AVR32736: AVR32 AP7 Getting Started

Features

- **Links to documentation and resources**
- **Available tools for a AVR®32 development**
- **Example setup for a standalone development**
- **Example setups for Linux® development**

1 Introduction

This document shall help you to get started with a development for the AVR32 AP7 application processor series. It introduces available software and hardware tools, lists references to documentation and provides development setup examples.

32-bit AVR® Microcontrollers

Application Note

Rev. 32092A-AVR32-04/08

2 Documentation and resources

2.1 Atmel documentation

Before starting with AVR32 development, the user should become familiar with the AVR32 architecture. Therefore reading the architecture document should be considered first. It is available from www.atmel.com/products/avr32 in the “Other Documents” section.

The next important document is the device datasheet. Here you will find all functional descriptions of the device itself. This is also available form the above mentioned link in the “Datasheet” section.

In addition application notes are available from the Atmel homepage. These documents describe the usage of on-chip modules, give design guidelines and help the user to make its first steps in AVR32 development. Application notes also often provide code examples. So when you start working on new AVR32 software a look there first because it might speed up your development.

2.2 AVR32 Linux kernel and U-Boot

The Linux kernel from www.kernel.org is patched for the Atmel AVR32 boards in the Buildroot build system. So if you use Buildroot in your development you do not have to care about patching and building the kernel. But if you want to use the latest AVR32 kernel release or new unstable drivers you can find these on the official web site for the AVR32 Linux kernel port www.avr32linux.org. Here you can find also a lot of additional resources such as:

- **HowTo**: Information about how to do specific tasks. There's also a “Getting Started” guide for new users.
- **Linux Kernel**: Linux kernel support for the AVR32 architecture and the Atmel AT32AP chips, including peripheral drivers.
- **Das U-Boot**: AVR32 architecture and AT32AP chip support for the “Das U-Boot” bootloader.
- **MicroClibc**: AVR32 architecture support for uClibc -- a small C library for embedded Linux systems.
- **Development Tools**: Developing the development infrastructure for AVR32.
- **Linux Applications**: Various applications for use with AVR32-based Linux systems.
- **Mailing Lists**: Where to send patches and discuss technical issues.

The most up-to-date AVR32 arch code can always be found in the 'avr32-arch' branch of the git repository at kernel.org: git://git.kernel.org/pub/scm/linux/kernel/git/hskinnemoen/avr32-2.6.git

The latest AVR32 U-Boot port can be found in following git repository:

git://www.denx.de/git/u-boot-avr32.git

If you are unfamiliar with git take a look at the official documentation at:

www.kernel.org/pub/software/scm/git/docs/

2.3 AVR freaks community site

The “AVR freaks” site www.avrfreaks.net is dedicated to all AVR products. This site has a forum where AVR specific questions are discussed and all sorts of questions are answered by the members. This is the place for all AVR developers where you can get in touch with others and to exchange knowledge.

A broad variety of documentation about AVR devices is available in the Wiki section on AVR freaks. It includes getting started guides, example configurations and many other things. As member you are also able to edit and add new stuff.

If you intend to start a new AVR development that you want to share with others it is possible to register the project on AVR freaks.

In order to get information about tools from other vendors, consultants in your region or the closest AVR distributor, a section exists that lists this information in a well arranged manner. Also a news section is available that provides information about new products, tools and AVR projects.

2.4 Technical support

Atmel provides an outstanding technical support for its products. The support center keeps track of your requests in order to make any progress visible to the user. Among other advantages of the support center a user may profit especially from the following features:

- Submit and edit your support requests.
- View all your submitted requests.
- Search in the FAQ section.
- Subscribe to receive email notifications regarding new products and document updates.

In order to contact the Atmel technical support either go to the respective product category on the Atmel homepage (www.atmel.com/products/avr32/ for the AVR32 products) and select the “Support Center” from the menu or go directly to www.support.atmel.no regarding all questions about the AVR32.

2.5 AVR TV

The AVR TV web site (www.avrtv.com) gives you the opportunity to watch AVR news as a PODcast. These PODcasts are produced by the Atmel’s AVR design center in Trondheim, Norway.

