AVR32003: AVR32 AP7 Linux Buildroot

32-bit AVR® Microcontrollers

Application Note

Features

- Integrated build scripts for:
  - Toolchain
  - Linux® kernel
  - Boot loader
  - Libraries
  - Applications
- Configurable and optimized.
- Easy adoptable to custom boards.
- Simple steps to integrate custom libraries and applications.
- Generates a fully working file system ready for deployment.

1 Introduction

Buildroot is a set of scripts that builds an entire root file system for a given target. A target can be ATNGW100 or ATSTK1000.

The scripts are based on a combination of Makefile and kconfig that 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 start 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 the user to deploy on his target.

Rev. 32062E-AVR32-11/08

2 Requirements

Buildroot is a script system that heavily depends on a Linux system. It is highly recommended that users either run a native Linux installation or run Linux within a virtual machine when running other operating systems.

Buildroot is supported by most host architecture and requires quite a lot of disk space. Having 5 GB free disk space before starting to work on you root file system is in general a good idea.

The build system also needs a set of host tools preinstalled on the build machine. Most Linux distributions allow the user to install a package covering the build essentials.

List of requirements for the build machine:

- C compiler (GCC)
- C++ compiler (for Qtopia® (G++))
- GNU make
- sed
- flex
- bison
- patch
- gettext
- libtool
- texinfo
- autoconf (version 2.13 and 2.61)
- automake
- ncurses library (development install)
- zlib library (development install)
- libacl library (development install)
- lzo2 library (development install)

2.1 Using a proxy when downloading

Some networks are connected to the Internet through a proxy. Buildroot has no means to detect your proxy settings, but you can tell your shell how your HTTP and FTP proxy works. Buildroot will then use this information when downloading the source code for various applications from the Internet.

To set HTTP proxy type the following in the shell you later will use for Buildroot:

```
export http_proxy=”http://<username>:<password>@<proxy URL>:<port>/
```

Example:

```
export http_proxy=”http://avr32:password@proxy.example.net:3128/
```

To set FTP proxy type the following in the shell you later will use for Buildroot:

```
export ftp_proxy=”ftp://<username>:<password>@<proxy URL>:<port>/
```

Example:

```
export ftp_proxy=”ftp://avr32:password@proxy.example.net:3128/
```

3 Getting started for AVR32 targets

This short getting started guide is intended for ATNGW100 and ATSTK1000 users.

Start off by downloading the latest release from Atmel®, extract it somewhere on you system and enter the buildroot-avr32-<version> directory.

To load the default configuration for these boards, simply type on of the following depending on your board:

```
make atstk1002_defconfig
make atstk1005_defconfig
make atstk1006_defconfig
make atngw100_defconfig
make atngw100-base_defconfig
make evklcd100_defconfig
make evklcd101_defconfig
```

Buildroot will now load the board’s configuration and save it in a file called *.config*.

An optional step is to download all the source files before starting the build. This can be initiated by typing:

```
make source
```

The next step for the user is to start the build process. This is done by typing:

```
make
```

Buildroot will now start to download software packages from the Internet, extract them on your local file system and compile them for AVR®32.

When the build process is finished and successful, you will find the created root file systems in the binaries/<board name>/ directory. There can be various types of ready binaries depending on what is chosen by the configuration system.

4 Directory structure

Buildroot has a well defined structure of directories. From a fresh extraction of the tarball, Buildroot will look like:

- *Config.in*
- *.defconfig*
- *docs/*
- *Makefile*
- *package/*
- *project/*
- *target/*
- *TODO*
- *toolchain/*

After a successful build there will be 6 extra directories in the base directory of Buildroot:

- *binaries/*
- *build_avr32/*
- *dl/*
- *include/*
- *project_build_avr32/*
- *toolchain_build_avr32/*

4.1.1 binaries/

This directory contains the successfully built images. Populated on the form binaries/<project name>/.

4.1.2 build_avr32/

Directory used for building each library and application. Also, by default, holds the staging_dir/ directory which contain the toolchain and libraries.

4.1.3 .config

File which contains the system configuration, this will appear after the first time running menuconfig or loading a default configuration.

4.1.4 Config.in

The top level configurations file for the kconfig system.

4.1.5 .defconfig

File with a default general configuration for Buildroot, mainly for x86 architecture.

4.1.6 dl/

This directory will contain all the downloaded source tar archives so the user only has to download libraries and applications once.

4.1.7 docs/

In this directory the documentation for Buildroot can be found.

4.1.8 include/

Directory made by kconfig, used for knowing which values are stored during configuration setup. Leave it as is, everything is auto generated.

