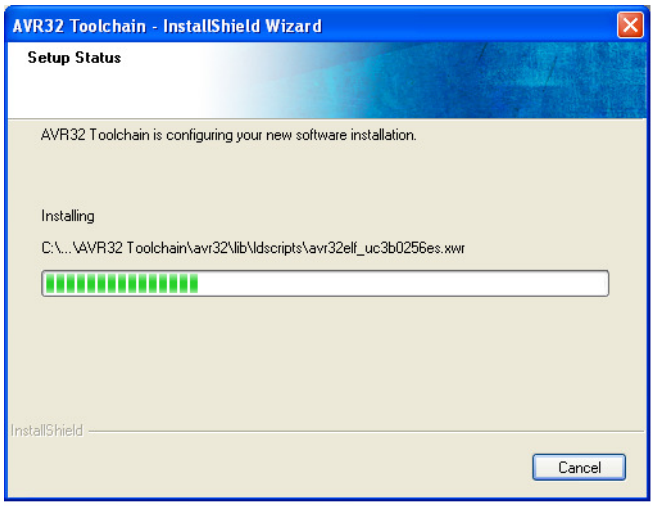

Check that the installation folder is correct and click **Next**.

***Figure 3-5.*** AVR32 GNU Toolchain installer configuration finished




Click **Install**.

***Figure 3-6.*** 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 3-7.*** USB Drivers installation start

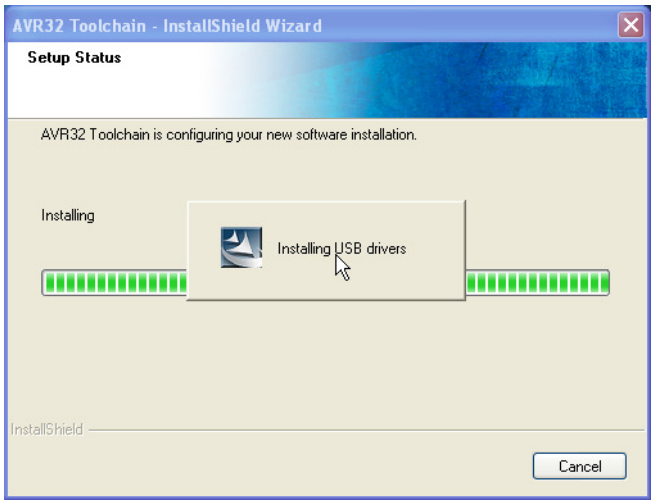
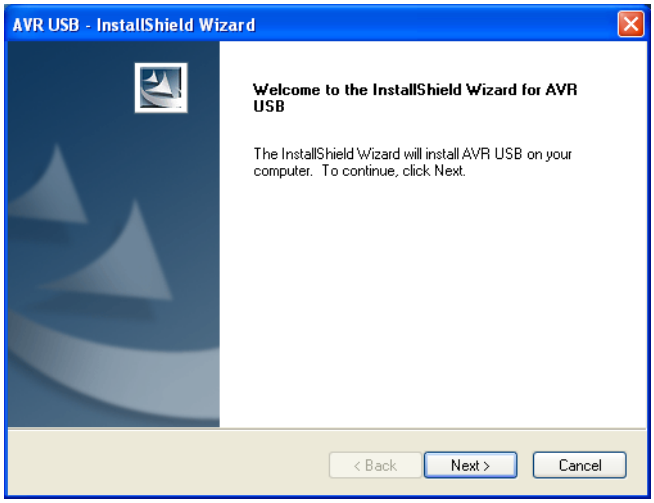
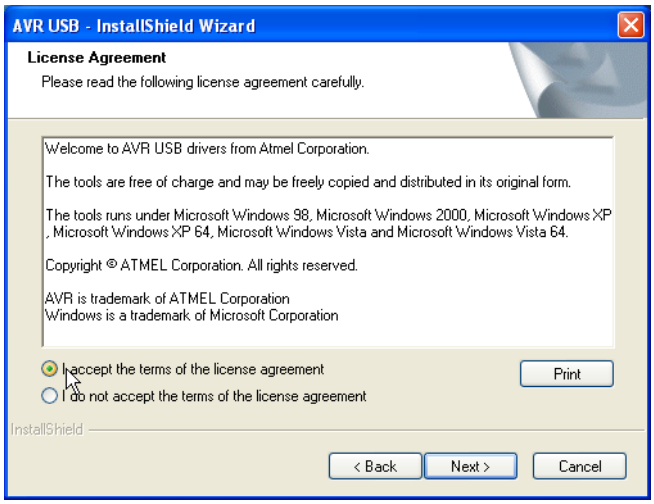



***Figure 3-8.*** USB Driver installer welcome



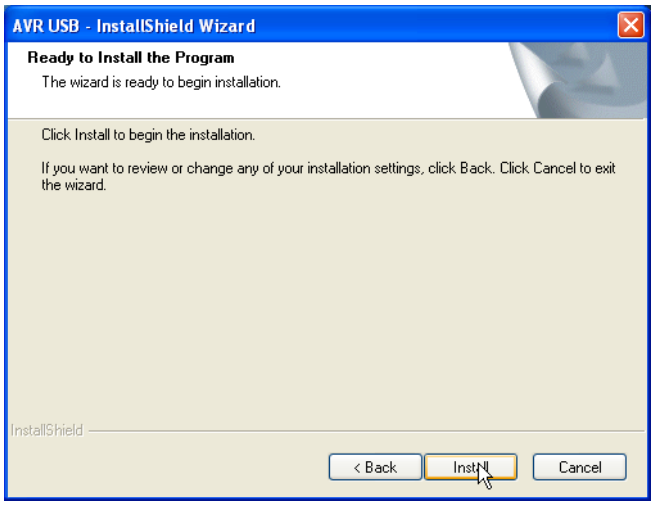

Click **Next**.

***Figure 3-9.*** USB Drivers licence agreement form




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

***Figure 3-10.*** USB drivers installer configuration finished




Click **Install**.

***Figure 3-11.*** USB Drivers installation progress indicator

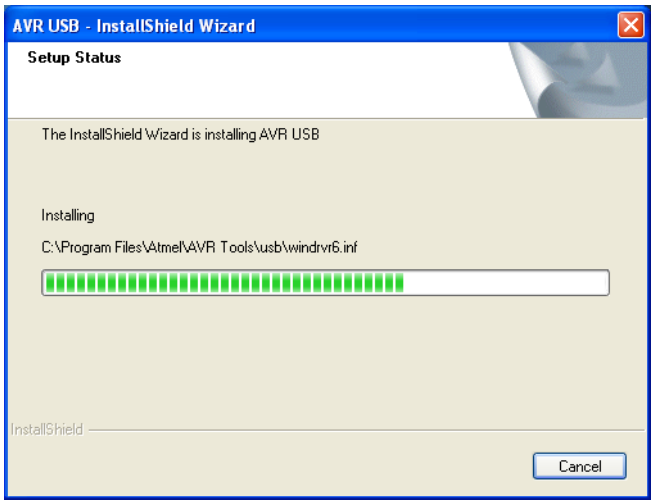
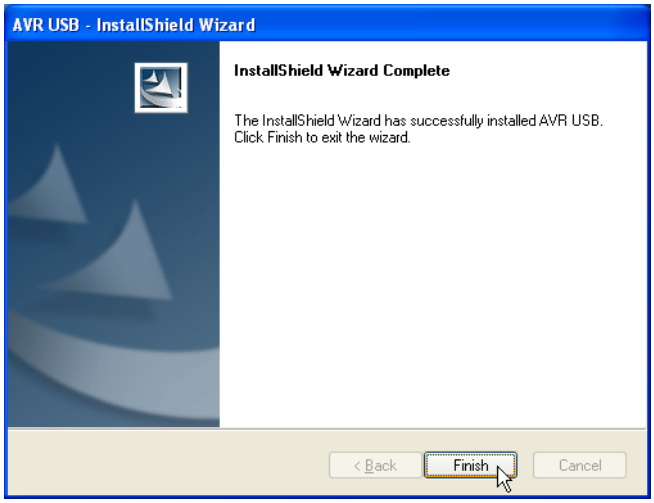
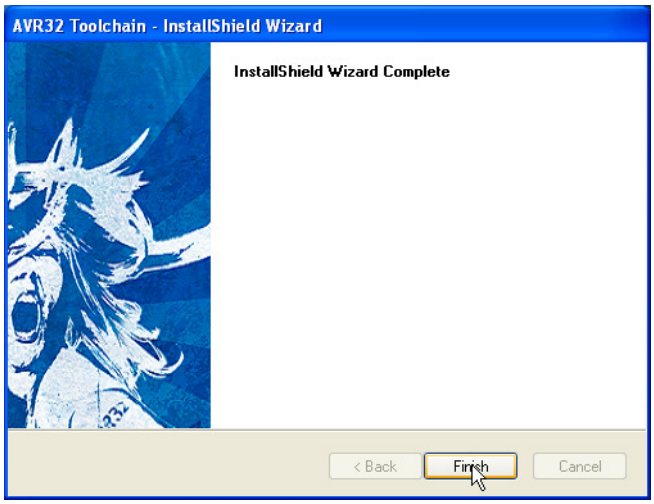



***Figure 3-12.*** USB Drivers installation complete




Click **Finish**.

***Figure 3-13.*** AVR32 GNU Toolchain installation complete

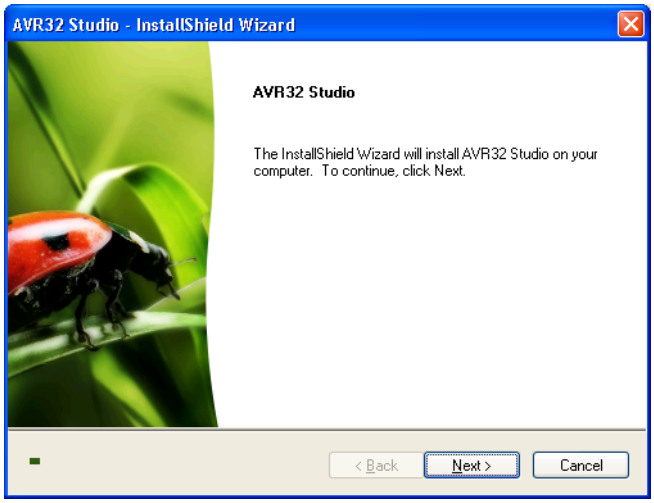



Click **Finish** to complete the AVR32 Toolchain installation process.

## 3.4 Install AVR32 Studio 2.5

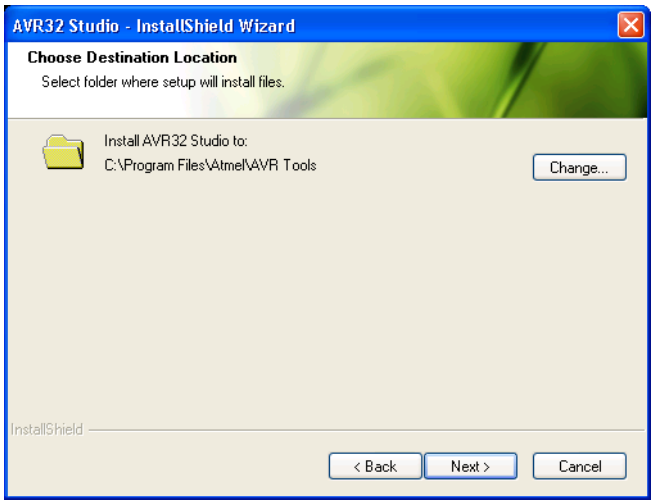
Double-click on the AVR32Studio-2.5.x.exe file to start the installation process.

***Figure 3-14.*** AVR32 Studio 2.5 installer welcome



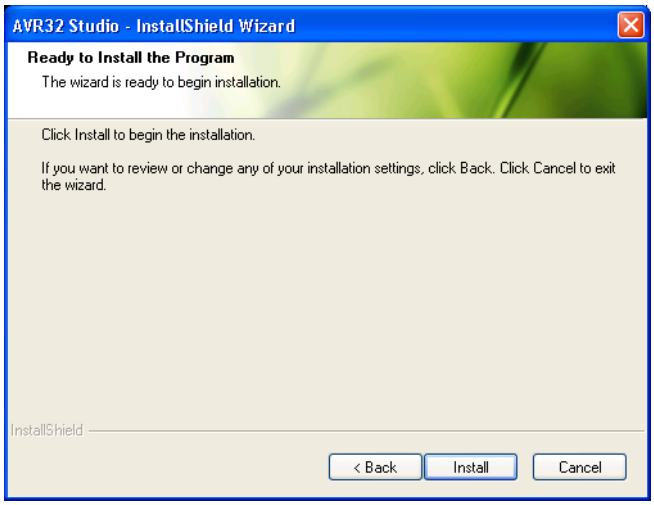

Click **Next**.