AVR TV will publish three main types of issues at an irregular rate:

1. In-depth: A posting category for the “in-depth” PODcast episodes. These are mainly product demos, or product trainings and tutorials.
2. Regular Issues: A posting category for the “regular” or “ordinary” PODcast episodes. These are newsflash episodes that contain product introductions and updates, and summaries from shows and other activities.
3. Special Issues: A posting category for the “special” PODcast episodes. These episodes mainly present interesting applications that feature AVR products, and customer success stories.

3 Tools

3.1 Software

3.1.1 AVR32 Studio

AVR32 Studio is an integrated development environment (IDE) for developing AVR32 applications. AVR32 Studio provides a complete set of features including project management and version control integration (CVS); a C/C++ editor with syntax highlighting and code completion; a debugger supporting run control including source and instruction-level stepping and breakpoints; registers, memory and I/O views; and target configuration and management. AVR32 Studio is built on Eclipse, enabling easy integration with third party plugins for increased functionality.

AVR32 Studio supports all of Atmel's AVR32 32-bit processors. AVR32 Studio supports development and debugging of both standalone (without an operating system) applications and Linux applications (for the AT32AP7 device family).

AVR32 Studio integrates with the GNU Toolchain for AVR32. The GNU C Compiler (GCC) is used for compiling C/C++ programs, while the GNU debugger (GDB) is used for debugging the application on target. Atmel's AVR32 utilities, avr32program and avr32gdbproxy, are used for deployment and debugging of standalone applications as well as target voltage and clock generator adjustments.

AVR32 Studio is free software and you can get hold of it from the Atmel web site (www.atmel.com/products/avr32).

3.1.2 IAR® Embedded Workbench

The IAR Embedded Workbench is, like the AVR32 Studio, a complete integrated development environment. IAR Embedded Workbench provides a suite of AVR32 development tools and offers a continuous workflow, efficient code generation and ease of use. It includes among other things:

- Integrated development environment with project management tools and editor
- C/C++ compiler
- Assembler and linker
- Run-time libraries
- AVR32 JTAGICE mkII debugger support

The IAR compiler is optimized to recognize patterns in the C-code that can use SIMD and DSP instructions, thus further increasing the ease of use and performance when running compiled C-code applications.

The IAR Embedded Workbench can not build or debug Linux applications. If you intend to develop Linux software for the AVR32 you can use the AVR32 Studio with the GNU toolchain.

For more information about the IAR Embedded Workbench visit www.iar.com.

3.1.3 GNU Toolchain

AVR32 GNU toolchain is a set of standalone command line programs used to create applications for AVR32 microcontrollers. The applications run either as embedded applications or on top of an embedded operating system, e.g. Linux. The AVR32 GNU toolchain is developed maintained and supported by Atmel and is free software.

Several of the tools are based on tools from GNU (www.gnu.org), and some are developed by Atmel. The toolchain comprises following features:

- Runs on Microsoft® Windows® and Linux platforms
- Cross compiler for AVR32 devices
- Assembler and linker for AVR32 devices
- Debugger for AVR32 devices
- Flash programming tools for AVR32 devices
- C-libraries for development of C/C++ programs

This toolchain is used by AVR32 Studio to compile, link, deploy and debug software on the AVR32. These tools can also be used directly from within a shell.

The toolchain for Windows is available from www.atmel.com/products/avr32 as an installer. It is also possible to install the toolchain together with AVR32 Studio included in one installer.

Developers working on a Linux host can get the toolchain from a repository. AVR32 Studio is distributed separately form the toolchain on Linux. Currently are packages for following distributions available:

3.2 Hardware

3.2.1 Emulators

3.2.1.1 Atmel JTAGICE mkII

The Atmel JTAGICE mkII emulator supports AVR32 and can be used together with all Atmel AVR32 development boards either with the GCC or the IAR compiler. The JTAGICE mkII supports basic runtime control and a limited trace capability through the JTAG interface. This emulator is supported in both the AVR32 Studio and the IAR Embedded Workbench.

