RZ/A1

Getting Started with Buildroot for RZ/A1

EU_00170
Rev.1.00
Aug 26, 2014

Introduction

Target Device

Contents

1. What is Buildroot

Directly from http://buildroot.uclibc.org/about.html

Buildroot is a set of Makefiles and patches that makes it easy to generate a complete embedded Linux system. Buildroot can generate any or all of a cross-compilation toolchain, a root filesystem, a kernel image and a bootloader image. Buildroot is useful mainly for people working with small or embedded systems, using various CPU architectures (x86, ARM, MIPS, PowerPC, etc.) : it automates the building process of your embedded system and eases the cross-compilation process.

While there is lots of documentation on the Buildroot website, the purpose of this document is give you a step by step setup of a build environment specifically for the RX/A1 and how to utilize it. From there, you can explore and add the many software packages that Buildroot supports.

2. Step 1: Downloading Buildroot

While this document was written using the latest stable version of Buildroot (2014.05), it will show you how to configure Buildroot from scratch such that you can always update to the latest version if you desire.

Simply navigate to the Download page (from http://buildroot.uclibc.org) and grab either the latest version, or version 2014.05 to be safe. You could also use the direct link and get it on the command line in a Linux terminal:

```
$ wget http://buildroot.uclibc.org/downloads/buildroot-2014.05.tar.bz2
```

To exact the tar.bz2 file, use this command line in Linux:

```
$ tar -xjf buildroot-2014.05.tar.bz2
```

For the tar.gz version, use this command line:

```
$ tar -xzf buildroot-2014.05.tar.bz2
```

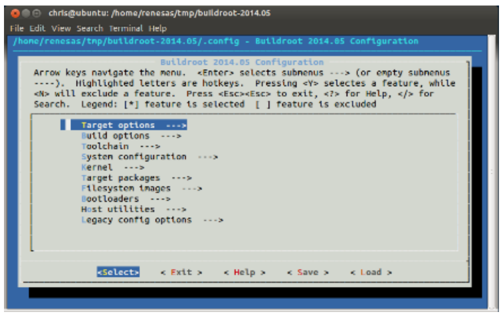
3. Step 2: Configuration

Now it is time to start configuring Buildroot. This is done in a simplistic graphical method. Change into the directory were you extracted Buildroot to and type to the follow:

```
$ make menuconfig
```

Next, you should see something that looks like the screenshot on the right.

To navigate, use the arrow keys and press Enter to go into a sub menu (--->), and ESC+ESC to back out of a menu.



```
chris@ubuntu: /home/renesas/tmp/buildroot-2014.05
File Edit View Search Terminal Help
/home/renesas/tmp/buildroot-2014.05/.config - Buildroot 2014.05 Configuration
                    Buildroot 2014.05 Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
 ----).  Highlighted letters are hotkeys.  Pressing <Y> selectes a feature, while
 <N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for Help, </> for
 Search.  Legend: [*] feature is selected  [ ] feature is excluded

         Target options  --->
         Build options  --->
         Toolchain  --->
         System configuration  --->
         Kernel  --->
         Target packages  --->
         Filesystem images  --->
         Bootloaders  --->
         Host utilities  --->
         Legacy config options  --->

       <Select>    < Exit >    < Help >    < Save >    < Load >
```

3.1 Target options (Architecture Selection)

To start, use the menu system and make the following selections highlighted in red.

- Target options >> Target Architecture (ARM (little endian))
- Target options >> Target Architecture Variant (cortex-A9)
- Target options >> Target ABI (EABIhf)

NOTE: To be able to select/use pre-built toolchains by **Linaro**, you must select **EABIhf** here. To be able to select/use pre-built toolchains by **Sourcery CodeBench**, you must select **EABI** here.

3.2 Build options

The defaults are fine for just getting started, so you can skip this section.

3.3 Toolchain selection

In Buildroot, you can either download a toolchain or supply your own. Additionally, you can choose to have Buildroot go out and get all the complete source code for the toolchain (GCC, binutils, glib, uclib, etc..) and build a toolchain from scratch. You might want that for archiving purposes, or if you ever find yourself in a situation where you wish you could look inside the C library to see what the code does.

Of course, downloading and building the entire toolchain will inevitably take a while, so if this is the first time you are building a root file system, you might want to consider just downloading a prebuild package toolchain.

To build a toolchain from scratch (Buildroot toolchain), select the following as a good starting point:

- Toolchain >> Toolchain type (Buildroot toolchain)
- Toolchain >> Kernel Headers (Linux 3.14.x kernel headers)
- Toolchain >> C library (uClibc)
- Toolchain >> [*] Enable large file (files > 2 GB) support

NOTE: Some software packages might require additional toolchain build settings to be set. They will let you know this when you try to select them.

To download a pre-built toolchain (recommended), select:

- Toolchain >> Toolchain type (External toolchain)
- Toolchain >> Toolchain (Linaro 2014.02)
- Toolchain >> Toolchain origin (Toolchain to be downloaded and installed)

NOTE: If you plan to have multiple Buidroots to have different version of file system, you can save some hard disk space by keeping the toolchain in one central location, and then use the "Toolchain origin (Pre-installed toolchain)" option instead and just point the "Toolchain path" to that location.

3.4 System configuration

Use the menu system and make the following selections highlighted in red.

- System configuration >> /dev management (Dynamic using devtmpfs only)

This is means the /dev directory will be automatically populated based on what devices are found. Note that your kernel must have CONFIG_DEVTMPFS and CONFIG_DEVTMPFS_MOUNT enabled. Also, the /dev directory will actually be a mounted RAM based directory so it will get rebuilt each time you boot and not actually change anything in your physical /dev directory. While this will take a little more time during boot, it is very helpful in your development stage. For your final project where all your hardware will be known and static, you could hard code all your device node files to decrease boot time.

- System configuration >> getty options >> (ttySC2) TTY port
- System configuration >> getty options >> Baudrate (115200)

NOTE: This is the serial debug port on the RZ/A1 RSK board. Basically, this is just used in the creation of the /etc/inittab file in the target rootfs.

3.5 Target packages

By default, only Busybox is selected to be downloaded and built since it is really all you need for a minimal system. So for now, we'll just skip over this section. Later you can go back and select other software packages as needed.

3.6 Kernel

Leave as default (blank). Generally, you'll want to build the kernel someplace else since building it within Buildroot is more cumbersome given that you will not be building a vanilla kernel and will need to apply patches specific to the RZ/A1.

3.7 Filesystem images

You can skip this section for now as "[*] tar the root filesystem" should already be selected by default.
This means that after your build is complete, your file system will be packaged up in a tar file and located here:

```
buildroot-2013.05/output/images/rootfs.tar
```

It will then be ready to be extracted into the medium of your choice (eMMC, SD Card, USB Flash Drive, etc…)

3.8 Bootloaders

Leave as default (blank). Generally, you'll want to build your bootloader (u-boot) someplace else since building it within Buildroot is more cumbersome given that you will not be building a stock u-boot and will need to apply patches specific to the RZ/A1.

3.9 Host utilities

Leave as default (blank).

3.10 Legacy config options

Leave as default (blank).

4. Step 3: Building

4.1 How to Build

Once you've set up your initial configuration, all you have to do is type 'make' and Buildroot will begin the process of downloading and building everything you need, including host utilities needed to build your target file system and file system images. Therefore, you might see programs being downloaded and built that are actually being built for x86 execution, not your target ARM processor. This ensures that the same final rootfs can be built on any system regardless of what host software is (or isn't) currently installed, and can be easily duplicated on any other machine.

To build everything, simply type:

```
make
```

4.2 How to Rebuild

One important aspect of Buildroot to keep in mind is that its main function is a software package builder and file system creator. For each software package, there are multiple steps needed to create and *install* the final software. In order to keep track of what steps have already been performed, Buildroot creates 'stamps' which are basically blank files that it leave in the source code directories. You can see them in each of the various output/build/xxx/ directories. This makes sure as you add new packages, it skips over the packages that have already been completed.

It is now important to mention that adding new packages through the menuconfig interface (under the "Target packages" submenu) and then typing 'make' again works quite well. However, removing them does not. Buildroot does not have a package remove option. Basically the reason is that Buildroot simply asks the package distribution to *install* itself to the target file system but doesn't keep track of what files get copied. Therefore Buildroot has no idea how to remove those file if you no longer wish to keep that package. Therefore, to remove a package from your target system, first remove it from menuconfig, but then you must also delete the following directories:

```
buildroot-2014.05/output/build
buildroot-2014.05/output/images
buildroot-2014.05/output/target
```

Essentially, this means everything needs to be rebuild from scratch (extracted/patched/configured/built/installed). The good thing is that the software packages do not have to be re-downloaded because they were all tucked away in the 'dl' directory (ex, buildroot-2014.05/dl).

While this may seem extreme, this is the safest and recommended method as it ensures you will end up with only what you want, and more importantly it will be completely reproducible later or on another build system.

5. Step 4 - Programming your rootfs

While you can select to have multiple types of rootfs image files created, by default you should at least have a rootfs.tar located here:

```
buildroot-2013.05/output/images/rootfs.tar
```

Buildroot was designed to be run without system root permissions, but that leaves it without the ability to create device nodes (the files under /dev) in the target root file systems. However, it can make them inside a tar file or other file image container. This means when you attempt to extract your rootfs.tar file, you must have superuser privileges otherwise you will be presented with errors. For example, to properly extract a rootfs.tar to a USB Flash Drive, you would type something like:

```
sudo tar –C /media/usb_drive –xf rootfs
```

NOTE that this assumes that the target "/media/usb_drive" is a USB flash drive that is already formatted as an appropriate ext2 or ext3 file system.

When booting your new system, you can log in using the username "root". No password will be required.

6. Busybox Configuration

As you may (or may not) know, Busybox is a single executable that emulates many common utilities and has its own menuconfig interface. Therefore in Buildroot there is a special command to access that interface. On the command line, type:

```
make busybox-menuconfig
```

This will bring you into same type of graphical interface as Buildroot and from here you can select what additional utilities you would like to include support for.

Note that after leaving the graphical interface and returning to command prompt, you will see that Buildroot automatically deletes the ".stamp_built" and ".stamp_target_installed" files in the busybox build directory. This means that the next time you type 'make' for Buildroot, it will know that it needs to rebuild busybox (adding in support for your new utilities selections) and also install them into your target rootfs. Of course while this works good for adding new support, removing support for a utilities will result in the busybox application not supporting that feature, but an application link still existing in /bin to your utility that you've delete, so you have to be careful of that. Refer to the section on Rebuilding to learn how to completely rebuild your rootfs in the case that you have removed a feature.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Aug 26, 2014	—	First edition issued