***Figure 3-15.*** AVR32 Studio installation folder select



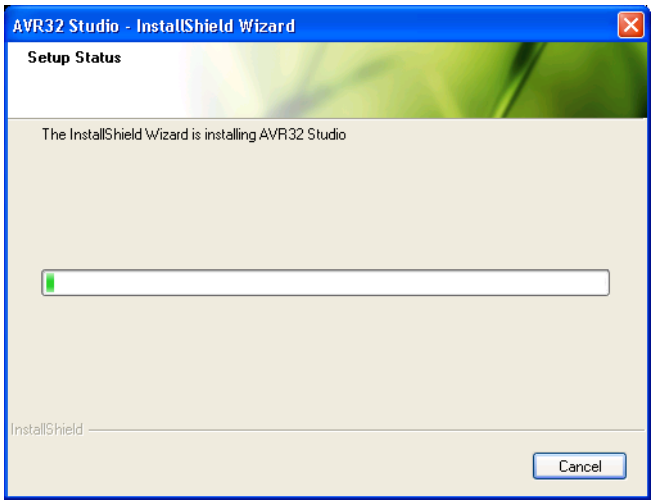

Check that the installation folder is correct and click **Next**.

***Figure 3-16.*** AVR32 Studio installer configuration finished




Click **Install** to start the installation.

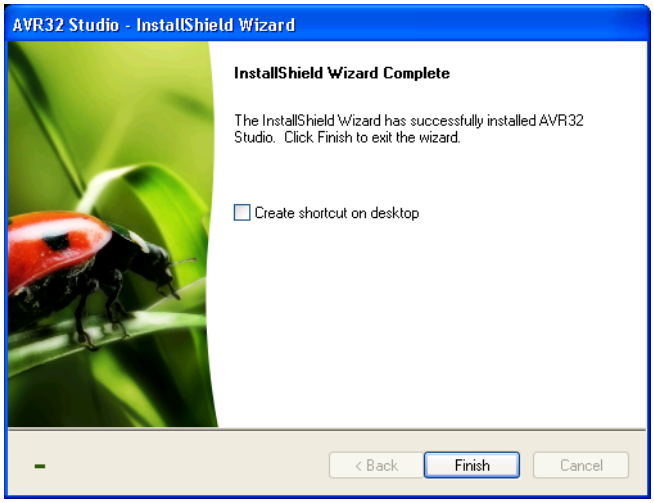
***Figure 3-17.*** AVR32 Studio installation progress indicator




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 3-18.*** AVR32 Studio installation process complete




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

## 3.5 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 3-19.*** AVR ONE! connected to power and USB

## 3.6 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 3-20.*** "New hardware" notification pop-up

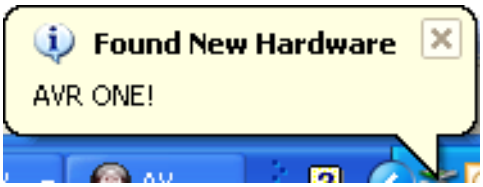
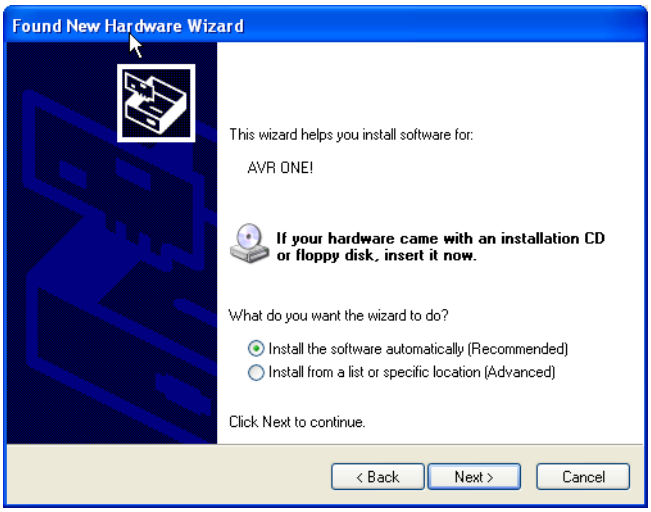



***Figure 3-21.*** AVR ONE! Hardware installation wizard

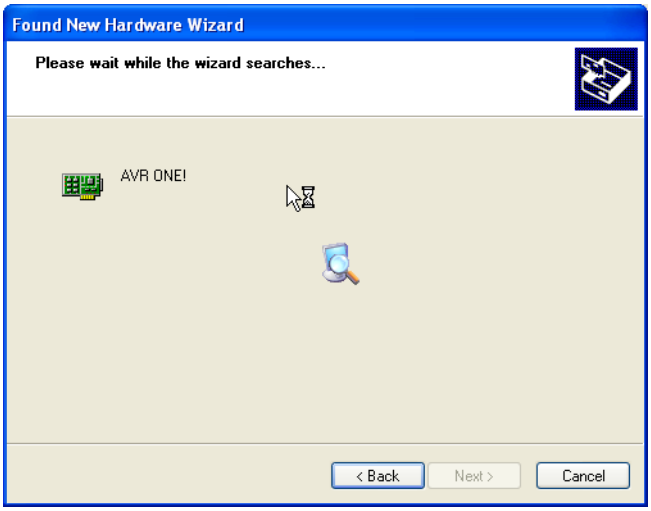

When the hardware installation wizard pops up, select **No, not this time** and click **Next**.

***Figure 3-22.*** Hardware installation wizard configuration



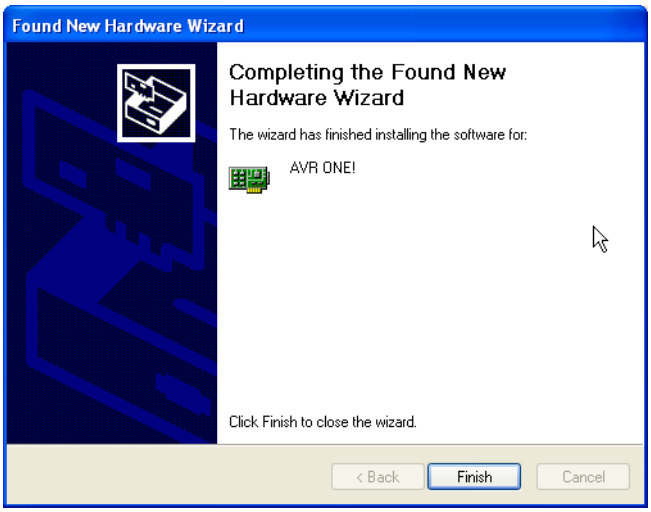

Select **Install the software automatically** and click **Next**.

***Figure 3-23.*** Hardware installation in progress




Wait for the installation process to complete.

***Figure 3-24.*** Hardware installation wizard complete



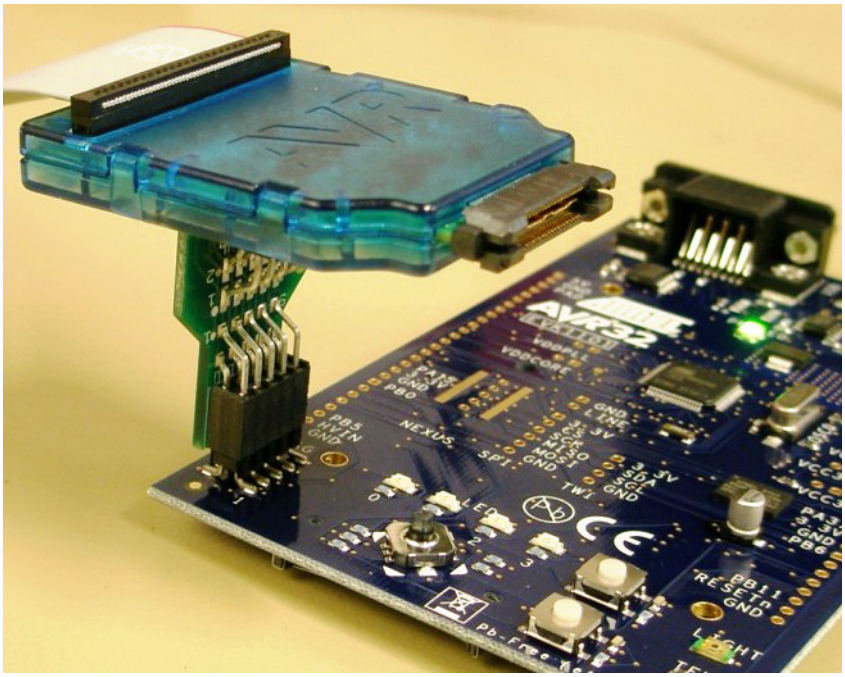

Click **Finish**.

# Section 4

## 4.1 Connect the AVR ONE! to the EVK1101

Connect the AVR ONE! debugger to the EVK1101 evaluation board using the 10 pin JTAG connector. To make it possible to use the joystick while the AVR ONE! is connected to the JTAG connector, the 100mil - 100mil JTAG stand-off adapted can be used.

***Figure 4-1.*** AVR ONE! connected to the EVK1101



## 4.2 Connect the EVK1101 to power and RS232

Connect the EVK1101 to power and turn it on. The easiest way to provide power is to use the supplied USB cable. Also connect the RS232 port to your PC using the supplied RS232 cable.

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

***Figure 4-2.*** Powering the EVK1101 using the USB cable


Note: If the EVK1101 contains the Control Panel Demo Application, you may be requested to install drivers for it. Just cancel this request (you do not need to install this driver).

# Section 5

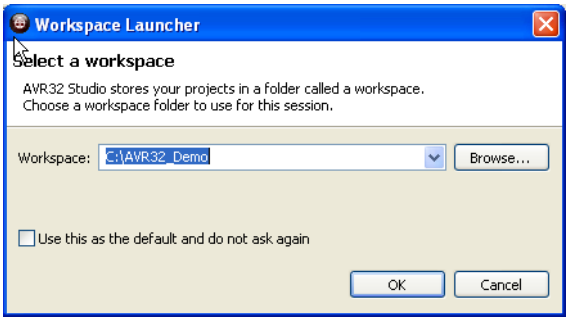
# Create demo application

## 5.1 Start AVR32 Studio

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

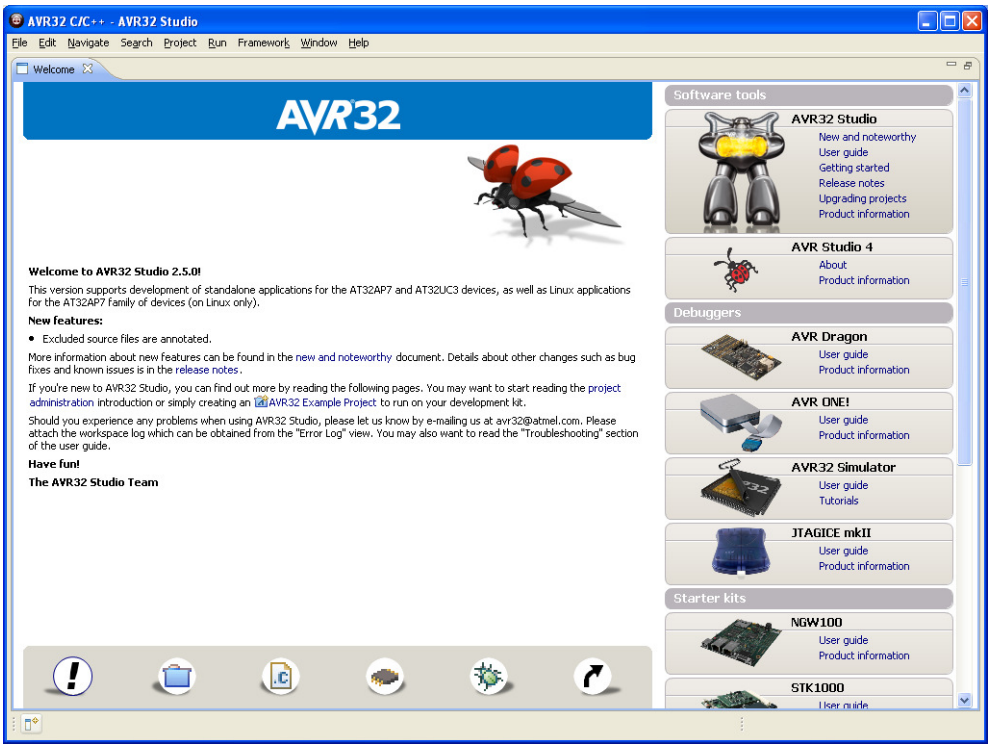
***Figure 5-1.*** AVR32 Studio splash screen


***Figure 5-2.*** 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 5-3.*** AVR32 Studio Welcome view




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

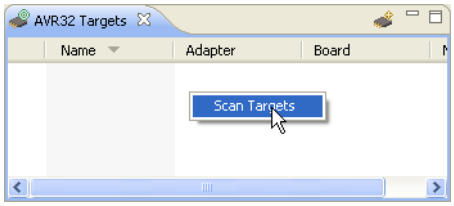
## 5.2 Configure adapter and target

Before you can use the AVR ONE! and the EVK1101, you have to tell AVR32 Studio what type of equipment is connected to your PC.

“Target” refers to the MCU on the EVK1101 evaluation board, and “Adapter” refers to the tool connecting the target to the PC (in this case, the AVR ONE!).

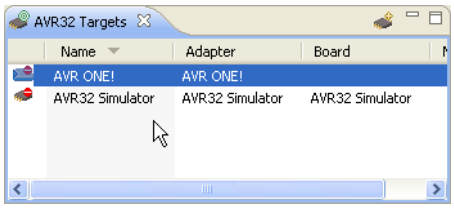
### 5.2.1 Add and configure the adapter (AVR ONE!)

***Figure 5-4.*** Scan Targets



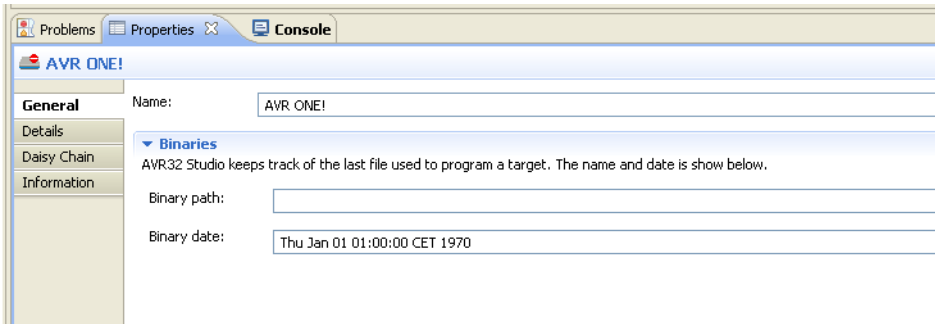

Right-click in the **AVR32 Target**-view and select **Scan Targets**.

***Figure 5-5.*** Available targets




Select the AVR ONE!

***Figure 5-6.*** Selecting the properties view

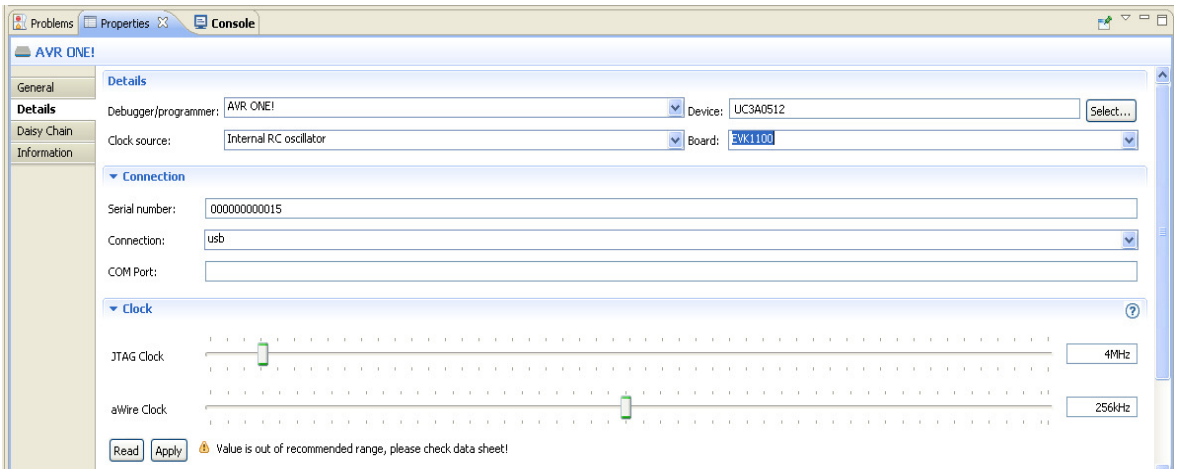



Click on the **Properties** tab.

You are now looking at the *Target* properties. 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.

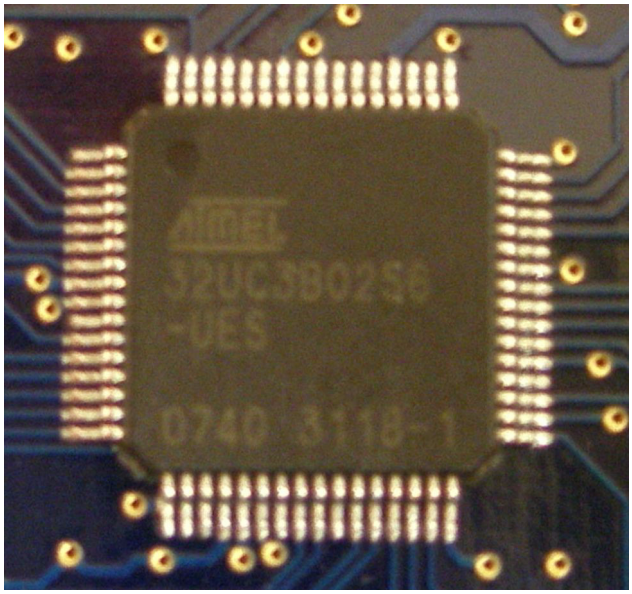
### 5.2.2 Configure target board and MCU

***Figure 5-7.*** Details configuration tab




Set **Device** to **UC3B0256** or **UC3B0256ES**, depending on what MCU is installed on your EVK1101.

***Figure 5-8.*** MCU Markings



To check which type of MCU is mounted on your EVK1101 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 **EVK1101**.

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

### 5.2.3 Target MCU Chip erase

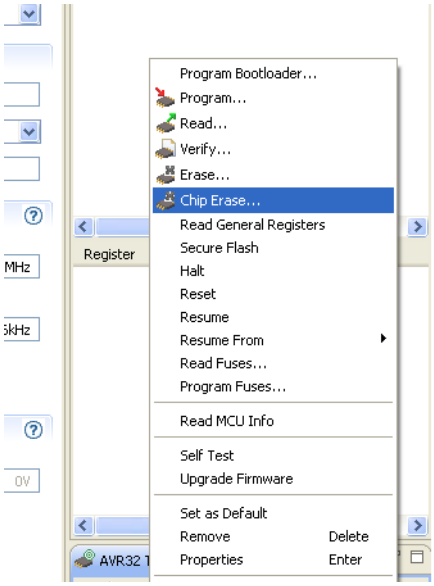
If the EVK1101 evaluation board is brand new, or if it still contains the original demo application (Control Panel Demo), the FLASH lock-bits need to be cleared. Right-click on the AVR ONE! In the *AVR32 Target* view and select **Chip Erase**.

WARNING! This process will erase the original demo application programmed at the factory. After this operation the EVK1101 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 EVK1101 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.

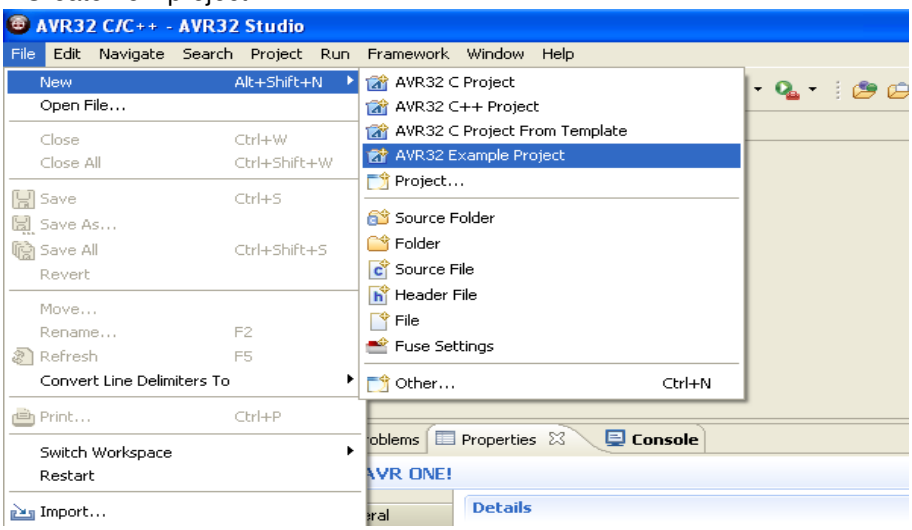
***Figure 5-9.*** Chip erase operation




Right click on the target (AVR ONE!), and select **Chip Erase**.

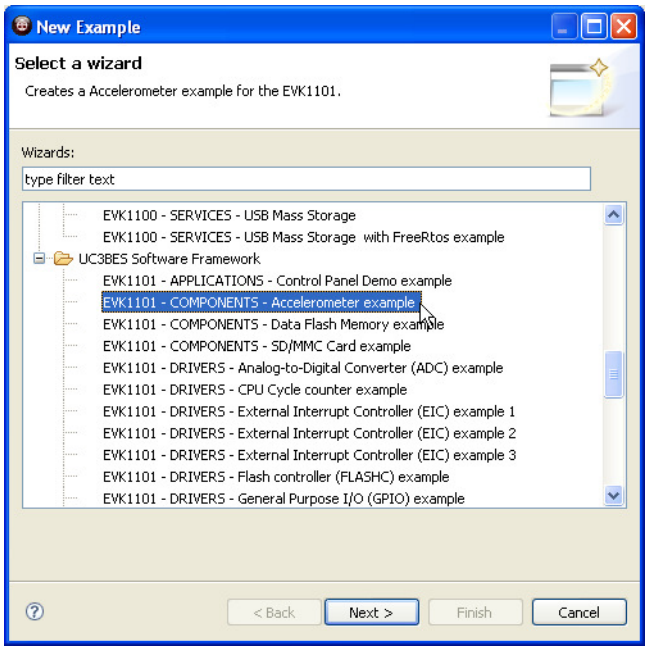
## 5.3 Create a demonstration project

***Figure 5-10.*** Create new project



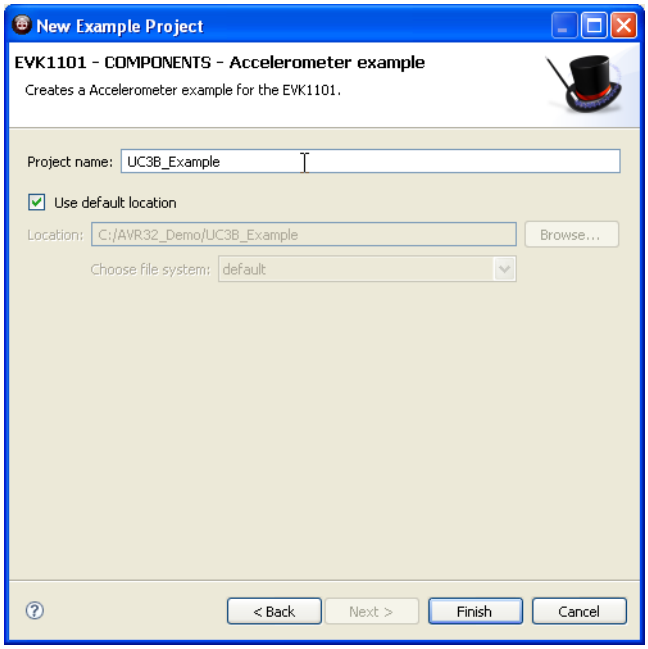

Create a new project by clicking *File>New>AVR32 Example Project.*

***Figure 5-11.*** Select project example



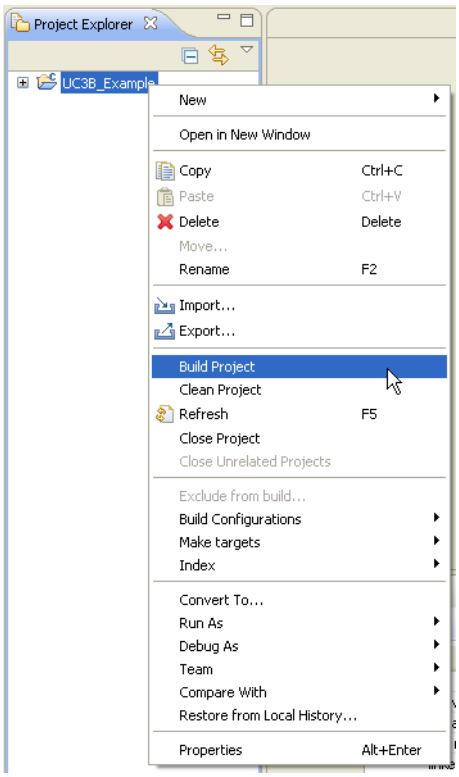

Select **EVK1101 – Components - Accelerometer example**, then click **Next**

***Figure 5-12.*** New project name



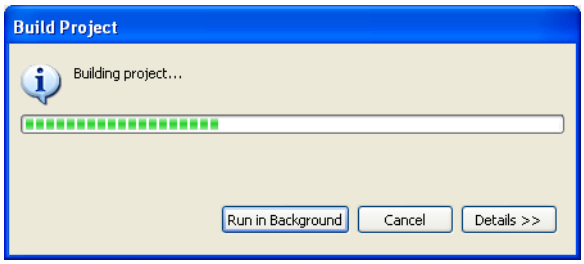

Enter a name for the project, and click **Finish**.

***Figure 5-13.*** Build project



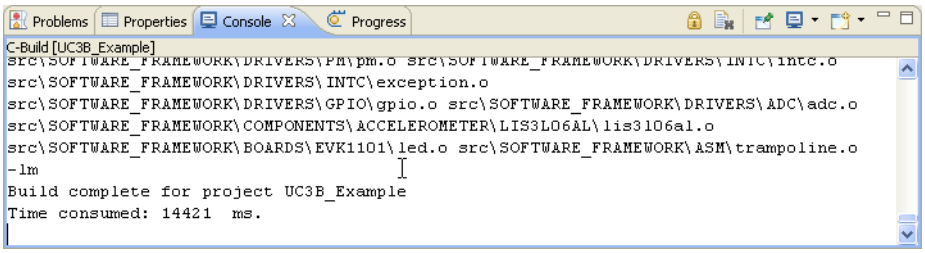

Right-click on the project in **Project Explorer**-view and select **Build Project** (or press CTRL+B).