Figure 3-1 JTAGICE mkII

3.2.1.2 Ashling PathFinder, Vitra and Opella

The AP7 series is also supported by the PathFinder, Vitra and the Opella products from Ashling (www.ashling.com). These tools provide highend debugging capabilities such as sustained trace and SQA (software quality assurance).

- **PathFinder**: Source-level debugger
- **Vitra**: USB based emulator
- **Opella**: Network emulator with real-time trigger and trace

The Ashling emulators are not supported by the AVR32 Studio!

3.2.2 Development boards

3.2.2.1 ATSTK1000

The Atmel STK®1000 board provides a complete AT32AP7000 development environment. The kit has among other peripherals two Ethernet ports, a high quality QVGA LCD, a loudspeaker, and connectors for USART, PS/2, VGA, and USB. An expansion header can be used for prototyping. The STK1000 is a base board that can be used with different top modules containing the application processor and additional features.

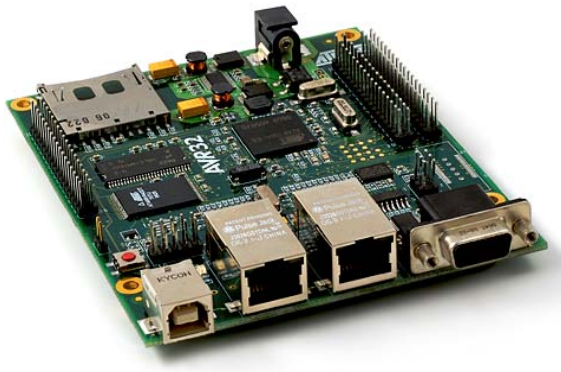
Figure 3-2 STK1000 mounted with STK1002 top-module.

A pre-installed Linux image on the enclosed SD card ensures that the user can boot Linux and start program development directly after power up.

3.2.2.2 ATNGW100

The ATNGW100 board uses the AT32AP7000 which combines Atmel's state of the art AVR32 Digital Signal Processor CPU with two Ethernet ports, SD and MMC card reader, connectors for USB, RS232 and JTAG. All other peripherals like SPI, LCD Controller, Image Sensor Interface and so forth are available on the extension headers. This makes the NGW100 to an ideal development board for the AT32AP7000.

Figure 3-3 ATNGW100 Network Gateway Reference Design



The board is preloaded with Linux and runs by default a lot of services such as a DHCP server, FTP server, Web server … These services combined with the two Ethernet ports make this board to a network gateway.

4 Standalone development

4.1 Software Framework

Atmel provides a Software Framework for the AVR32AP7 series. This framework includes drivers for all on-chip modules and examples on how to use them.

4.2 Development setup

For a standalone development (no operating system on the target) the setup in Figure 4-1 is the most common version. The target board is connected to a JTAG programmer/debugger which is connected to the host PC. On the host PC runs an Integrated Development Environment (IDE) such as the AVR32 Studio or the IAR Embedded Workbench. These two IDEs support the JTAGICE mkII. The IAR IDE supports also other JTAG devices. Take a look at the IAR documentation on www.iar.com for an overview of all supported devices.

It is also possible to program and debug the target board through the JTAGICE mkII on a

Figure 4-1 Standalone development setup

5 Linux development

5.1 Resources for Linux development:

5.1.1 AVR32 Buildroot

Buildroot is a set of scripts and menu system which builds an entire root file system for a given target. A target can be ATNGW100, ATSTK1000 or any other Atmel AVR32 board that supports Linux. Buildroot snapshots as compressed archives are available from www.atmel.no/buildroot. Available documentation is also listed on the referenced link.

The scripts are based on a combination of Makefile and kconfig which is commonly used in many projects. Kconfig is used to give the user an easy configuration interface that is stored in a file. The Makefile system then reads out the values stored by kconfig and configures a set of rules in which different software is compiled.

Buildroot starts by compiling the toolchain if requested, or it can use an external toolchain. It then moves over to the Linux kernel, software libraries and applications. Finally it combines all the applications with the needed libraries and kernel to a file system image. This image is ready for a deployment to the target system. It is also possible to use the file system image on the host as a NFS root file system for the target.