4.1.9 Makefile

Top level makefile for Buildroot describing the initial rules for how to build a root file system. This file will again include all the other makefiles spread out in the sub directories.

4.1.10 package/

Directory containing all the makefiles that describes how each library and application shall be built.

4.1.11 project/

Directory containing information about project building that allows Buildroot to be able to have multiple projects ongoing in the same extracted source tree. Buildroot will share compiled applications, libraries and toolchain whenever possible.

4.1.12 project_build_avr32/

The project specific part of the build is located in this directory. Is populated on the form project_build_avr32/<project name>/. Here you will find the target specific Linux kernel; Busybox and uncompressed root file system.

4.1.13 target/

Directory containing information about targets that is where the different boards are defined, board specific patches to the Linux kernel and board specific files. Typically this is where the base for the file system is located, called target_skeleton.

4.1.14 TODO

TODO list for upstream Buildroot describing what areas is in the works. Feel free to work on TODO material if you are confident with how Buildroot works.

4.1.15 toolchain/

Directory containing makefiles for the toolchain that describes how it should be built and what options a user have. The toolchain is a vital part of buildroot, since you are not capable to build anything without a working toolchain. Given that the user has chosen an internal toolchain.

4.1.16 toolchain_build_avr32/

Directory containing the extracted toolchain source tarballs and where Buildroot builds the toolchain.

5 Flexibility and configuration

Buildroot is also highly flexible. For starter we recommend following the getting started guide, but you can also configure which applications to put into your image.

Enter the base directory of Buildroot and type:

- *make menuconfig*

A curses based menu system will guide you around the different choices.

If a user wants to use the atngw100 or atstk1002 default configuration as a base, but do minor adjustment the following short guide will configure Buildroot to build it as a new project.

Start off in the base directory for Buildroot and type:

- *make atstk1002_defconfig* or what is appropriate for the target board.
- make menuconfig
    - Project Options --->
      Project name, change this to something descriptive.

Exit the menu system and save the configuration, then type:

- *make*

Your file system will now be located in binaries/<project name>/ directory.

6 Deploy binaries to target system

Buildroot creates binaries ready to download onto the target, using a flash programmer like avr32program or a SD-card. In the *binaries/* directory, in the base directory of Buildroot, the user can find the following files depending on the configuration system:

- <board base name>-linux-<version>.gz
- rootfs.avr32.ext2
- rootfs.avr32.ext2.bz2
- rootfs.avr32.jffs2
- rootfs.avr32.jffs2-<partition_name>
- rootfs.avr32.tar
- rootfs.avr32.tar.bz2
- u-boot.bin

6.1.1 <board base name>-linux-<version>.gz

This is the kernel targeted for your board. It is also included in the root file system images so the user does not need to explicit download this into flash.

6.1.2 rootfs.avr32.ext2

This is the EXT2 version of the root file system. It is targeted for SD/MMC-card boot, or other media that can hold an EXT2 file system. The .bz2 version of this file is a bzip2-compressed version of the same file system.

6.1.3 rootfs.avr32.jffs2

The JFFS2 root file system is target for the onboard flash device. Onboard flash is typically NOR, NAND or DataFlash®.

6.1.4 rootfs.avr32.jffs2-<partition_name>

Buildroot can make multiple JFFS2 images when the target device has more than one flash device or partition. In this case an extension is added to each image file indicating the target flash device or partition.

For example a build for the ATNGW100 can generate three images:

- rootfs.avr32.jffs2
- rootfs.avr32.jffs2-root
- rootfs.avr32.jffs2-usr

Where the first file without an extension is only a file touched by the make system and should be ignored. The other two files are for programming into the two flash devices on the ATNGW100. They are named according to their content, where the root image is “/” and the usr image is “/usr”.

6.1.5 rootfs.avr32.tar

A full file system is available in a tar archive. This is useful if the user wants to have the root file system mounted over network or extract the file system to a removable medium like SD/MMC-card.

6.1.6 u-boot.bin

U-boot is the boot loader made for the target board; this must be programmed with a flash programming tool like avr32program.

7 Download

Buildroot for AVR32 can be downloaded from http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4401.

8 Further reading

For adding user specific packages or custom target board to Buildroot, see the on-line documentation and application notes on http://www.atmel.com/AVR32.

General online documentation provided by Atmel about Buildroot is available on http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4401.

For upstream documentation visit http://buildroot.uclibc.org/buildroot.html, an external site not in affection with Atmel.

Headquarters

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

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
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