***Figure 5-14.*** Project build progress




Wait for the project build process to finish.

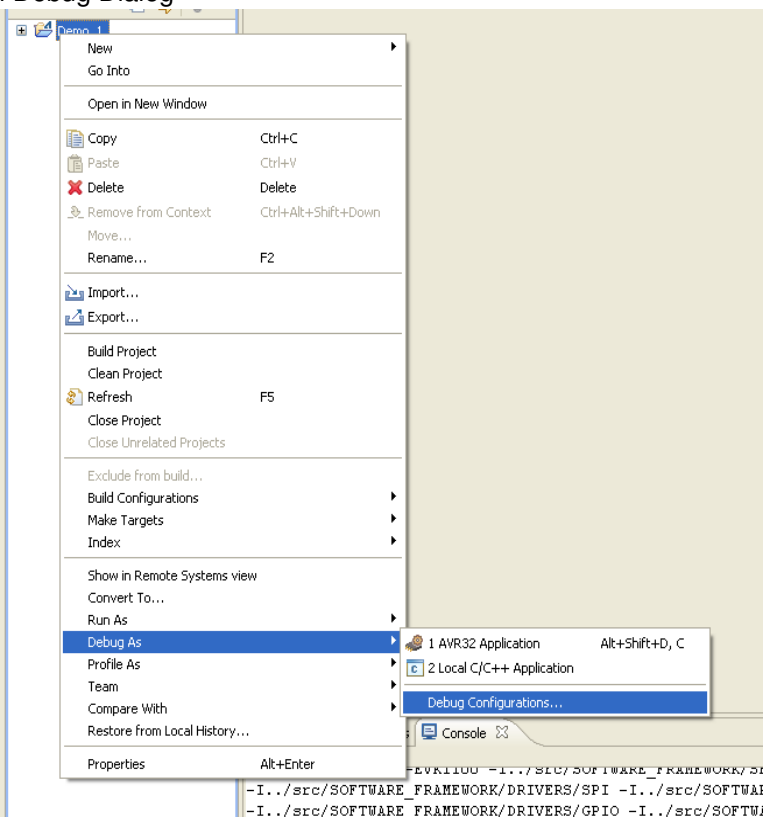
***Figure 5-15.*** Console view




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.

## 5.4 Configure AVR32 Studio for a debug session using trace

***Figure 5-16.*** Open Debug Dialog



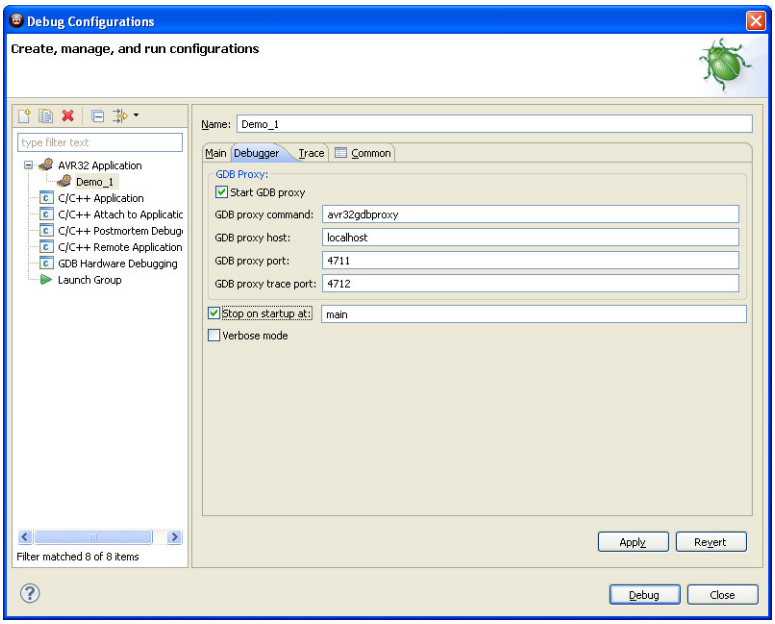

When the build process is finished, right-click on the project in the *Project Explorer* view and select *Debug As>Debug Configurations.*

### 5.4.1 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.

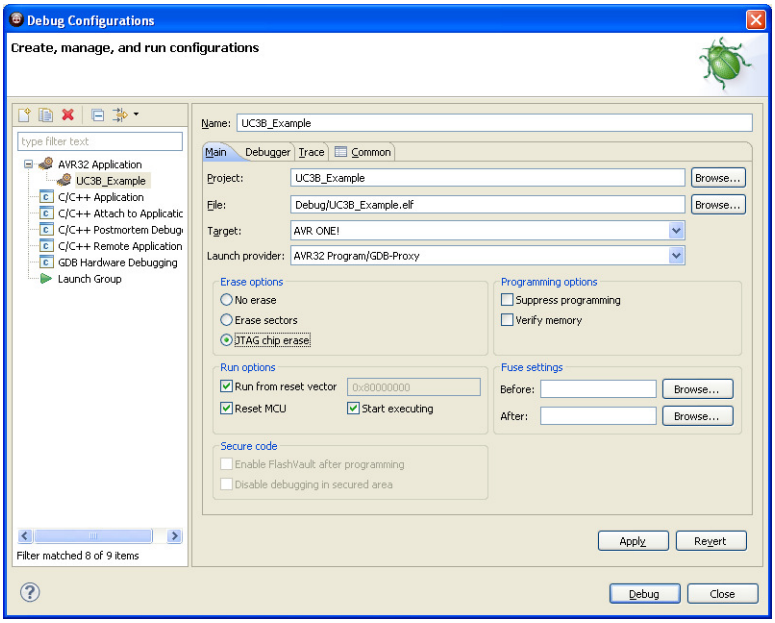
Select the *Debugger* tab and tick the **Stop on startup at: main** option.

***Figure 5-17.*** **Debugger** tab



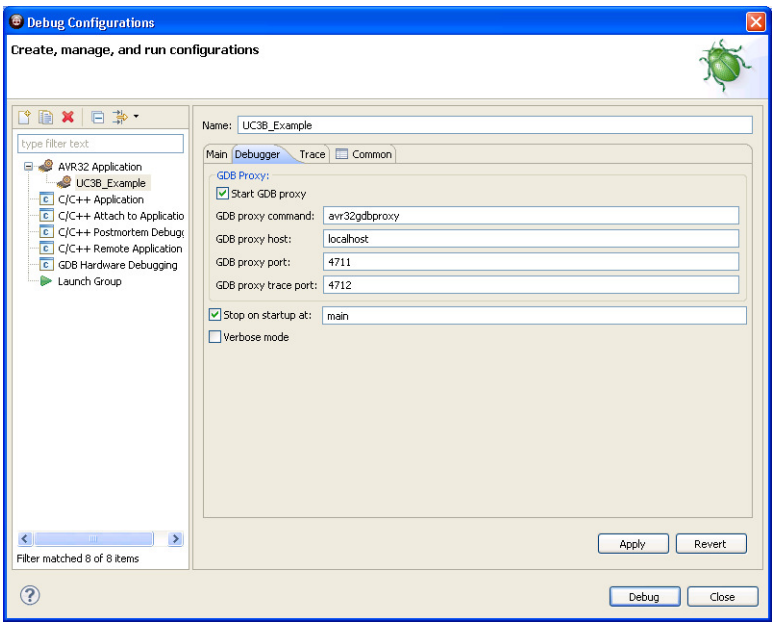
### 5.4.2 Configure the target trace module for program trace

***Figure 5-18.*** Debug configurations, **Main** tab



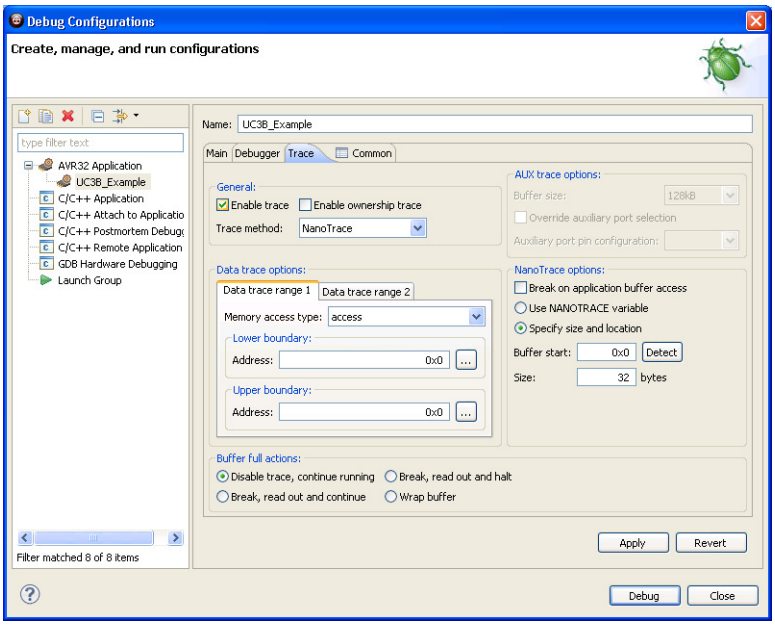

In the **Main** tab, make sure that *Target* is set to ***AVR ONE!***

***Figure 5-19.*** Debug configurations, **Debugger** tab



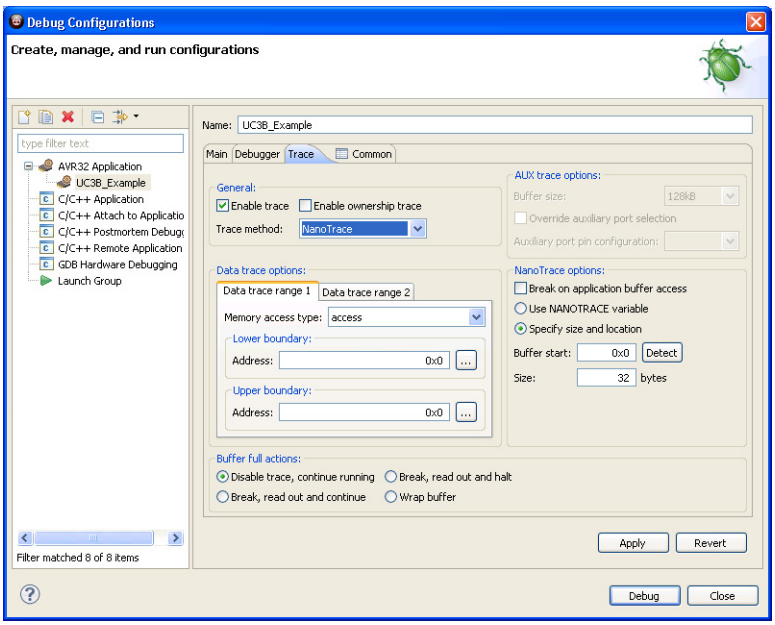

Select the **Debugger** tab and check the checkbox at the option **Stop on startup at: main**.

***Figure 5-20.*** Enable Trace




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

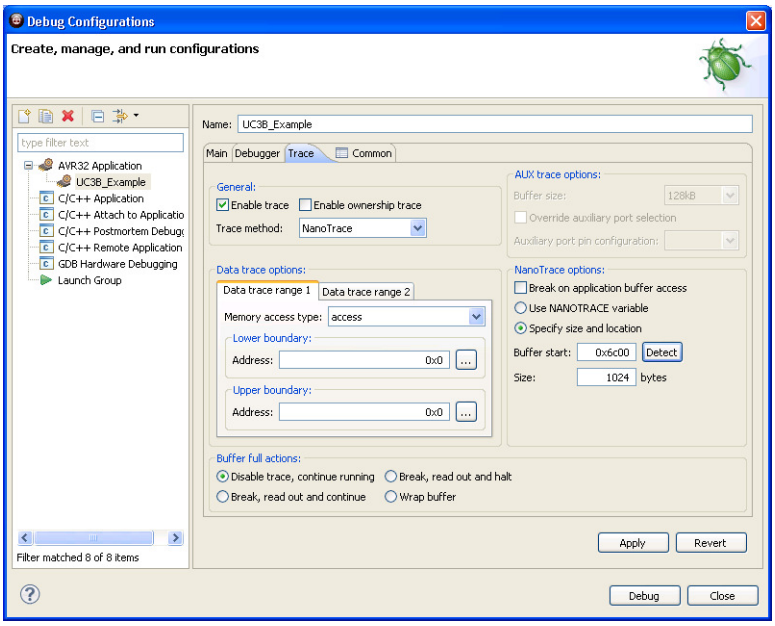
***Figure 5-21.*** Preferred Trace method




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

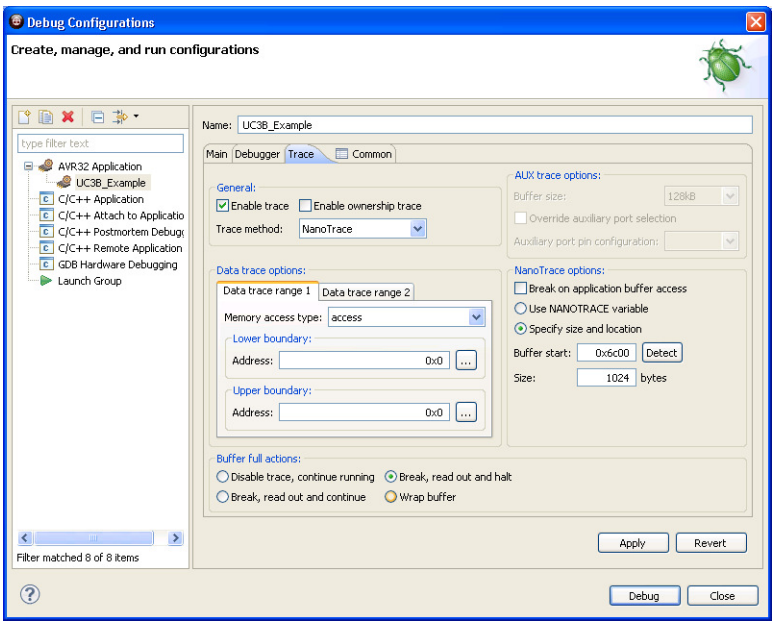
Deselect **Break on application buffer access**.

***Figure 5-22.*** Trace buffer size




Select **Specify size and location** option. Then click **Detect** to configure trace buffer size and location.

***Figure 5-23.*** Buffer full action

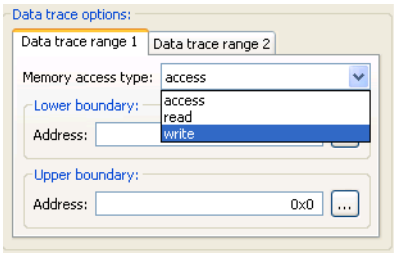



Selected the preferred action when buffer is full. In this case we choose **Break, read out and halt**.

### 5.4.3 Configure the target trace module for data trace

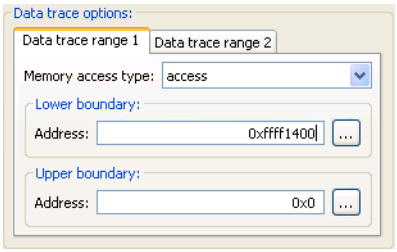
We would like to trace all data written to the debug UART. We do a quick lookup in the datasheet and find that the UART registers are located between 0xffff1400 and 0xffff1d00. Although we only use one UART in this application, we configure the data trace range to cover all UARTs.

***Figure 5-24.*** Memory access type



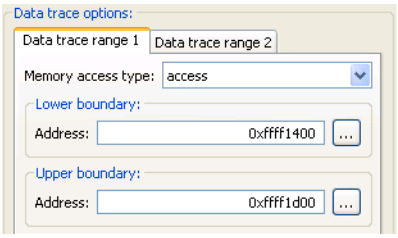

Set Memory access type to **write**.

***Figure 5-25.*** Data trace lower boundary



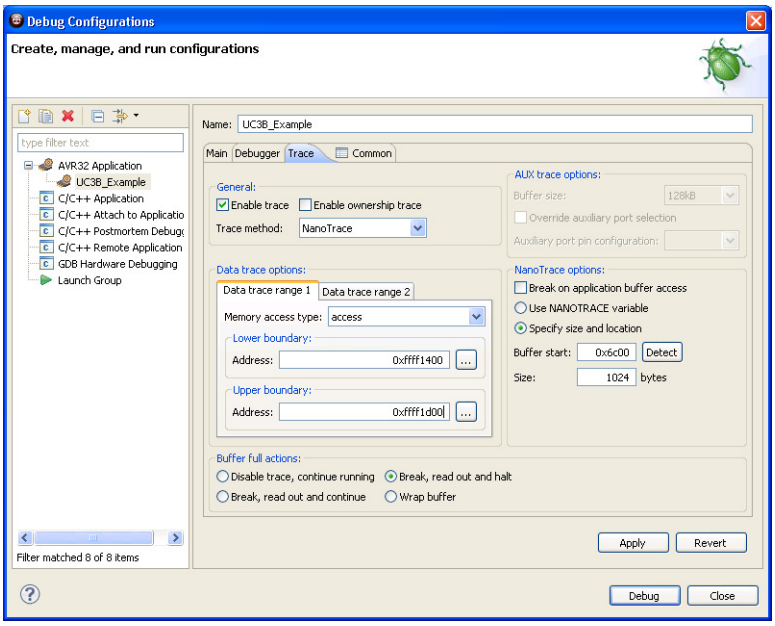

Set lower boundary to 0xffff1400.

***Figure 5-26.*** Data trace upper boundary




Set upper boundary to 0xffff1d00.

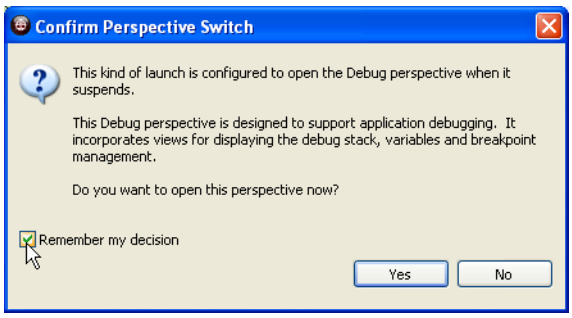
***Figure 5-27.*** Configured trace



## 5.5 Start a debug session and configure the debugger for trace

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

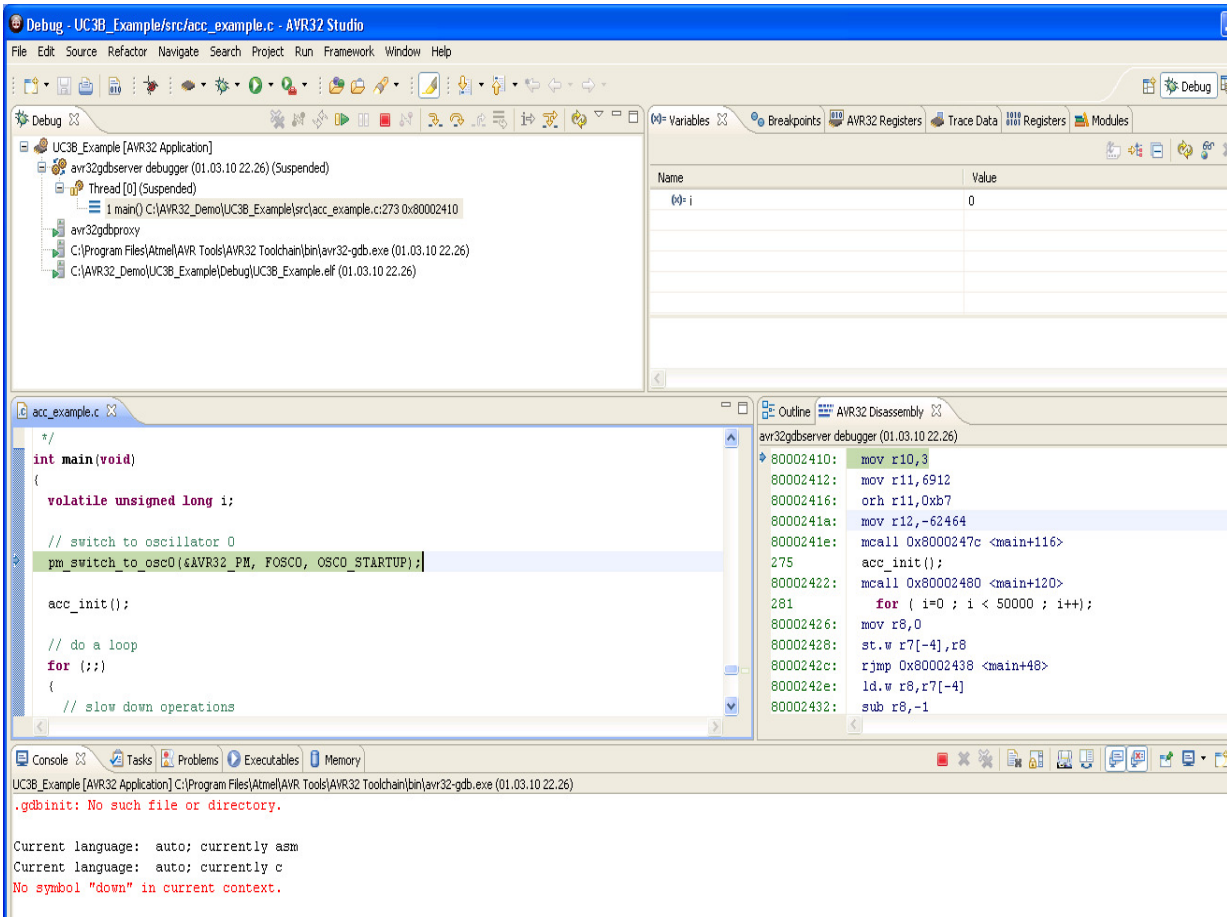
***Figure 5-28.*** Switching perspective




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**.

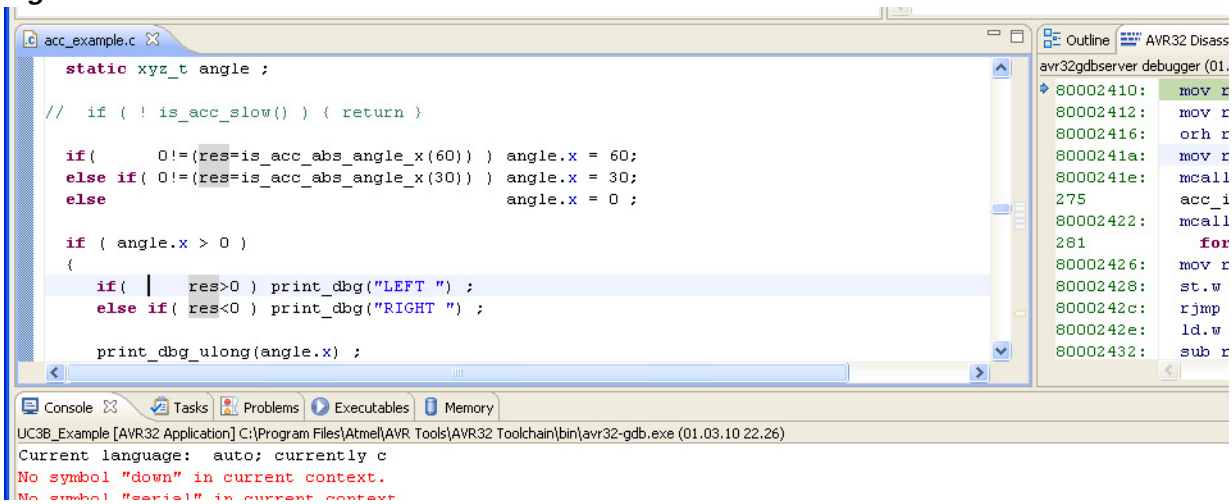
Wait until the target has stopped at the first instruction in the `main()` routine.

***Figure 5-29.*** Program halted at `main()`



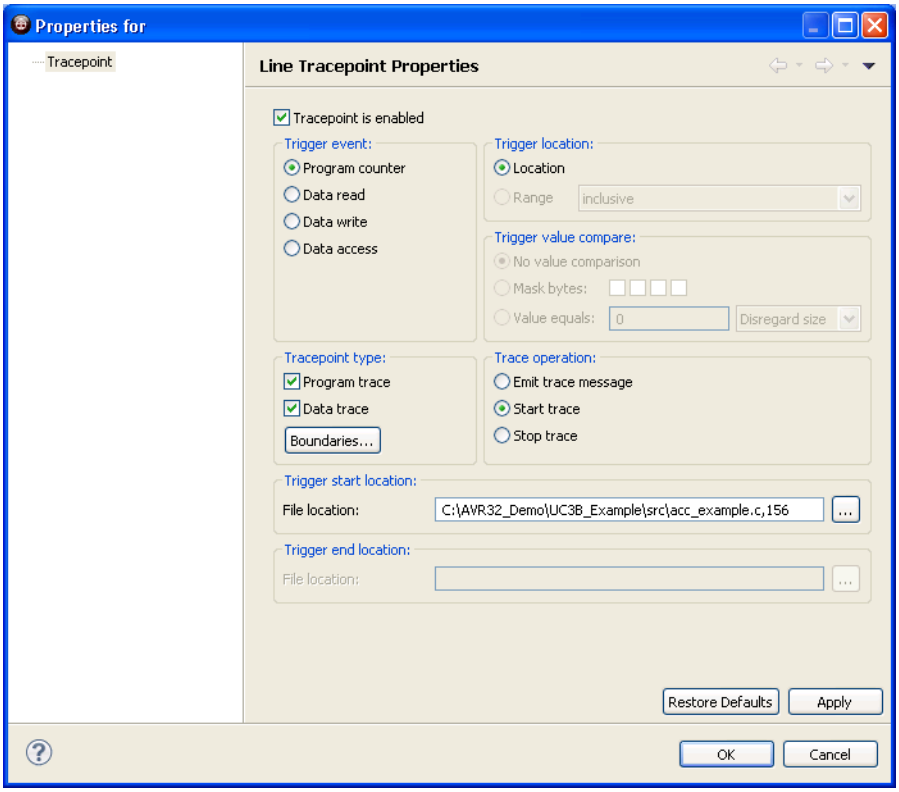
## 5.6 Add start and stop trace-points

***Figure 5-30.*** Source code editor




Scroll up to line 156 in the file acc_example.c and right-click at the left edge of the editor. Select **Add Tracepoint...** from the pop-up menu.

***Figure 5-31.*** Tracepoint (Start)



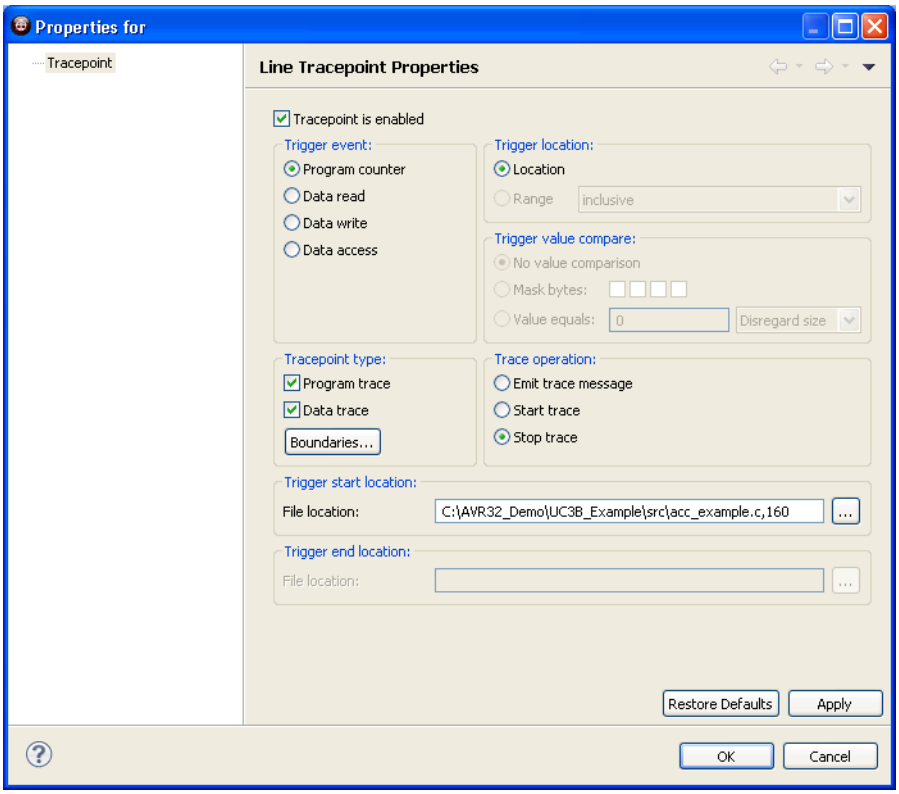

Set Tracepoint Configuration values:

- 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 line 160 in the file acc_example.c and right-click at the left edge of the editor. Select **Add Tracepoint...** from the pop-up menu.

***Figure 5-32.*** Tracepoint (Stop)



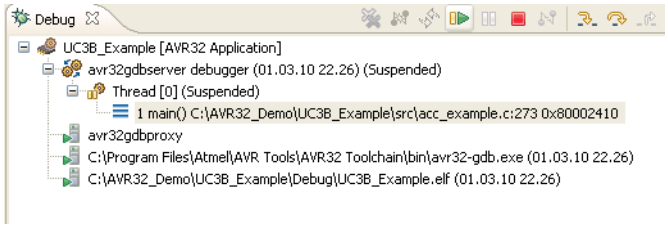

Set Tracepoint Configuration values:

- 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.

## 5.7 Start the trace debug session

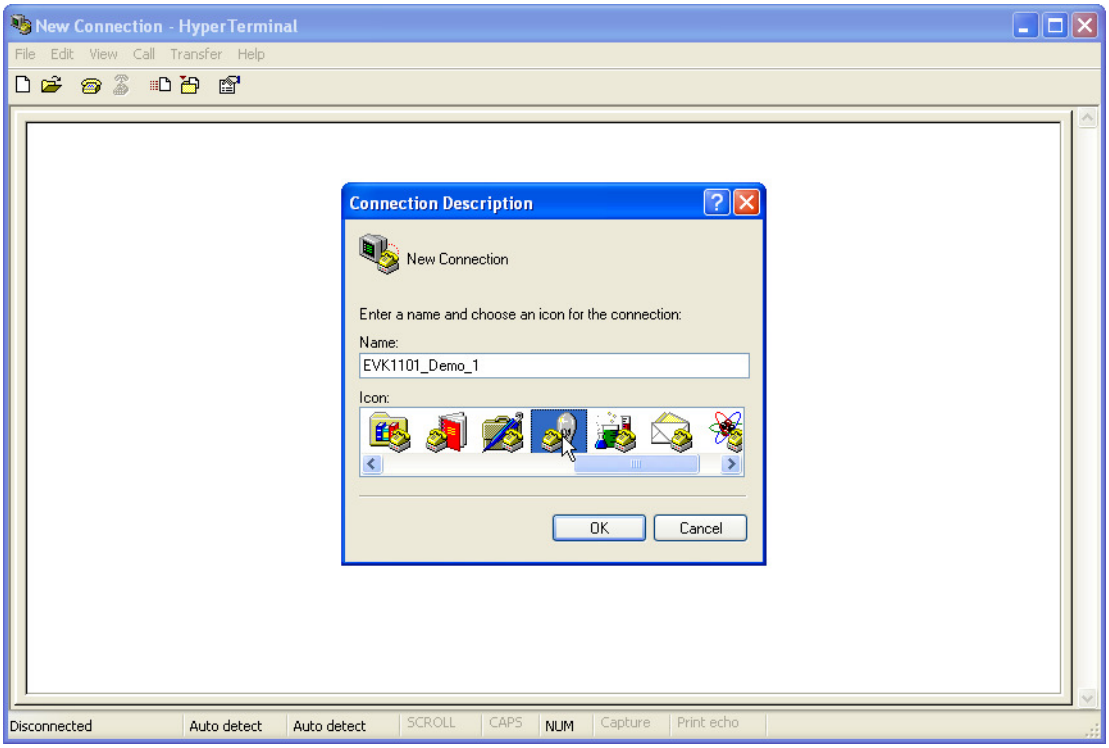
***Figure 5-33.*** Resume debug session




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

Start a serial port terminal to view the output from the debug UART. To make it simple, we just start Hyperterminal. Click on *Start>All Programs>Accessories>Communications>Hyperterminal*.

***Figure 5-34.*** New Hyperterminal

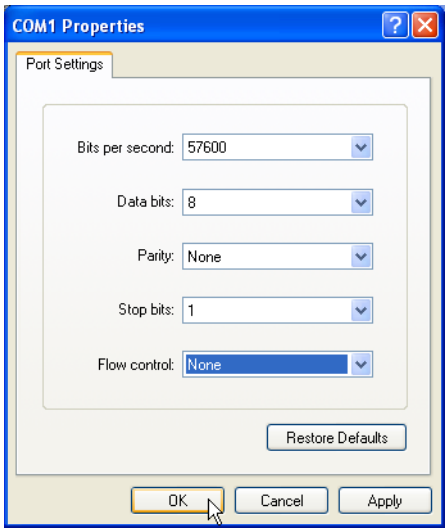



Enter a name for the session and click **OK**.

***Figure 5-35.*** Hyperterminal port selection


Seletc the com-port that you connected the EVK1101 to (in this case we use **Com1**).

***Figure 5-36.*** Hyperterminal port configuration

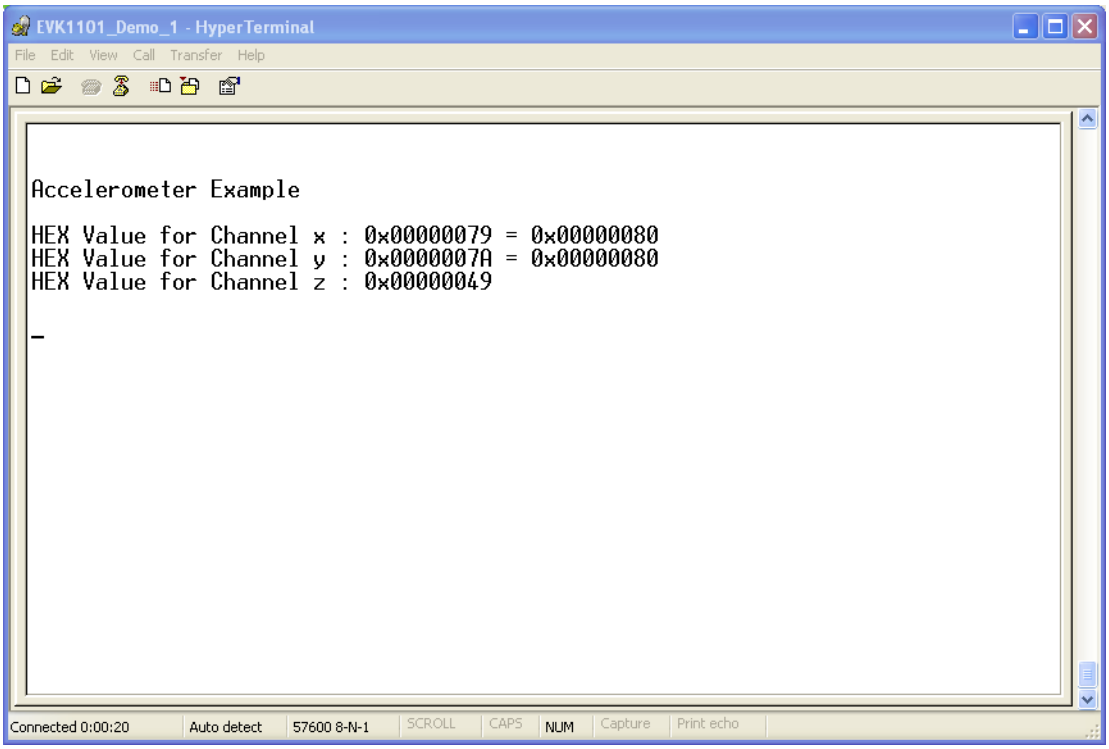



Set port parameters:

- Bits per second: 57600
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: None

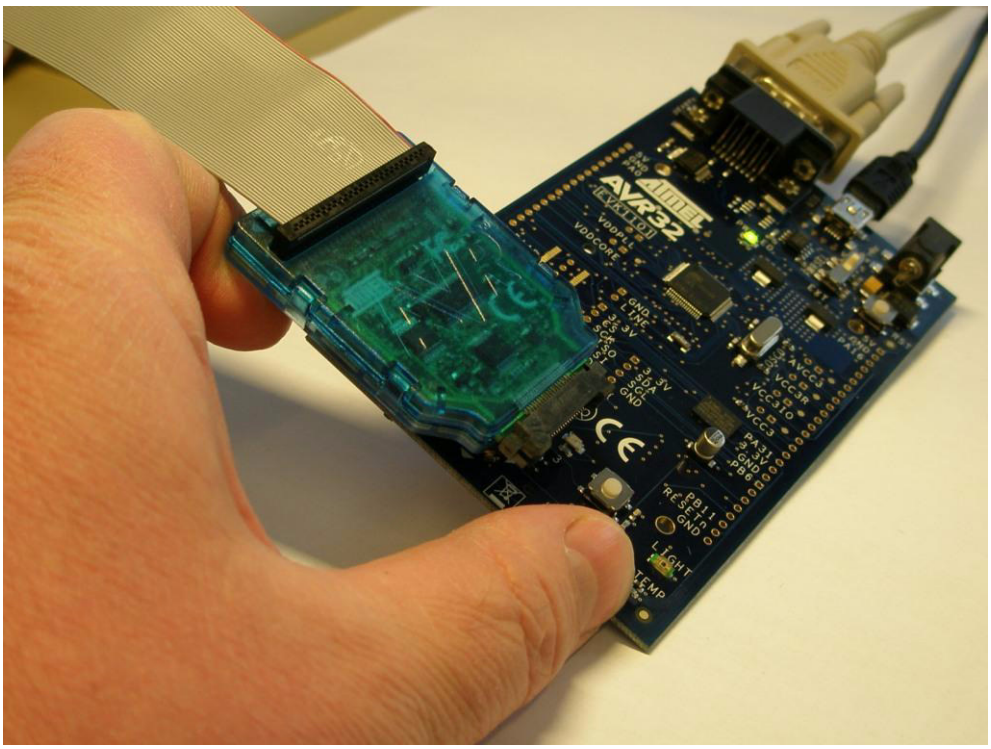
Click **OK**.

***Figure 5-37.*** Demo application output



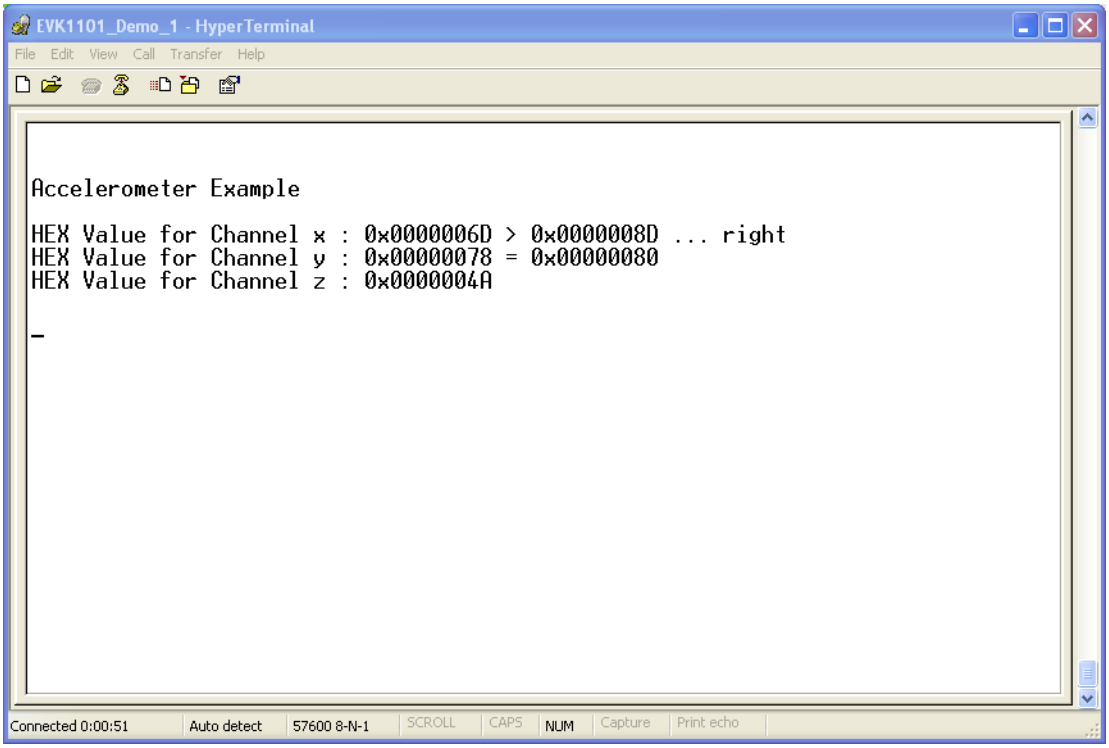

Tilt the EVK1101 board carefully as shown in the photograph. Start with the board laying flat on the table, and increase the tilt slowly.

***Figure 5-38.*** Tilting the EVK1101 board



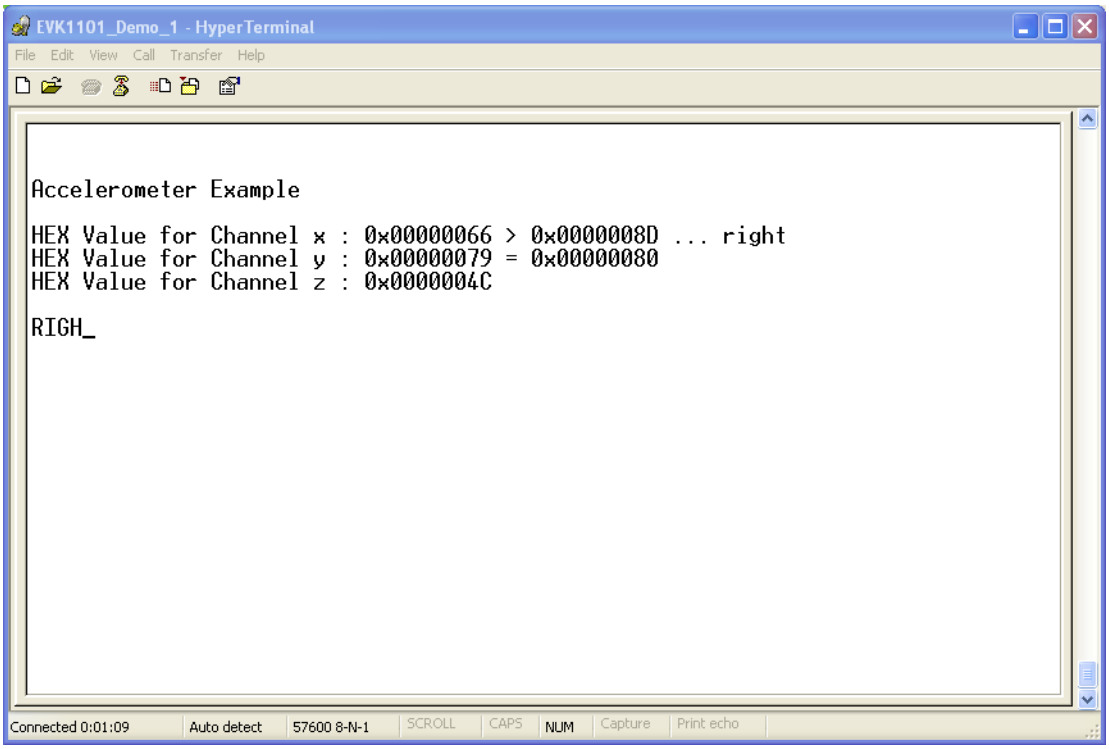
When the tilting angle reaches a certain value, the debug output notifies you about which direction the board is being tilted.

***Figure 5-39.*** Tilt direction indicator




When the tilt angle reaches 30 degrees, the program wants to print an additional message saying **RIGHT30**. When this happens, the program counter hits the start tracepoint, and trace data will start being collected.

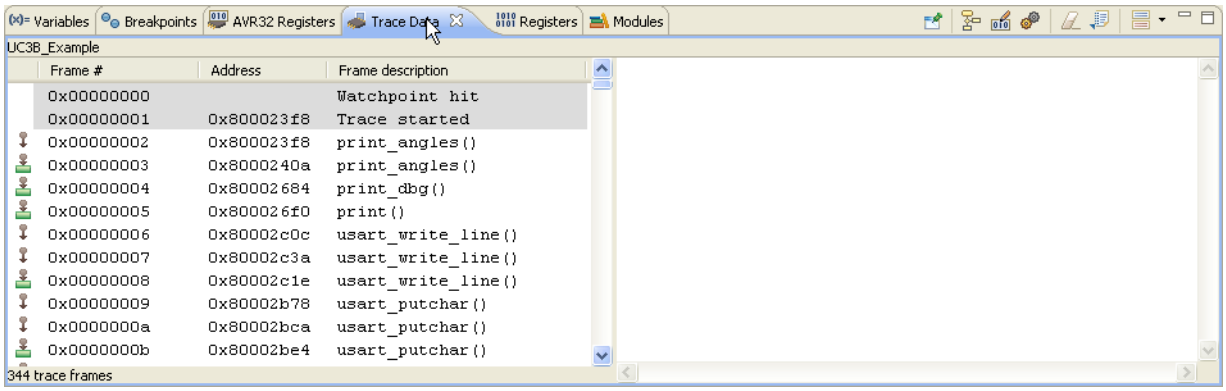
***Figure 5-40.*** Debug output stopped when trace buffer full




When the trace buffer in the target MCU is full, the program will break, and trace data will be uploaded to AVR32 Studio for inspection.

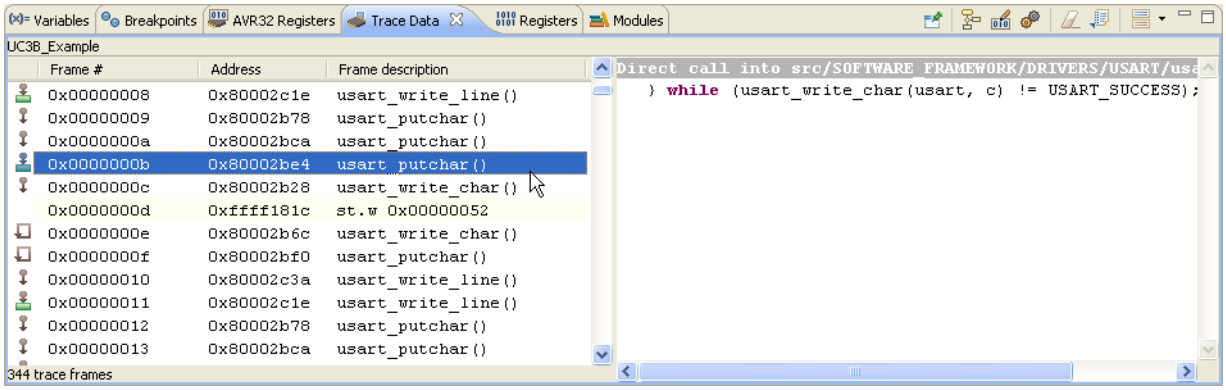
## 5.8 View trace data

***Figure 5-41.*** Trace data view



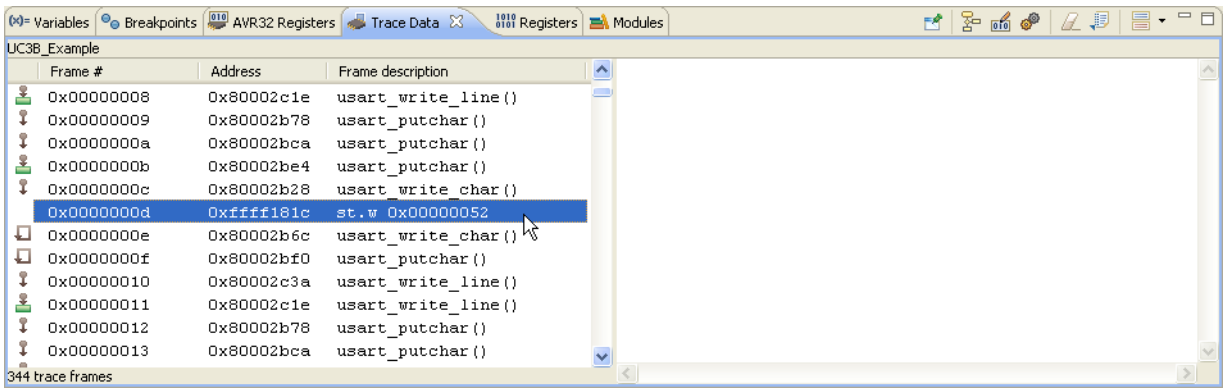

Click on the **Trace Data** tab to view the trace frames.

***Figure 5-42.*** Reconstructed source code



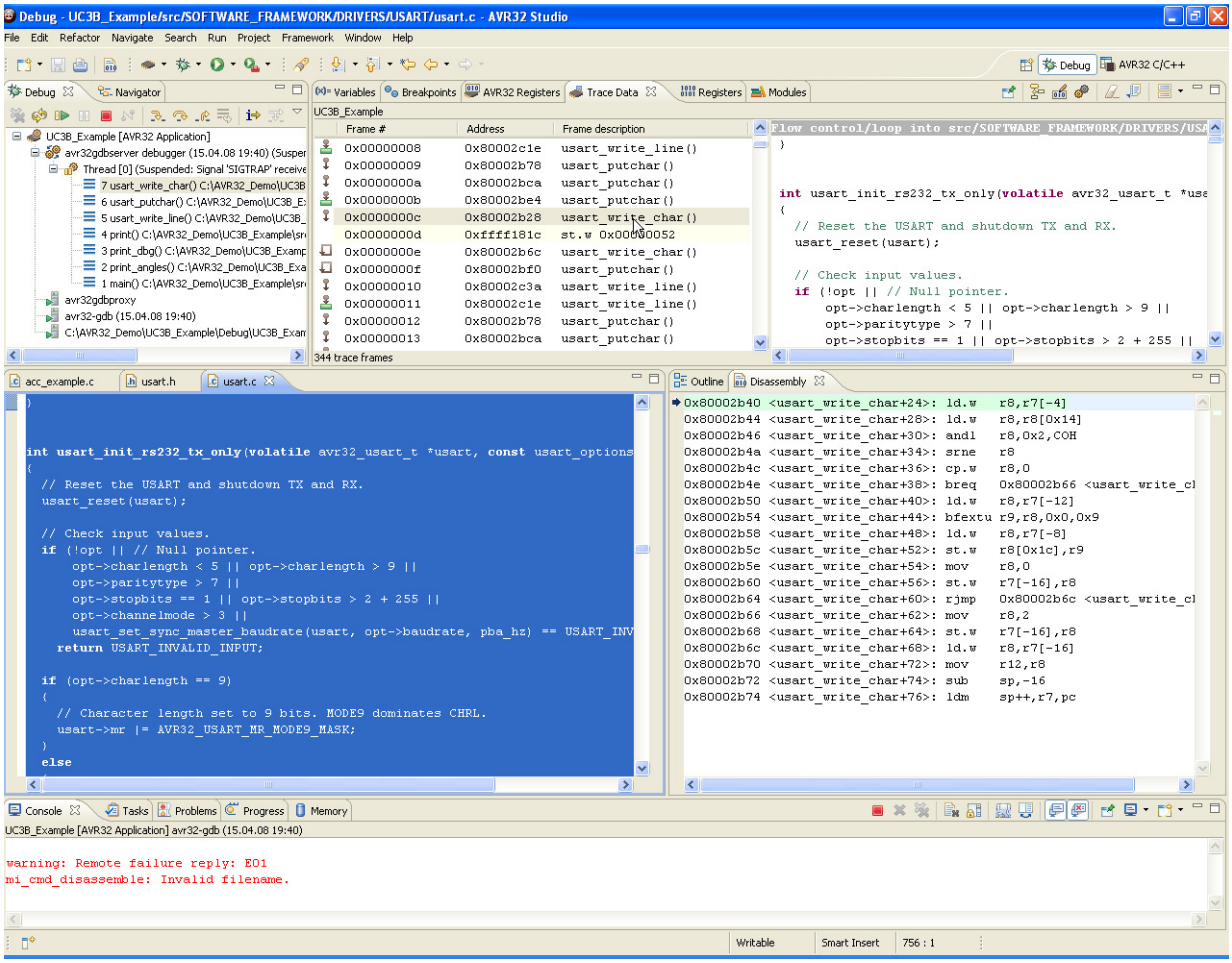

Click on a trace frame to view the reconstructed code that was executed by the MCU.

***Figure 5-43.*** Data trace frame




A data trace frame showing a byte being written to the debug UART transmit register. By enabling data trace only, we can see all characters being sent to the terminal.

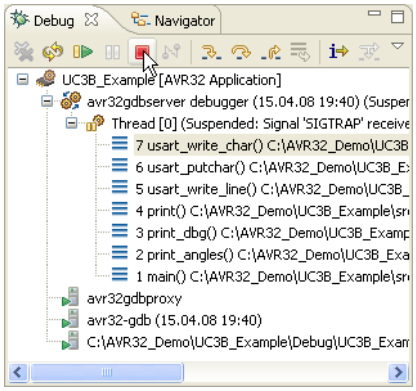
***Figure 5-44.*** Source code in editor




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

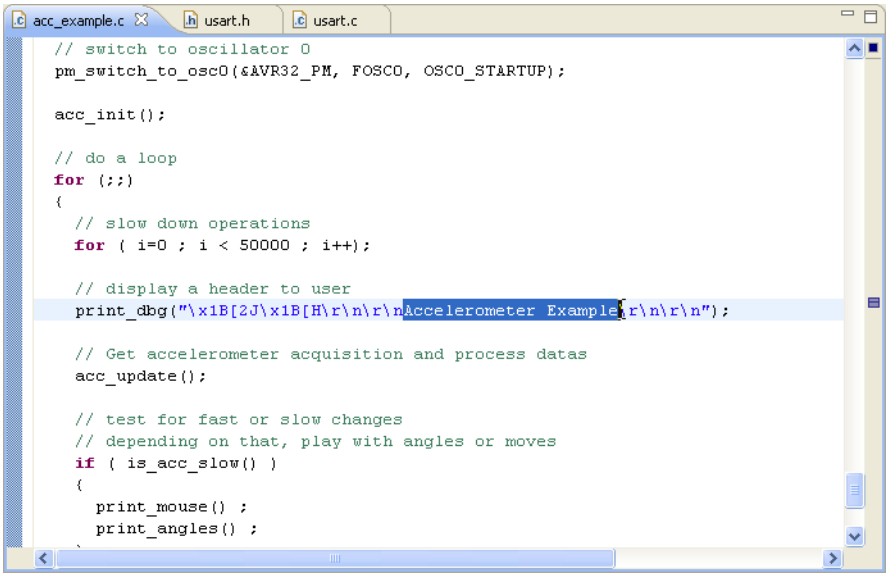
## 5.9 Modify the application

***Figure 5-45.*** Terminate the debug session




Click the **Stop** icon to terminate the debug session.

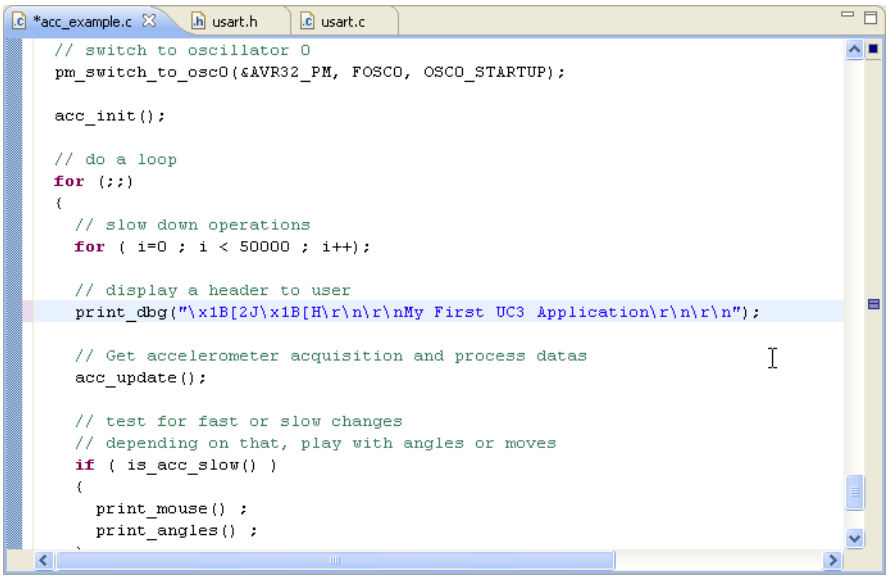
***Figure 5-46.*** Go to source code




Edit the string.

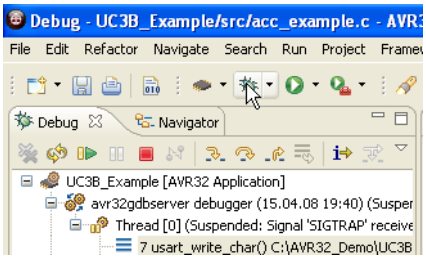
Click on the **acc_example.c** tab and scroll to line 276. Delete the text Accelerometer Example and replace it by your own text.

***Figure 5-47.*** Modified source code




When the source code has been edited, simply restart the previous debug session.

***Figure 5-48.*** Restart debug session

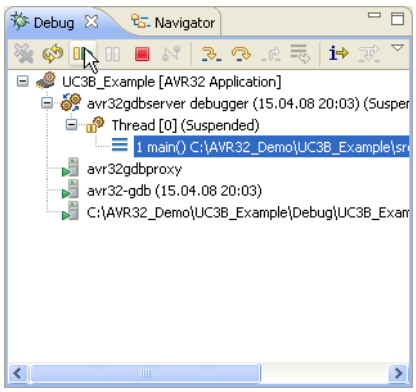



Click on the **Debug** icon to start a new debug session using the same launch configuration as the previous session.

***Figure 5-49.*** Save modifide source code

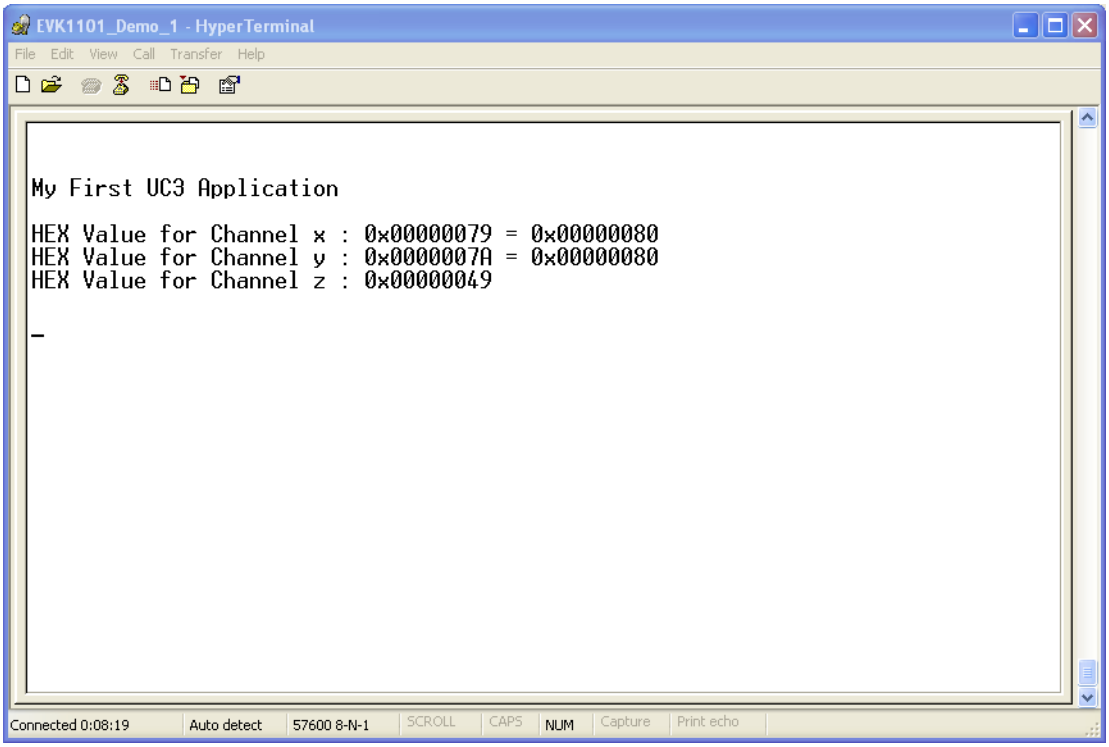

If you did not save the modifide source code, you will be notified now (click **Yes**). After the source code has been saved, AVR32 Studio will re-compile the application and program the target MCU before starting the debug session. The code will run break at `main()` again.

***Figure 5-50.*** Resume debug session




Click on the **Resume** icon to resume the debug session.

***Figure 5-51.*** Modified debug output




Observe the modified output containing your own text.

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 6

# Firmware Upgrade

## 6.1 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.

## 6.2 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 6-1.*** Firmware version test

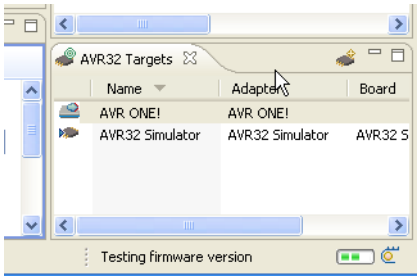
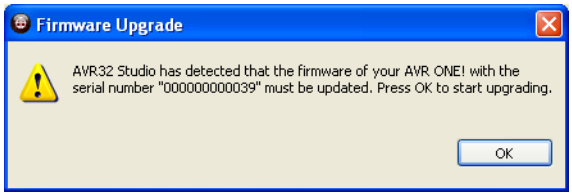


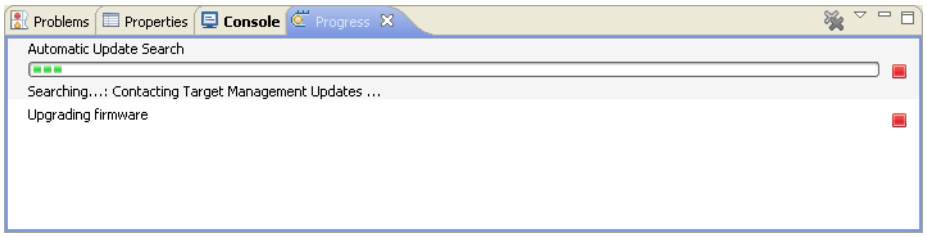

***Figure 6-2.*** Firmware upgrade message




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

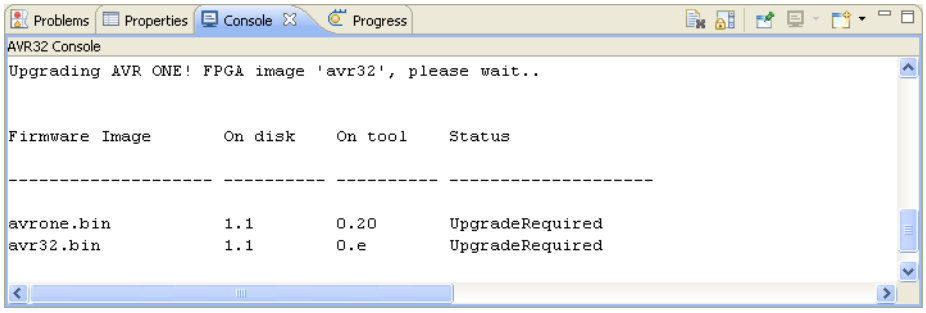
Firmware upgrade progress can be monitored by activating the *Progress* view.

***Figure 6-3.*** Firmware upgrade progress




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

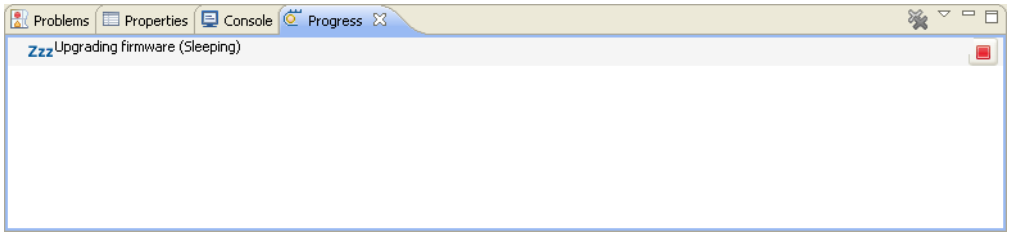
***Figure 6-4.*** Firmware upgrade report




## 6.3 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 6-5.*** 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

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Microchip:

ATAVRONE-PSU