5.2 Development setup

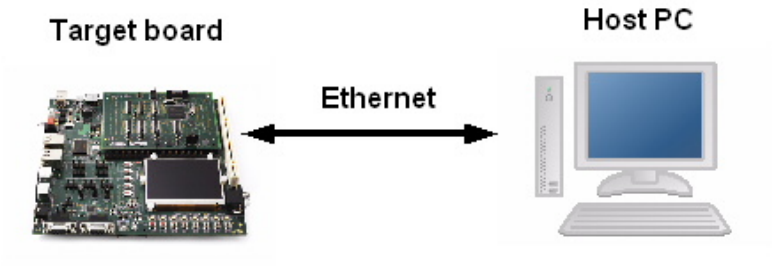
5.2.1 AVR32 Studio setup

To use AVR32 Studio as deployment tool and debugger for Linux applications the host PC needs an Ethernet connection to the target.

AVR32 Studio needs FTP and SSH services on the target to transfer an application. Before you can use these services they have to be installed and configured correctly on the target. On the NGW100 and the STK1000 boards are these services already enabled and configured properly. Take a look at the AVR32 Studio documentation, the application note “AVR32015: Getting started with AVR32 Studio” and the “AVR freaks” Wiki for further documentation.

To debug the application on the target form the host a gdb-server (GNU debugger) must be used on the target. A gdb-client on the host connects then to the gdb-server on the target and initiates a debug session. AVR32 Studio uses this approach and lets the user control the debug session through a graphical user interface.

Figure 5-1 AVR32 Studio Linux development setup



5.2.2 Network File System

File systems and kernels created by the Buildroot build system can be used by the target directly from the host. To achieve this, the host exports the file system as a network file system (NFS) and the target mounts this.

This approach has the advantage that no application has to be transferred to the target by the user. The user just copies his applications to the NFS file system on the host. This setup is commonly used in the development phase as it gives the developer an easy and fast access to the target file system for updates and fixes.

A NFS setup needs a kernel on the target system as it is only possible to mount the file system from a running kernel. Because of that an update of the kernel on the NFS has no impact on the target. Drivers may be updated on the NFS as they are loaded after the file system mounting.

5.2.3 Trivial File Transfer Protocol

The bootloader on the target board supports the Trivial File Transfer Protocol (TFTP). Therefore is it possible to transfer images from the host to the system RAM. From there the image may be written to the flash or executed in place.

This is especially useful when a kernel image must be transferred form the host to the target. With TFTP is it possible to run different kernel versions very quickly on the target.

5.2.4 SD-Card

A SD-card can be used as root file system container. The file system generated by Buildroot can be installed on a SD-card. The bootloader on the board is able to load the Linux kernel form the SD-card to RAM and start it there. Linux will then mount the root file system on the SD-card.

5.2.5 On-board flash

The journaling flash file system image created by Buildroot can be downloaded directly to the on-board flash with a programmer or with the bootloader that gets the image from a TFTP-server on the host. It is also possible to program the flash from the Linux system on the target but this need to be enabled on the kernel command line. To flash the whole file system on the target is an approach that should be used in production or upon firmware upgrades.

6 References

AVR32 section on the Atmel web site: www.atmel.com/products/avr32

Official Linux kernel developers site: www.kernel.org

AVR32 Linux kernel git repository branch on kernel.org:

www.git.kernel.org/pub/scm/linux/kernel/git/hskinnemoen/avr32-2.6.git

AVR32 Linux kernel port: www.avr32linux.org

AVR32 U-Boot git repository branch: git://www.denx.de/git/u-boot-avr32.git

Git documentation: www.kernel.org/pub/software/scm/git/docs/

Buildroot build system for AVR32: www.atmel.no/buildroot

AVR community web site: www.avrfreaks.net

Atmel AVR support: www.support.atmel.no

AVR TV: www.avrtv.com

IAR AVR32 products: www.iar.com

Ashling AVR32 products: www.ashling.com

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

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature