AVR ONE!

Quick-start Guide

.....

.....

EVK1100 + Windows[®]



32103C-AVR ONE!-02/10

Section 1

Introducti	on	1-1	
1.1	General	. 1-1	
1.2	Requirements	1-1	

Section 2

art guide (short version)	2-1
Install Hardware and software	2-1
Create a demonstration project	2-1
Configure target MCU for a debug session using trace	2-1
Start the debug session and configure AVR32 Studio 2.5 for trace	
Start the trace debug session	2-2
	Install Hardware and software Create a demonstration project Configure target MCU for a debug session using trace Start the debug session and configure AVR32 Studio 2.5 for trace

Section 3

ardware preparation

Section 4

Software	Installation	4-1
4.1	Download the software	4-1
4.2	Download the two installation files to your disk	4-2
4.3	Install AVR32 GNU Toolchain	4-2
4.4	Install AVR32 Studio 2.5	4-8
4.5	Connect the AVR ONE! to power and USB host	4-11
4.6	Install AVR ONE! Driver	4-12

Section 5

5.1	Connect the AVR ONE! to the EVK1100	5-1
5.2	Connect the EVK1100 to power	5-2

Section 6

Create c	lemo ap	plication	6-1
6.1	Start A	VR32 Studio	6-1
6.2	Configu	ure adapter and target	6-2
	6.2.1	Add and configure the adapter (AVR ONE!)	6-3
	6.2.2	Configure target board and MCU	6-5
	6.2.3	Target MCU Chip erase	6-7
6.3	Create	a demonstration project	6-8
6.4	Configu	re AVR32 Studio for a debug session using trace	. 6-11
	6.4.1	Create a new debug launch configuration	. 6-12
	6.4.2	Configure the target trace module for program trace	. 6-13
	6.4.3	Configure the target trace module for data trace	. 6-16
6.5	6.5 Start a debug session and configure the debugger for trace		. 6-17



i.

Table of Contents (Continued)

	6.5.1	Add start and stop trace-points	6-19
6.6	Start the	e trace debug session	6-23
6.7	Modify t	he code and restart the debug session	6-26

Section 7

Firmware	Upgrade	7-1
7.1	Firmware upgrade overview	7-1
7.2	Firmware version test and upgrade	7-1
7.3	Adapter in use	7-2



ii



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).

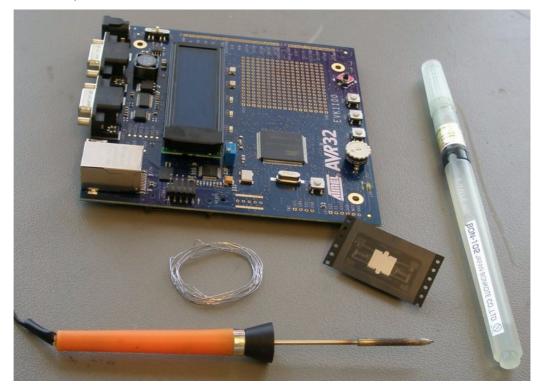


Figure 3-1. Required hardware and tools for installation of Mictor38

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.

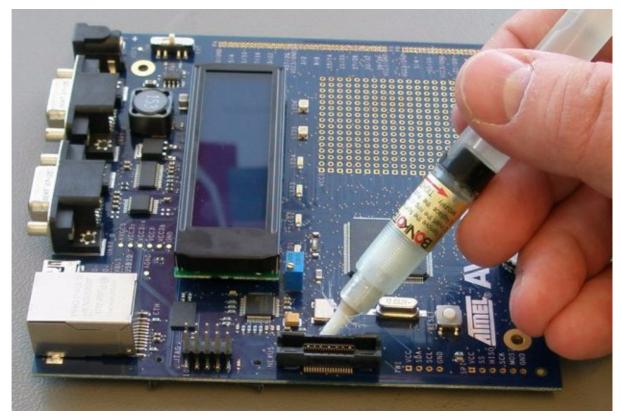
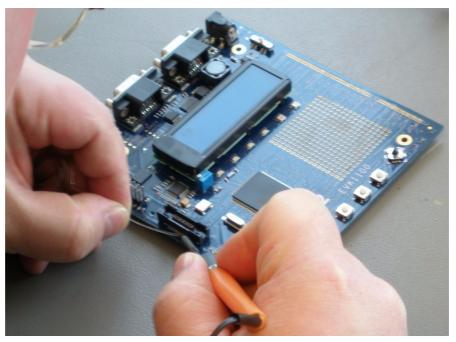


Figure 3-3. Place Mictor38 and apply flux



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.

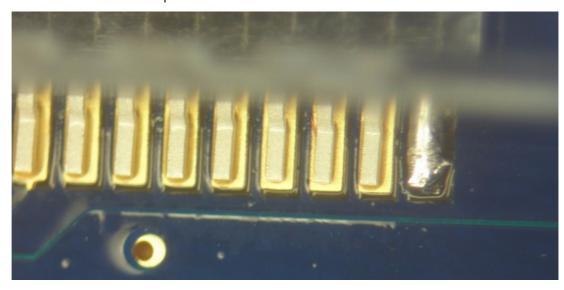
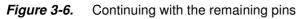
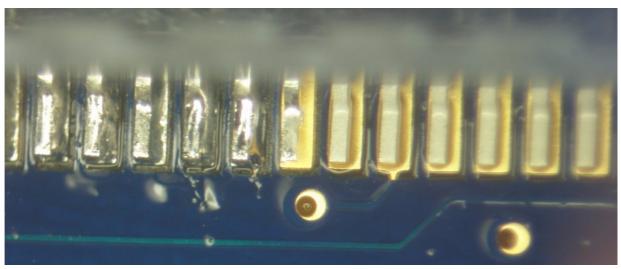


Figure 3-5. Soldered corner pin



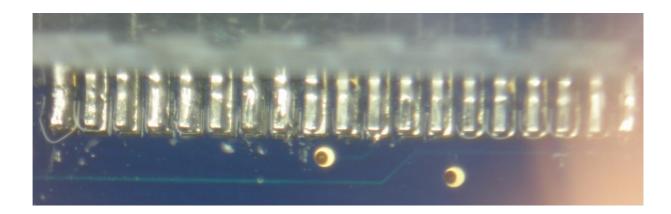
Solder the remaining pins.





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

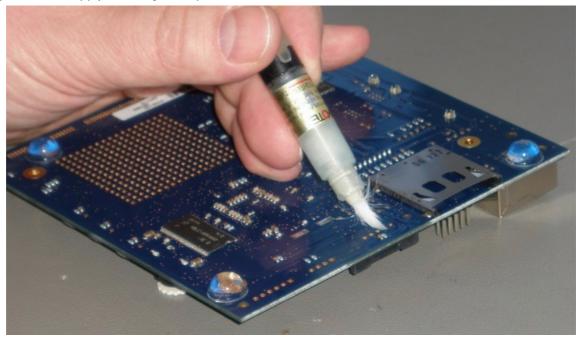
Figure 3-7. All pins finished





Turn the board and apply flux on the ground pins.

Figure 3-8. Apply flux on ground pins



Solder the five ground pins.

Solder ground pins

Figure 3-9.





Section 4

Software Installation

4.1 Download the software

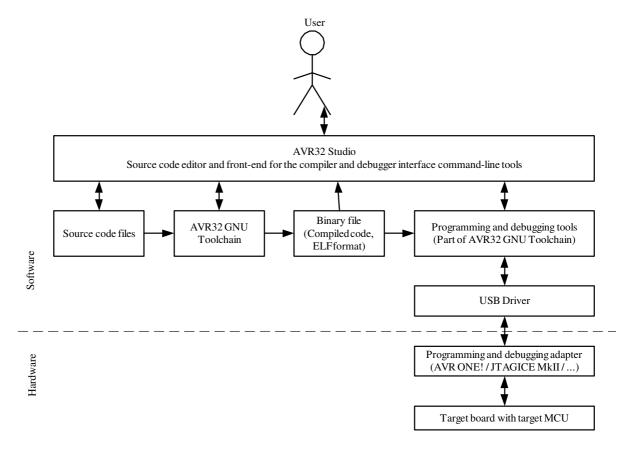
To use the AVR ONE!, you must download and install two software packages:

- 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.

AVR32 Toolchain - InstallSh	ield Wizard	×
	Welcome to the InstallShield Wizard for AVR32 Toolchain Version: 2.4.2	
	< <u>Back</u> <u>N</u> ext > Cancel	

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

AVR32 To	olchain - InstallShield Wizard	×
Installati	on folder	
	Install AVR32 Toolchain to: C:\\Atmel\AVR Tools\AVR32 Toolchain	כ
InstallShield -	< <u>Back</u> Cancel)

Check that the installation folder is correct and click Next.



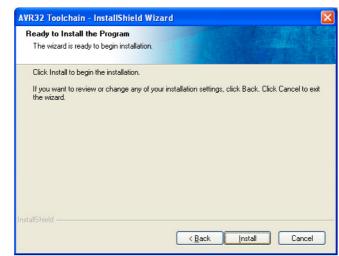


Figure 4-5. AVR32 GNU Toolchain installer configuration finished

Click Install.

Figure 4-6. AVR32 GNU Toolchain installation progress indicator

AVR32 Toolchain - InstallShield Wizard	
Setup Status	
AVR32 Toolchain is configuring your new software installation.	
Installing	
C:\\AVR32 Toolchain\avr32\lib\ldscripts\avr32elf_uc3b0256es.xwr	
InstallShield	
	Cancel

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.

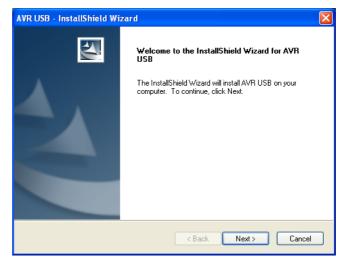


Software Installation

Figure 4-7. USB Drivers installation start

AVR32 Toolchain - Ins	tallShield Wizard	×
Setup Status		
AVR32 Toolchain is co	nfiguring your new software installation.	_
Installing	Installing USB drivers	
InstallShield		
		Cancel

Figure 4-8. USB Driver installer welcome



Click Next.



Figure 4-9. USB Drivers licence agreement form

AVR USB - InstallShield Wizard	×
License Agreement Please read the following license agreement carefully.	
Welcome to AVR USB drivers from Atmel Corporation. The tools are free of charge and may be freely copied and distributed in its original form. The tools runs under Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP , Microsoft Windows XP 64, Microsoft Windows Vista and Microsoft Windows Vista 64. Copyright © ATMEL Corporation. All rights reserved. AVR is trademark of ATMEL Corporation Windows is a trademark of Microsoft Corporation	
Igaccept the terms of the license agreement Print Ido not accept the terms of the license agreement InstallShield Kack Next > Cancel	

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

Figure 4-10. USB drivers installer configuration finished

AVR USB - InstallShield Wizard	
Ready to Install the Program The wizard is ready to begin installation.	
Click Install to begin the installation. If you want to review or change any of your installat the wizard.	ion settings, click Back. Click Cancel to exit
InstallShield	< Back Inst Cancel

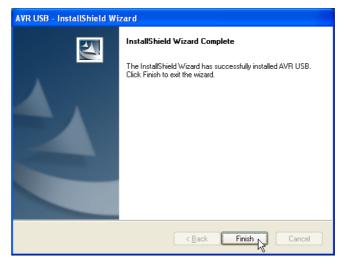
Click Install.



Figure 4-11. USB Drivers installation progress indicator

AVR USB - InstallShield Wizard	
Setup Status	No.
The InstallShield Wizard is installing AVR USB	
Installing	
C:\Program Files\Atmel\AVR Tools\usb\windrvr6.inf	
InstallShield	
יישר שונאריי	Cancel

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

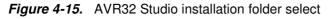
Double-click on the AVR32Studio-2.5.x.exe file to start the installation process.

Figure 4-14. AVR32 Studio 2.5 installer welcome



Click Next.





AVR32 Stu	dio - InstallShield Wizard			
	Pestination Location Ider where setup will install files.	-76		
	Install AVR32 Studio to: C:\Program Files\Atmel\AVR Tools			Change
InstallShield -		< Back	Next >	Cancel

Check that the installation folder is correct and click Next.

Figure 4-16. AVR32 Studio installer configuration finished

AVR32 Studio - InstallShield Wizard	$\mathbf{\times}$
Ready to Install the Program The wizard is ready to begin installation.	
Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.	
InstallShield]

Click Install to start the installation.



W <mark>R32 Studio - Inst</mark> Setup Status	allShield Wizard			×
The InstallShield Wi	ard is installing AVR3.	2 Studio		
nstallShield ————			C	Cancel

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	Figure 4-18.	AVR32 Studio ins	stallation process	complete
---	--------------	------------------	--------------------	----------

AVR32 Studio - InstallShield Wizard			
	InstallShield Wizard Complete The InstallShield Wizard has successfully installed AVR32 Studio. Click Finish to exit the wizard.		
-	< Back Finish Cancel		

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

Found New Hardware Wizard				
	Welcome to the Found New Hardware Wizard			
	Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission). <u>Read our privacy policy</u>			
	Can Windows connect to Windows Update to search for software?			
	 Yes, this time only Yes, now and every time I connect a device No, not this time 			
	Click Next to continue.			
< Back Next > Cancel				

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



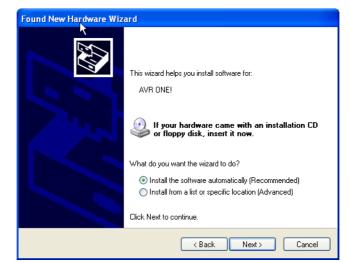


Figure 4-22. Hardware installation wizard configuration

Select Install the software automatically and click Next.

Figure 4-23. Hardware installation in progress

Found New I	Hardware Wizard			
Please wa	it while the wizard sea	rches		
H	AVR ONE!			
		<u>S</u>		
		< Back	Next >	Cancel

Wait for the installation process to complete.



Software Installation



Figure 4-24. Hardware installation wizard complete

Click Finish.



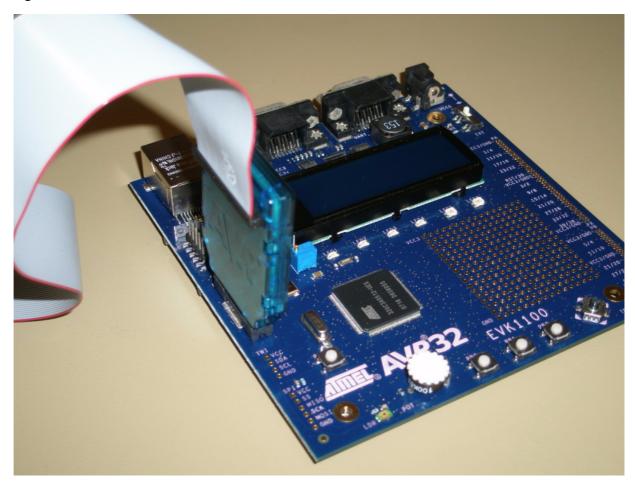


Section 5

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

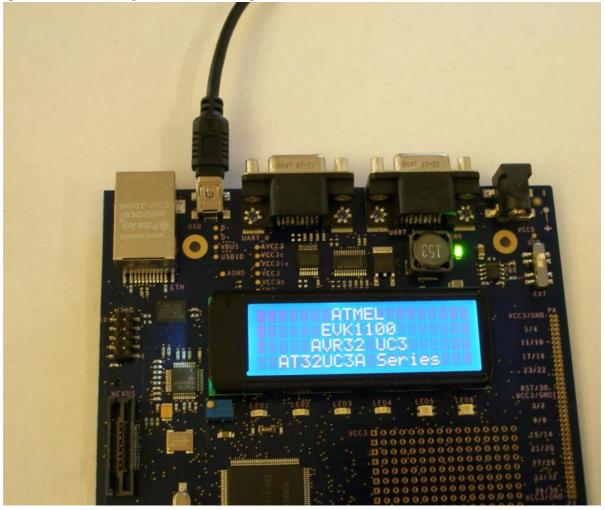


5.2 Connect the EVK1100 to power

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

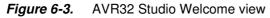


Figure 6-2. AVR32 Studio workspace selection

🙆 Worksp	ace Launcher		X
	rorkspace io stores your projects in a folder called a workspace. orkspace folder to use for this session.		
Workspace:	C:\Avr325tudio2_work		Browse
✔ Use this a	is the default and do not ask again	ОК	Cancel

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**.

AVR32 C/C++ - AVR32 Studio		
Elle Edit Navigate Search Project Run Framework Window Help		
Welcome X		- 8
AV/232	Software tools AVR32 Studio New and noteworthy	
	User quide Getting started Release notes Upgrading projects Product information	
	AVR Studio 4	
Welcome to AVR32 Studio 2.5.0!	About Product information	
This version supports development of standalone applications for the AT32AP7 and AT32UC3 devices, as well as Linux applications for the AT32AP7 family of devices (on Linux only).	Debuggers	
New features:		\rightarrow
Excluded source files are annotated.	AVR Dragon	
More information about new features can be found in the new and noteworthy document. Details about other changes such as bug fixes and known issues is in the release notes.	User guide Product information	
If you're new to AVR32 Studio, you can find out more by reading the following pages. You may want to start reading the project administration introduction or simply creating an MAVR32 Example Project to run on your development kit.	AVR ONE!	
Should you experience any problems when using AVR32 Studio, please let us know by e-mailing us at avr32@atmel.com. Please attach the workspace log which can be obtained from the "Error Log" view. You may also want to read the "Troubleshooting" section of the user guide.	User guide Product information	
Have fun!	AVR32 Simulator	
The AVR32 Studio Team	User guide Tutorials	
	JTAGICE mkII	
	User guide Product information	
	Starter kits	
	NGW100	
	User guide Product information	
1 💼 🙆 🗢 🥸 💆	STK1000	_ ,
	User nuide	×



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

🥔 A	VR32 Targets 🛛		📣 🗖 🗖
	Name 💌	Adapter	Board
	AVR ONE!	AVR ONE!	
*	AVR32 Simulator	AVR32 Simulator	AVR32 S
<			>

Select the AVR ONE!

Figure 6-6. AVR ONE! Selected

🥔 A	VR32 Targets 🙁		💣 🗖 🖬
	Name 💌	Adapter	Board
	AVR ONE!	AVR ONE!	
۲	AVN\$32 Simulator	AVR32 Simulator	AVR32 S
<			>



Create demo application

Figure 6-7.	Selecting the properties view
-------------	-------------------------------

🖹 Problems 🛛 🔲 Properties O errors, O warnings, O infos	📃 Console			
Description A	Resource	Path	Location	

Click on the **Properties** tab.

Figure 6-8. Properties view

Problems	Properties 🛛	💂 Console	<u>₹</u> ∨ ⊓ Б
🛋 AVR ON	El .		
General Details Daisy Chain Information	Name: Binaries AVR32 Studio ke Binary path:	AVR ONE! eps track of the last file used to program a target. The name and date is show below.	
	Binary date:	Thu Jan 01 01:00:00 CET 1970	

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 typ	e
--	-------------	----------------------------------	---

Problems	Properties 🛛 📃 🕻	Console	
AVR ONE!			
General	Details		
Details	Debugger/programmer	: AVR ONE! Device:	UC3A0512
Daisy Chain Information	Clock source:	Internal RC oscillator 🕑 Board:	EVK1100
	▼ Connection		
	Serial number: 0	0000000015	
	Connection:	sb	
	COM Port:		
	- Clock		
	JTAG Clock	•••••••••••••••••••••••••••••••••••••••	1 1 1 1
	aWire Clock		
	Read Apply 💧	Value is out of recommended range, please check data sheet!	

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





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.



6-5

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.



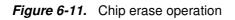
6.2.3 Target MCU Chip erase

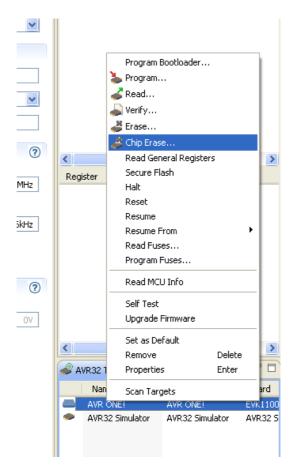
If the EVK1100 evaluation board is brand new, or if it still contains the original demo application (Control Panel Demo), the FLASH lock-bits need to be cleared. Right-click on the AVR ONE! In the *AVR32 Target* view and select **Chip Erase**.

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.

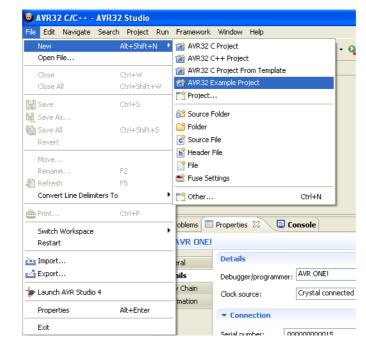
You should now perform the Chip Erase operation.





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

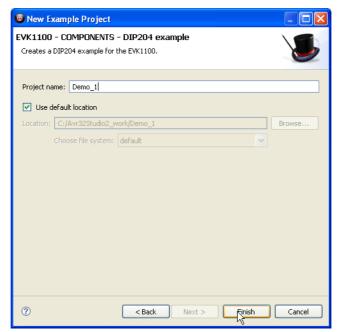
🙆 New E	ram p le	
Select a Creates a	wizard DIP204 example for the EVK1100.	
Wizards:		
type filter	text	
	C3AES Software Framework EVK1100 - APPLICATIONS - Control Panel Demo example EVK1100 - COMPONENTS - Data Flash Memory example EVK1100 - COMPONENTS - DIP204 example EVK1100 - COMPONENTS - SD/MMC Card example EVK1100 - COMPONENTS - SDRAM example EVK1100 - ORIVERS - Analog-to-Digital Converter (ADC) example EVK1100 - DRIVERS - CPU Cycle counter example EVK1100 - DRIVERS - External Interrupt Controller (EIC) example 1 EVK1100 - DRIVERS - External Interrupt Controller (EIC) example 3 EVK1100 - DRIVERS - External Interrupt Controller (EIC) example 3	
	EVK1100 - DRIVERS - General Purpose I/O (GPIO) example EVK1100 - DRIVERS - Interrupt Controller (INTC) example	~
0	< Back Next > Finish	Cancel

Select EVK1100 - Components - DIP204 example, then Next



6-8

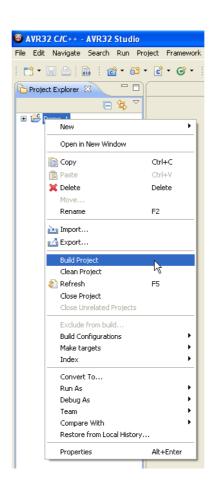
Figure 6-14. New project name



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



Figure 6-15. Build project



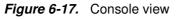
Right-click on the project in **Project Explorer**-view and select **Build Project** (or press CTRL+B).

Figure 6-16. Project build progress

Build Project	
Building project	
	Run in Background Cancel Details >>

Wait for the project build process to finish.





🖹 Problems 🔲 Properties 🖳 Console 🕴	â 🖡	12	• 📬 • 🗖 🗖
C-Build [Demo_1]			
src\SOFTWARE FRAMEWORK\COMPONENTS\DISPLAY\DIP204\dip204.o			~
src\SOFTWARE_FRAMEWORK\BOARDS\EVK1100\led.o src\SOFTWARE_FRAMEWORK\ASM\tr	ampolin	ie.o	
Build complete for project Demo_1			
Time consumed: 18985 ms.			
			~

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

0			
	32 C/C++ - AVR32 Studio		
File Edit	t Navigate Search Project	Run Framework Wind	ow Help
1 📬 -		• 🚳 • 🖻 • 🞯 •	🚳 • 🛞 • 🛛 🗢 • 🏇 • 🔾 • 隆 •
🔁 Proje	ect Explorer 🛛 🗖 🗖	(
	□ 🔄 😜 🔽		
E [24]	Demo 1		
	New		•
	Go Into		
	Open in New Window		
	📄 Сору	Ctrl+C	
	💼 Paste	Ctrl+V	
	💢 Delete	Delete	
		Ctrl+Alt+Shift+Down	
	Move		
	Rename	F2	
	🚵 Import		
	🛃 Export		
	D. (Id Decide th		-
	Build Project Clean Project		
	8 Refresh	F5	
	Close Project	15	
	Close Unrelated Projects		
	Exclude from build		•
	Build Configurations Make Targets		
	Index		
			_
	Show in Remote Systems vi	ew	
	Convert To		
	Run As		AVR32 Application Alt+Shift+D. (
	Debug As Profile As		· · · · · · · · · · · · · · · · · · ·
	Team		C 2 Local C/C++ Application
	Compare With		Debug Configurations

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

Debug Configurations			X
Create, manage, and run con	figurations		Ť.
Image: Second system Image: Second system <th>Name: Demo_1 Main Oebugger Tract GDB proxy: ✓ Start GDB proxy GDB proxy command: GDB proxy host: GDB proxy host: GDB proxy port: GDB proxy trace port: ✓ ✓ Stop on startup at: ✓ ✓ Verbose mode ✓</th> <th>avr32gdbproxy localhost 4711</th> <th></th>	Name: Demo_1 Main Oebugger Tract GDB proxy: ✓ Start GDB proxy GDB proxy command: GDB proxy host: GDB proxy host: GDB proxy port: GDB proxy trace port: ✓ ✓ Stop on startup at: ✓ ✓ Verbose mode ✓	avr32gdbproxy localhost 4711	
Filter matched 8 of 8 items			Apply Revert
?			Debug Close



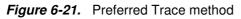
6.4.2 Configure the target trace module for program trace

Figure 6-20. Debug configurations, Trace tab

reate, manage, and run confi	gurations	Ť
Yeno Image: Second	Name: Demo_1 Main Debugger Trace Ceneral: Egable trace Enable ownership trace Trace mgthod: NanoTrace Image: Strate and Strat	ALX trace options: Buffer size: 128kB Override auxiliary port selection Auxiliary port pin configuration: 0 NanoTrace options: Break on application buffer access Use NANOTRACE variable Specify size and location Buffer start: 0x0 Detect Size: 32 bytes
Filter matched 8 of 8 items	Disable trace, continue running Break, read out and b. Break, read out and continue Wrap buffer	Apply

Select the Trace tab and click Enable Trace.





Debug Configurations			
Create, manage, and run cor	figurations		-
Create, manage, and run cor	figurations Name: Demo_1 Main Debugger Irace © Enable trace Enable ownership trace Trace method: Buffered AUX Trace Data trace range 1 Data trace range 2 Memory access v Lower boundary: 0x0 Upper boundary: 0x0	AUX trace options: Buffer size: Override auxiliary port selection Auxiliary port pin configuration: NanoTrace options: Break on application buffer acco Use NANOTRACE variable Specify size and location Buffer start: OXO Detect Size: 32 bytes	ess
Filter matched 8 of 8 items	Buffer full actions: Disable trace, continue running Break, read out and <u>b</u> Break, read out and continue <u>W</u> rap buffer	alt Apply Debug	Revert

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



Figure 6-22. Trace buffer size

Debug Configurations	
Create, manage, and run con	gurations to the second s
Ype filter text	Name: Demo_1 Main Debugger frace © Enable Common AUX trace options: Buffer size: 648 © Deta trace Buffered AUX Trace Override auxiliary port selection Auxiliary port pin configuration: 0 0 Data trace options: Image: Common optication buffer access 0 Data trace range 1 Data trace range 2 Image: Common optication buffer access Lower boundary: Image: Cox option optication Image: Cox option optication Upper boundary: Image: Cox option optication Image: Cox option optication Upper boundary: Image: Cox option optication Image: Cox optication Upper boundary: Image: Cox optication Image: Cox optication Buffer full actions: Image: Cox optication Image: Cox optication Image: Cox optication optication optication Image: Cox optication Image: Cox optication Buffer full actions: Image: Cox optication Image: Cox optication Image: Cox optication Image: Cox optication optication optication optication Image: Cox optication Image: Cox optication Image: Cox optication Image: Cox optication
Filter matched 8 of 8 items	ApplyRevert
?	Debug Close

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.



Create demo application

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

Data trace options:	
Data trace range 1	Data trace range 2
Memory access type:	access
Lower boundary:	access
Address or file location	write
Upper boundary:	
Address or file location	on: 0x0

Set Memory access type to write.

Figure 6-25. Data trace boundaries

Data trace range 1 Data trace range 2
Memory access type: write
Address or file location: 0x0
Upper boundary: Address or file location: 0x0

Select memory location for lower and upper boundaries.

Figure 6-26. Variable address selection dialogue

Address s	election	
Please select a	n address.	
Address	Length	Name
0xc0	4	_int_line_handler_table_5
0xc4	4	_int_line_handler_table_6
0xc8	4	_int_line_handler_table_7
0xcc	4	_int_line_handler_table_8
0xd0	4	_int_line_handler_table_9
0xd4	4	_int_line_handler_table_10
0xd8	4	_int_line_handler_table_11
0xdc	4	_int_line_handler_table_12
0xe0	4	_int_line_handler_table_13
0xe4	12	_int_line_handler_table_14
0×f0	4	_int_line_handler_table_15
0xf4	4	_int_line_handler_table_16
0×f8	4	_int_line_handler_table_17
0xfc	4	_int_line_handler_table_18
0×100	4	_int_line_handler_table_19
0×104	2	pwm_duty
0×108	32	pwm_channel
0×128	2	display 🗸 🗸
?		OK Cancel

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

Data trace options:	
Data trace range 1 Data trace range 2	
Memory access type: write	~
Clower boundary:	
Address or file location:	0x9a4
Upper boundary:	
Address or file location:	0x9a4

6.5 Start a debug session and configure the debugger for trace

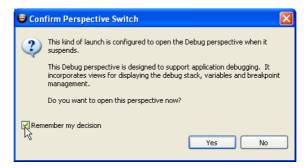
Figure 6-28. Starting a debug session

reate, manage, and run conf	gurations	- TO
Yee Yee type filter text Yee Demo_1 C/C++ AVR32 Application C/C++ Attach to Application C/C++ Attach to Application C/C++ Period C/C++ Remote Application Launch Group	Name: Demo_1 Main Debugger Irace General: Image: Common Image: Common Image: Common General: Image: Common Image: Common Im	AUX trace options: Buffer size: Override auxiliary port selection Auxiliary port pin configuration: NanoTrace options: Break on application buffer access Use NANOTRACE variable Specify size and location Buffer start: OXO Detect Size: 32 bytes alt
C C C C C C C C C C C C C C C C C C C		Apply Reyert

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



Figure 6-29. Switching perspective

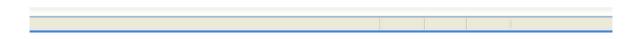


When the debug session starts, AVR32 Studio 2.5 will change to the *Debug* perspective (desktop layout designed for use during debug sessions). You should click **Yes**. To avoid being asked every time you start a debug session, you should also click the **Remember my decision** box before answering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

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

Figure 6-30. Program halted at main()

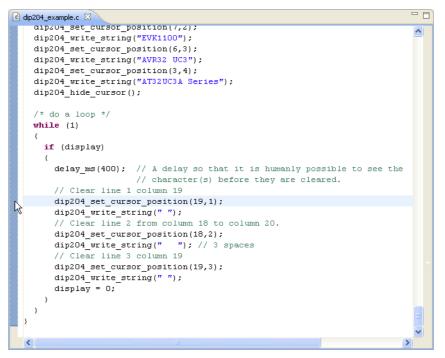
Refactor Navigate Search Run Project Framework Window Help			
] 👜 📾 👄 • 🏇 • 🔕 • 💁 • ! 🛷 ½ - 🖓 - 🏷			😭 🏇 Debug
x 🔁 👬 👼 🖈 🕲 🕺 🛸 🕺 🕺	🗖 🗖 🗱 Variables 🕺 💁 Breakpoints 🕮	AVR32 Regist 62 Exp	ressions 🐗 Trace Data 👭 Registers 🛋 M
no_1 [AVR32 Application]			約 📲 📄 🔮
avr32gdbserver debugger (11.03.08 16:27) (Suspended)	Name	1.4	Value
1 Thread [0] (Suspended)	🗄 🥭 spiOptions	4	}
☐	E 🥏 DIP204_SPI_GPIO_MAP	0	x8000b218
avr32gdbpr0xy avr32-gdb (11.03.08 16:27)			
C:\Avr325tudio2 work\Demo 1\Debuq\DIP204 EXAMPLE.elf (11.03.08 16:27)			
	<		
example.c 🛛			🗄 Outline 🕴 🛛 🖓 💊
wrief main function : do init and loop (poll if configured so)			board.h
rier main function : do init and foop (poil if configured so)		-	compiler.h
ain (void)			dip204.h
tic const gpio_map_t DIP204_SPI_GPIO_MAP =			gpio.h
			🔜 pm.h
DIP204_SPI_SCK_PIN, DIP204_SPI_SCK_FUNCTION }, // SPI Clock.			cycle_counter.h
DIP204_SPI_MISO_PIN, DIP204_SPI_MISO_FUNCTION}, // MISO.			spi.h
DIP204_SPI_MOSI_PIN, DIP204_SPI_MOSI_FUNCTION), // MOSI.	-		avr32/io.h
DIP204_SPI_NPCS_PIN, DIP204_SPI_NPCS_FUNCTION) // Chip Select NPC:	3.		GPIO_CHARSET GPIO BACKLIGHT MINUS
			# GPIO_BACKLIGHT_MINUS
Switch the CPU main clock to oscillator O			 display : unsigned short
switch to osc0(&AVR32 PN, FOSCO, OSCO STARTUP);			 TimeOut : unsigned short
			current_char : unsigned short
Disable all interrupts.			S compare_irg_handler(void) : void
able_global_interrupt();			dip204_example_PB_int_handler(void)
		~	• S dip204_example_Joy_int_handler(void
and a second		>	
🛛 🖉 Tasks 🖹 Problems 🔋 Memory			🔳 🗙 💥 🕞 🚮 🖨 🖉 🚽 🗳 -





6.5.1 Add start and stop trace-points

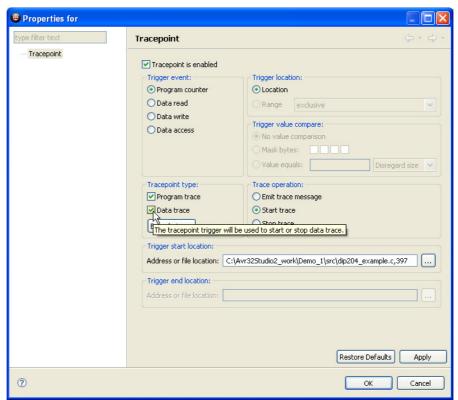
Figure 6-31. Source code editor



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:

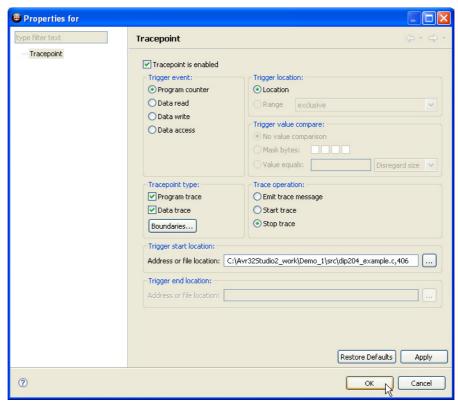
- Set Trigger Event to Program Counter
- Set Trace Operation to Start Trace
- Set Tracepoint type to both Program trace and Data trace
- Click OK

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)



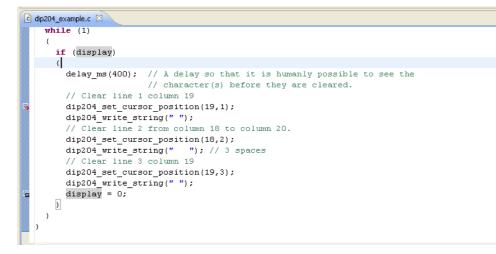
Set Tracepoint Configuration values:

- 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)

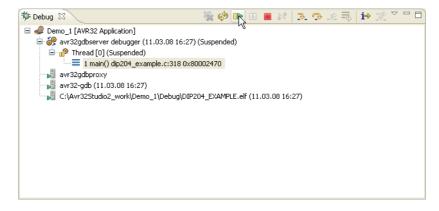
(×)= Variables	©⊚ Breakpoints	🕮 AVR32 Regist	ିଙ୍କୁ Expressio	ns 📣 Trace D	ata X	lill Re	gisters 🚬 M	lodules 🗖 🖞
Demo_1					43	宫 🛛 🔁	of 🤣 🛛 🖉	2 🔎 🖶
Frame a	#	Address	Frame desc	/* Select	one	or more	program	trace f
<			>					E
				<				>

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 TREATMENT OF THE TREATMENT OF THE	ŀ
	EVK1100	
	AVR32 UC3	
	AT32UC3A Series	
1 .		

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

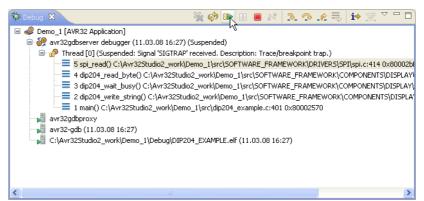


Figure 6-39. Trace data view (not empty)

(×)= y	/ariables 💊 Breakpoint	s 💷 AVR32 Register	's 🖓 Expressions 📣 Trace Data 🛛	1010 F	Registers 🛋 Modules 🛛 🛃 🛃 🚼 🛃 🥔 🖉 🖉 🗐 🛨 🗖
Demo	0_1				
	Frame #	Address	Frame description	^	Direct call into src/SOFTWARE_FRAMEWORK/COMPONENTS
	0x00000000		Watchpoint hit		- {
	0x00000001	0x80002554	Trace started		<pre>spi_selectChip(DIP204_SPI, DIP204_SPI_NPCS);</pre>
2	0x00000002	0x80002554	main()		
8	0x00000003	0x80003958	dip204_set_cursor_posi		
1	0x00000004	0x80003a50	dip204_select()		
1	0x00000005 V	∛ 0x800027fa	spi_selectChip()		
1	0x00000006	0x80002872	spi_selectChip()		
Ð	0x00000007	0x80002888	spi_selectChip()		
Ð	0x0000008	0x80003a60	dip204_select()		
2	0x00000009	0x80003978	dip204_set_cursor_posi		
1	0x0000000a	0x80003a80	dip204_write_byte()		
2	d0000000x0	0x80003ab6	dip204_write_byte()	~	×
2 345	i trace frames				

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

Figure 6-40. Changing trace view format

(×)= \	/ariables 💊 Breakpoints	💷 AVR32 Register	s 📣 Trace Data 🛛 👫 🕅 Registers 🔳	Mod	1odules) 🛃 🛃 🛃 🛃 🛃 🛃 🛃
Demo	0_1				Set Format to Disassembly
	Frame #	Address	Frame description	^	Direct call into src/SOFTWARE_FRAMEWORK/COMPONENTS
	0x00000000		Watchpoint hit		- (
	0x00000001	0x80002554	Trace started		<pre>spi_selectChip(DIP204_SPI, DIP204_SPI_NPCS);</pre>
2	0x00000002	0x80002554	main()		
8	0x00000003	0x80003958	dip204_set_cursor_posi		
<u>۽</u>	0x00000004	0x80003a50	dip204 select()		
1	0x00000005	0x800027fa	spi_selectChip()		
1	0x00000006	0x80002872	spi_selectChip()		
Ð	0x00000007	0x80002888	spi_selectChip()		
Ð	0x0000008	0x80003a60	dip204_select()		
8	0x00000009	0x80003978	dip204_set_cursor_posi		
1	0x0000000a	0x80003a80	dip204_write_byte()		
-	d0000000x0	0x80003ab6	dip204_write_byte()	~	
1				_	

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



(×)= \	/ariables 🔍 Breakpoints	🕮 AVR32 Register	rs 🙀 Expressions 📣 Trace Data 🛛	111 R	egisters	🛋 Modules	2 20	ര് 🧬	a 🔋 🗄 •	
Demo	0_1						Source Only			
	Frame #	Address	Frame description	^	Direc	t call in	ito sr Mixed Source	and Disasse	embly DPON	ENTS 📥
	0x00000000		Watchpoint hit	-	{		Disassembly	•	~	
	0x00000001	0x80002554	Trace started		sp	i_selectC	Chip(DIP204_SPI,	DIP204	_SPI_NPCS)	;
2	0x00000002	0x80002554	main()							
2	0x00000003	0x80003958	dip204_set_cursor_posi							
1	0x00000004	0x80003a50	dip204_select()							
1	0x00000005	0x800027fa	spi_selectChip()							
1	0x00000006	0x80002872	spi_selectChip()							
Ð	0x00000007	0x80002888	<pre>spi_selectChip()</pre>							
Ð	0x0000008	0x80003a60	dip204_select()							
8	0x00000009	0x80003978	dip204_set_cursor_posi							
1	0x0000000a	0x80003a80	dip204_write_byte()							
-	d0000000x0	0x80003ab6	dip204_write_byte()	~						~
2 345	5 trace frames				<					>

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

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

(×)= \	/ariables 💊 Breakpoints	AVR32 Register	s 🙀 Expressions 📣 Trace Data 🛛	1010 R	tegisters 🛋 Modules 🛛 🛃 🚼 🛃 🛃 🛃 🛨 🗖
Demo	0_1				
	Frame #	Address	Frame description	^	Direct call into src/SOFTWARE_FRAMEWORK/COMPONENTS
	0x00000000		Watchpoint hit		{
	0x00000001 0x80002554 1		Trace started		<pre>spi_selectChip(DIP204_SPI, DIP204_SPI_NPCS);</pre>
2	0x00000002	0x80002554	main()		0x80003a50 stmSP, R7, LR
-	0x00000003	0x80003958	dip204_set_cursor_posi		0x80003a54 mov R7, SP
-	0x00000004	0x80003a50	dip204_select()		0x80003a56 mov R11, 2
1	0x00000005	0x800027fa	spi_selectChip()		0x80003a58 mov R12, -55296 0x80003a5c mcall PC[8]
1	0x00000006	0x80002872	spi_selectChip()		Oxoboosase meall Peloj
Ð	0x00000007	0x80002888	spi_selectChip()		
Ð	0x00000008	0x80003a60	dip204_select()		
1	0x00000009	0x80003978	dip204_set_cursor_posi		
1	0x0000000a	0x80003a80	dip204_write_byte()		
1	d00000000x0	0x80003ab6	dip204_write_byte()	~	\sim
2 345	5 trace frames				

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



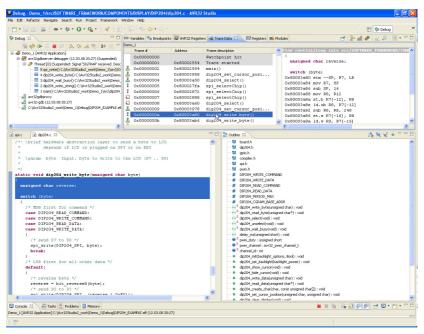
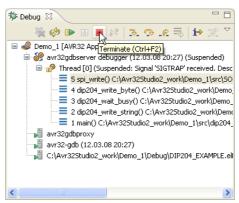


Figure 6-43. Trace frame highlighting source code in the editor

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.

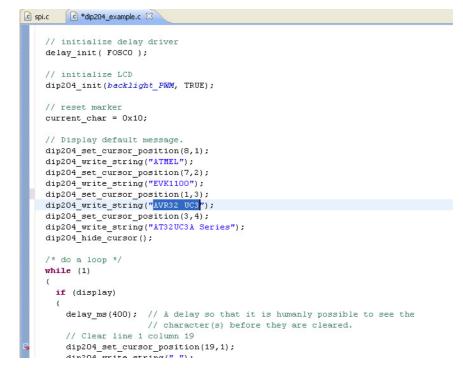
Figure 6-44. Terminating the debug session



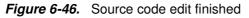


Edit the source code. This example changes the cursor position in line 342 from (6,3) to (1,3), then the text in line 343.

```
Figure 6-45. Editing source code line 343
```







ⓓ dip204_example.c ⊠
<pre>dip204_init(backlight_PWM, TRUE);</pre>
// reset marker
current_char = 0x10;
// Display default message.
dip204_set_cursor_position(8,1);
dip204_write_string("ATMEL");
dip204_set_cursor_position(7,2);
dip204_write_string("EVK1100");
dip204_set_cursor_position(1,3);
dip204_write_string("My demo is working");
dip204_set_cursor_position(3,4);
<pre>dip204_write_string("AT32UC3A Series");</pre>
dip204_hide_cursor();
/* do a loop */
while (1)
(
if (display)
{
delay_ms(400); // A delay so that it is humanly possible to see the
<pre>// character(s) before they are cleared.</pre>
// Clear line 1 column 19
<pre>dip204_set_cursor_position(19,1);</pre>
<pre>dip204_write_string(" ");</pre>
// Clear line 2 from column 18 to column 20.
<pre>dip204_set_cursor_position(18,2);</pre>
<pre>dip204_write_string(" "); // 3 spaces</pre>
// Clear line 3 column 19
are and and and and an area and and and an area and an are

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

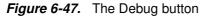
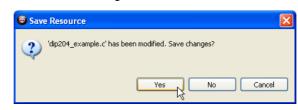




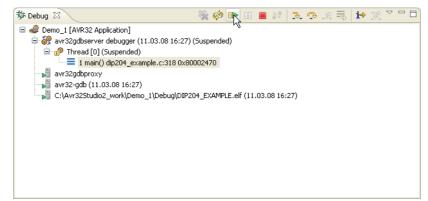
Figure 6-48. Save edited source code dialog



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

IN REPORT OF A DESCRIPTION OF A DESCRIPR
EUK1100
My Demo is working
AT32UC30 Series
HISZUCSH Series

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.

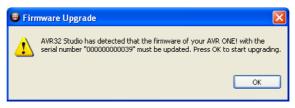
7.2 Firmware version test and upgrade

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



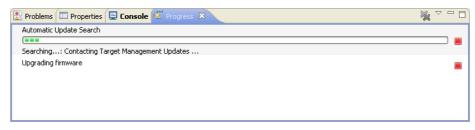
Figure 7-2. Firmware upgrade message



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

🖹 Problems 🔲 Properties	🖳 Console 🛛	🧵 Progress		🔓 🔒	2	-	- 13	
AVR32 Console								
Upgrading AVR ONE!	FPGA image	'avr32', ple	ase wait					^
Firmware Image	On disk	On tool	Status					
avrone.bin avr32.bin	1.1 1.1	0.20 0.e	UpgradeRequired UpgradeRequired					
<								>

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).

🖹 Problems 🔲 Properties 📮 Console 🙋 Progress 🕴	🍇 🗸 (- 8
Zzz Upgrading firmware (Sleeping)		





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

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDI-TIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDEN-TAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others, are the registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Windows® and others are registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be trademarks of others.

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