Application Note 221

RealView Development Suite 4.0 ARM Compiler for Scratchbox

Document number: ARM DAI 0221B
Issued: December 2009

ARM

Application Note 221
RealView Development Suite 4.0 – ARM Compiler for Scratchbox

Release information

The following changes have been made to this Application Note.

Change history

Date	Issue	Change
February 2009	A	First release
November 2009	B	Second release. -Supports wrapping with any CodeSourcery glibc installed in Scratchbox

Proprietary notice

Confidentiality status

This document is Open Access. This document has no restriction on distribution.

Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

ARM web address

http://www.arm.com

Table of Contents

1 Introduction

1.1 Support Status

The information provided here is given for your reference only. Limited support can be provided for the instructions in this document to customers with a valid RealView tools support contract with ARM Limited. However, it is suggested that in the first instance your issue is discussed in one of the various public forums, such as http://www.arm.com/support/newsgroup/ and the ARM website forums.

Please note that ARM does not provide support on the use of Scratchbox, GNU/Linux Tools and Applications.

1.2 Document Scope

This document provides information on using the ARM Compiler 4.0 (RealView CompilationTools - RVCT 4.0) in Scratchbox v. 1.0.12. This can be done by using the ARM compiler 4.0 plugin builder, distributed by ARM separately for use with the *RealView Development Suite* (RVDS), to generate the plugin. The plugin was tested with Scratchbox versions 1.0.11 and 1.0.12.

Note *Scratchbox version 2.0 is not supported.*

1.3 Purpose

This Application Note describes how to use the ARM compiler to build GNU/Linux applications in Scratchbox, on a Linux host machine. Scratchbox is an OpenSource framework, used to cross-compile applications for Embedded Linux. Note that the procedure described in this document was tested *against CodeSourcery releases 2007q1-21, 2007q3-51* and *2009q1-203* but it is not guaranteed to work with other toolchains.

1.4 Requirements

The minimum requirements are:

- ARM compiler 4.0, available with RVDS 4.0 Standard and Professional.
- A host machine running a GNU/Linux distribution. Ubuntu 8.04 was used for the purposes of this document.
- Scratchbox installed (v 1.0.11 or 1.0.12). For installation and setup, see *Set up Scratchbox* on paragraph 2 page 5.
- CodeSourcery toolchain installed in Scratchbox.
- The armcc_pluginbuilder-v1.5.zip file that contains the pluginbuilder.pl script and pluginholder tarball. It is available for download from http://silver.arm.com for RVDS 4.0 and upwards licensees.

2 Setup Scratchbox

This section describes how to install Scratchbox, configure and setup a Scratchbox target, and build an example application using the CodeSourcery GCC 2007q3-51 toolchain that is available as a plugin for Scratchbox. ARM Compiler uses the glibc part of the CodeSourcery toolchain, that is already installed, to build the ARM Linux binaries.

Note *This section may be skipped if you have Scratchbox already installed and working on your Linux host machine. However you do need to have a CodeSourcery toolchain installed in Scratchbox.*

2.1 Scratchbox installation

To install Scratchbox, run the wget and tar Linux tools. Usually these are installed by default in most of the Linux distributions.The `wget` tool gets the denoted tarball from the web, and the `tar` tool uncompresses and unpacks the tarball

Note *You must be root to install Scratchbox. This is essential because the Scratchbox installation contains device nodes creation, which can only be performed with root privileges.*

The packages that you need are:

- scratchbox-core
- scratchbox-libs
- scratchbox-toolchain-host-gcc
- scratchbox-devkit-doctools
- scratchbox-devkit-git
- scratchbox-devkit-cputransp
- scratchbox-devkit-perl
- scratchbox-devkit-mtd
- scratchbox-toolchain-arm-linux-cs2007q3-51sb3

Download and decompress them with the following procedure. Scratchbox is installed in the root folder '/' because this is the recommended location for Scratchbox.

```
wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-core-1.0.12-i386.tar.gz
tar -C / -xzf scratchbox-core-1.0.12-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-libs-1.0.12-i386.tar.gz
tar -C / -xzf scratchbox-libs-1.0.12-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-toolchain-host-gcc-1.0.12-i386.tar.gz
tar xzf scratchbox-toolchain-host-gcc-1.0.12-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-devkit-doctools-1.0.10-i386.tar.gz
tar -C / -xzf scratchbox-devkit-doctools-1.0.10-i386.tar.gz

wget \
```

```
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-devkit-git-1.0.1-i386.tar.gz
tar -C / -xzf scratchbox-devkit-git-1.0.1-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-devkit-cputransp-1.0.8-i386.tar.gz
tar -C / -xzf scratchbox-devkit-cputransp-1.0.8-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-devkit-perl-1.0.4-i386.tar.gz
tar -C / -xzf scratchbox-devkit-perl-1.0.4-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-devkit-mtd-1.0-i386.tar.gz
tar -C / -xzf scratchbox-devkit-mtd-1.0-i386.tar.gz

wget \
http://scratchbox.org/download/files/sbox-
releases/apophis/tarball/scratchbox-toolchain-arm-linux-cs2007q3-51sb3-
1.0.9-1-i386.tar.gz
tar -C / -xzf scratchbox-toolchain-arm-linux-cs2007q3-51sb3-1.0.9-1-
i386.tar.gz
```

Once this step is completed, you should be able to see something such as the following in the default Scratchbox installation folder (/).

```
# ls -l /scratchbox
total 64
drwxr-xr-x 8  root root 4096 Feb 16 16:43 compilers
drwxr-xr-x 11 root root 4096 Nov 27 20:00 dev
drwxr-xr-x 7  root root 4096 Jan  9 18:44 device_tools
drwxr-xr-x 7  root root 4096 Nov 27 19:57 devkits
drwxr-xr-x 2  root root 4096 Sep 26 13:24 doc
drwxr-xr-x 5  root root 4096 Nov 27 20:00 etc
drwxr-xr-x 4  root root 4096 Sep 26 13:22 host_shared
-rwxr-xr-- 1  root sbox 7165 Sep 26 13:22 login
drwxr-xr-x 3  root root 4096 Sep 26 15:07 packages
-rw-r--r-- 1  root root   42 Sep 26 13:22 README
-rwxr-xr-x 1  root root 1414 Sep 26 13:22 run_me_first.sh
drwxr-xr-x 2  root root 4096 Sep 26 13:48 sbin
drwxr-xr-x 12 root root 4096 Nov 27 19:57 tools
drwxr-xr-x 3  root root 4096 Nov 27 20:02 users
```

Please note that the `login` executable belongs to group `sbox`.

Then as root run the following script and accept the default answers to all the prompts:

```
./scratchbox/run_me_first.sh
```

Note *For any problems running this file, see Appendix - FAQs on page 13.*

Scratchbox is now installed on your GNU/Linux box. The next step is to add a user to use it. This can be done with the next command, which should also be run as root:

```
./scratchbox/sbin/sbox_adduser <user>
```

where `<user>` is the name of an existing user in the Linux box.

Note *Scratchbox cannot be operated by root, therefore you are unable to add root as a Scratchbox user.*

Note *If the user added to Scratchbox users group (sbox) is already logged on the Linux box at the time of the addition, he should logout and log back in for the changes to take effect and to be able then to log into Scratchbox.*

For some Linux distributions, for example Ubuntu 8.04, more amendments are required for Scratchbox to function correctly. See *Appendix - FAQs* on page 13 for more information.

The described procedure above is taken from

http://linux.onarm.com/index.php/From_scratch,

where you should refer to for any updates. In such a case, it is possible that the versions of the packages described on the web site may differ from the ones described here.

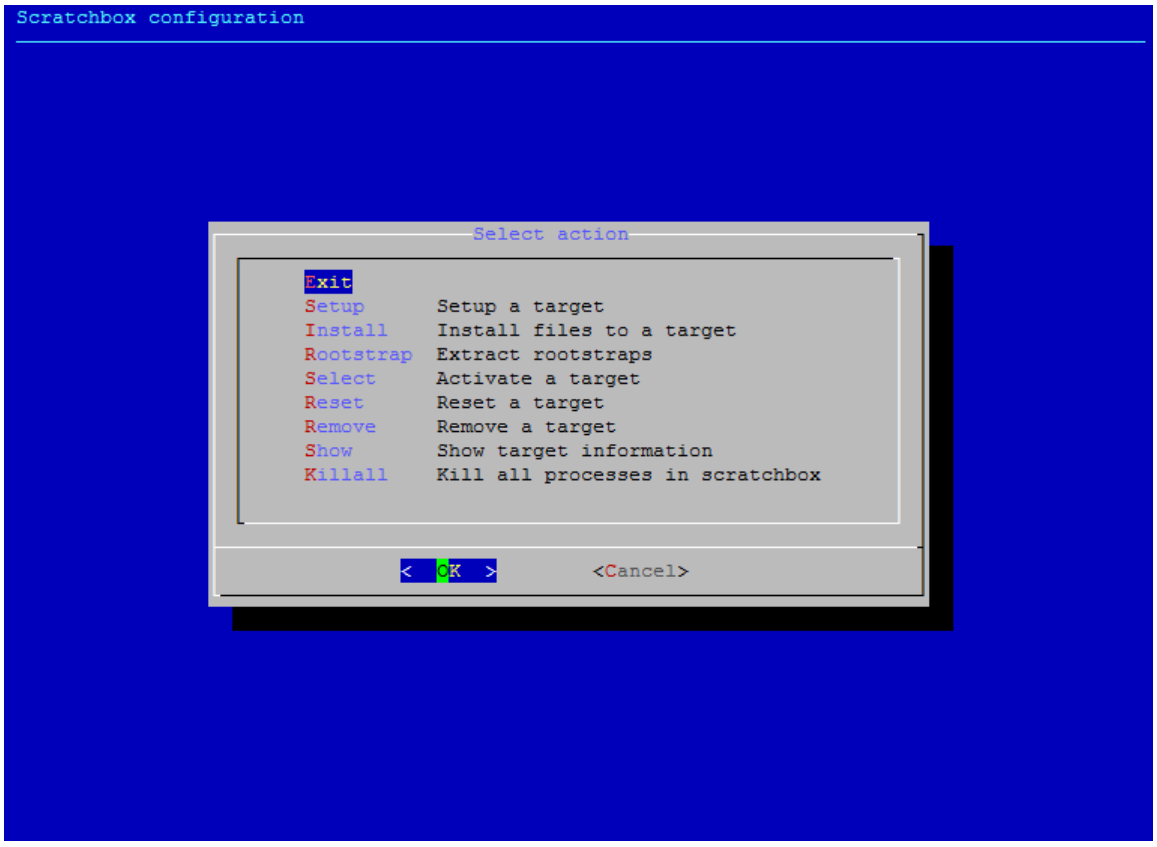
2.2 Target Setup

Firstly log in to Scratchbox by entering:

```
/scratchbox/login
```

Note *If there are problems logging in to Scratchbox (vdso, mmap), see Appendix - FAQs on page 13.*

A Scratchbox target can be set up by using the sb-menu utility. sb-menu is the Scratchbox configuration menu that allows you to create, configure, amend and delete Scratchbox targets. Scratchbox target is a folder created in the scratchbox installation tree and has the structure of a standard GNU/Linux root filesystem. The following figure depicts the sb-menu start-up
screen.



Step-by-step target creation and setup can also be found at:

http://www.scratchbox.org/documentation/user/scratchbox-1.0/html/installdoc.html#cctarget

Here we first demonstrate how to create a Scratchbox target that uses the CodeSourcery GNU toolchain 2007q3-51. Later in section 3, we will demonstrate how to use the ARM compiler in the Scratchbox environment.

In general the suggested procedure is as follows:

1. Run **`sb-menu`**.
2. Select **Set up a target**
3. Select **Create a new target** and enter a name for your new target; then proceed into configuring it in the next steps.
4. Select the compiler `arm-linux-cs2007q3-51 cross`
5. Select the following devkits: `doctools, perl, cputransp, mtd`
6. Select the CPU-transparency method, `qemu-arm-cvs-m`
7. Do not extract a `rootstrap` on the target, unless you have one that you want to use.
8. Install the `devkits` that are required by the target. These are `C-library, /etc, Devkits`, and `fakeroot`. Deselect `debug links` by pressing spacebar on your keyboard.

   Note *The installation might take some time. Wait for the next step before selecting so that the target can get initialized.*

9. Select the target.

This procedure was tested in the scope of this document, but you can select a different configuration if you want for other purposes.

2.2.1 Verify configuration

This section describes how to test Scratchbox with the `hello-world` project that is provided as
part of Scratchbox.

1. Untar the package from Scratchbox:

   ```
   tar -xzvf /scratchbox/packages/hello-world.tar.gz
   ```

2. Configure it:

   ```
   cd hello-world
   ./autogen.sh
   ```

3. Build it:

   ```
   make
   ```

4. Run it:

   ```
   ./hello
   ```

Running this project prints `Hello World!` on the standard output.

Note *In case you come across problems in this part, see Appendix - FAQs on page 13.*

3 Create and use the ARM Compiler Plugin

This section describes how to create, install and use the ARM compiler plugin for Scratchbox. It also provides instructions on how to compile an example application, and demonstrates how to switch toolchains to build the GNU/Linux project that was configured to build with GCC. The ability to switch compilers is enabled by the automatic GCC command-line translation in the ARM compiler.

The ARM compiler plugin works only with ARM compiler v4.0 or later.

3.1 Create the ARM Compiler plugin for Scratchbox

To perform the following you will have to be logged out of Scratchbox. To log out, enter `logout` or `exit`.

First, the ARM compiler plugin for Scratchbox must be created. For this purpose you must have the following installed on your Linux-running host:

- an existing RVDS 4.0 or RVDS 4.0 Professional installation
- perl v5.8.8 - This is the version of Perl that it was tested with
- armcc_pluginbuilder-v1.5.zip.

Unzip the `armcc_pluginbuilder-v1.5.zip` that you obtained from http://silver.arm.com

This will generate a folder that contains, among others, the following two files:

- the `pluginbuilder.pl` script
- `scratchbox-toolchain-armcc-vn-bn-1.0.11-i386-holder.tar.gz`

The pluginbuilder.pl script allows you to generate an ARM compiler plugin for Scratchbox using the ARM compiler installed as part of RVDS 4.0. This gives you the flexibility of having multiple plugins for Scratchbox, for example one plugin for each ARM compiler build installed on your Linux box.

Note *All these can coexist in Scratchbox, but they cannot be used at the same time. However, it is possible to access and invoke them all while in Scratchbox.*

The scratchbox-toolchain-armcc-vn-bn-1.0.11-i386-holder.tar.gz file, referred to as <pluginholder> below, operates as the plugin holder.

In order to create the plugin you must perform the following steps.

Note *Make sure that you execute the following as root, and that you set up the license environment variable for the ARM compiler so that it operates correctly. In addition make sure that you have at least 250MB of free space in the `/tmp` folder.*

Note *For more information on setting up your floating license server, see the ARM Technical Support Knowledge Article:*
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/3898.html

You can edit the pluginbuilder.pl script to meet your requirements, and you can also refer to it for more information since it contains comments that describe the whole procedure.

The syntax of the `pluginbuilder.pl` command is:

```
./pluginbuilder.pl <pluginholder> <armcc location> <libs path> <headers path> <codesourcery gcc path>
```

Where:

- <pluginholder> is the path of the respective file
- <armcc location> is the full path of the respective ARM Compiler binaries
- <libs path> is the full path of the respective ARM Compiler libraries
- <headers path> is the full path of the respective ARM Compiler header files.

For example, the command line might be:

```
./pluginbuilder.pl ./scratchbox-toolchain-armcc-vn-bn-1.0.11-i386-
holder.tar.gz \
       /opt/ARM/RVCT/Programs/4.0/650/linux-pentium/ \
       /opt/ARM/RVCT/Data/4.0/650/lib \
       /opt/ARM/RVCT/Data/4.0/650/include \
       /scratchbox/compilers/arm-linux-cs2007q3-51
```

Note *This script is executed from the same directory where it is stored. Ensure that you can execute the ARM compiler. Build 650 is a patch release for the ARM compiler 4.0. It is available for download from http://www.arm.com.*

Running this script produces a `scratchbox-toolchain-armcc-4.0-650-glibc-2007q3-51-1.0.11-i386.tar.gz` file. This is the ARM compiler plugin used in the rest of the procedure described in this document. If you have any problems when generating this plugin, for example a license problem preventing execution of the ARM compiler, then the `pluginbuilder.pl` will notify you.

During the wrapping procedure you will be asked to state the license server, that you should already have set up previously. The information must be entered in the `<PORT@LICENSE_SERVER>` form.

On completion, the resultant file name should resemble the following:

```
scratchbox-toolchain-armcc-4.0-650-glibc-2007q3-51-1.0.11-
i386.tar.gz
```

where 1.0.11 is the Scratchbox version, 650 is the ARM compiler build number, 4.0 is the ARM Compiler version, and glibc-2007q3-51 is the CodeSourcery glibc version that will be used.

Note *To get help on using the pluginbuilder.pl script you can use:*

```
./pluginbuilder.pl --help
```

3.2 Install the ARM Compiler Plugin for Scratchbox

The next step is to install the plugin to your Scratchbox installation. This step has to be performed out of Scratchbox. The following procedure uses the ARM compiler build number 650.

To install the plugin to Scratchbox, perform the following as root:

```
tar -C / -xzvf \
scratchbox-toolchain-armcc-4.0-650-glibc-2007q3-51-1.0.11-
i386.tar.gz
```

This places all the components of the ARM compiler plugin for Scratchbox into the appropriate places in your Scratchbox installation tree. The '`v`' in the arguments, will allow you to see where actually are the plugin components are installed. They should be placed under `/scratchbox` folder.

3.3 Use the ARM Compiler in Scratchbox

To use the ARM compiler 4.0 plugin, you must set up a Scratchbox target using `sb-menu` - as described in *Target setup* on page 7- but this time select the `armcc-4.0-650-glibc-2007q3-51 cross` as the compiler. You can also re-setup the target that you used previously and change the `arm-linux-cs2007q3-51sb3 cross` to `armcc-4.0-650-glibc-2007q3-51 cross`.

Note *The following instructions must be performed while logged into Scratchbox.*

After you select the target, perform the following:

```
source /scratchbox/compilers/armcc-4.0-650-glibc-2007q3-51/setup-rvct
```

to set up the environment variables required by the ARM compiler. In case you have not entered the license server information in the wrapping step, or if you have changed license serve, perform the following to point to your license server:

```
export ARMLMD_LICENSE_FILE=<PORT@LICENSE_SERVER>
```

or edit the corresponding line in `/scratchbox/compilers/armcc-4.0-650-glibc-2007q3-51/setup-rvct`.

Now you can invoke the ARM Compiler through the wrappings of Scratchbox by entering:

```
gcc
```

which is actually calling the armcc binary with GCC command line flags translation. You can verify the ARM compiler version by invoking with:

```
gcc –help
```

At this point you can test your ARM Compiler plugin for Scratchbox, by building the hello binary again.

Perform the following:

1. Enter the `hello-world` folder
2. clean-up

   ```
   make distclean
   ```

3. Configure

   ```
   ./autogen.sh
   ```

4. Build

   ```
   make
   ```

5. Test

   ```
   ./hello
   ```

This again should print `Hello world!` in the standard output.

You could also run:

```
file hello
```

which will give you information about the hello ARM binary, and:

```
fromelf hello | less
```

to get even more information about the binary, using the ARM `fromelf` tool.

Note *If you try to build a more complex application and run into problems, see Appendix - FAQs on page 13 that lists some of the known cases and the corrections or workarounds.*

If you wish to try building more complex examples, see http://linux.onarm.com for more information.

4 Appendix – FAQs

- *Question*: Why do I get the following message when I try to build an application?

```
Error: L6449E: While processing /scratchbox/compilers/armcc-4.0-
650/lib/gcc/arm-none-linux-gnueabi/4.2.0/libgcc.a: Symbol #7 in symbol
table section #10 is defined in section #17 but the last section is #11
```

  Answer: The libgcc.a file contained in the 2007q1-21 and 2007q3-51 CodeSourcery distributions contains object files with invalid symbols in the symbol table. For this reason, armlink generates an error when trying to read these object files.

  This can be corrected in 3 ways (see *Application Note 212*):
  - Under the GCC emulation mode of the ARM Compiler, invoke the link step using `-shared-libgcc` option. This forces the use of the shared library version of libgcc.
  - Strip your copy of the libgcc.a member object files to remove the corrupt symbols from the symbol table. See below for more instructions.
  - Upgrade to a newer CodeSourcery release. This issue is fixed in the 2008-q1 release.

  The procedure for stripping `libgcc.a` member object files to remove corrupt symbols from the symbol table is as follows:

  Note *It is recommended that you backup the original library and then replace it with the packaged one.*

  1. Unpack the library:

```
mkdir tmp
cd tmp
arm-none-linux-gnueabi-ar x \
/scratchbox/compilers/arm-linux-2007q1-21/.../libgcc.a
```

  2. Strip the broken object files:

```
arm-none-linux-gnueabi-strip --strip-unneeded _divdi3.o _divsi3.o \
_udivdi3.o _udivsi3.o
```

  3. Repackage the library:

```
arm-none-linux-gnueabi-ar cru libgcc_new.a *.o
```

- *Question*: Why do I get the following message?

```
Inconsistency detected by ld.so: rtld.c: 1192:dl_main: Assertion `(void *)
ph->p_vaddr == _rtld_local._dl_sysinfo_dso failed!
```

  Answer: In Ubuntu release 8.04 (Hardy), you must disable VDSO to have Scratchbox running. As a root enter:

```
echo 0 > /proc/sys/vm/vdso_enabled
```

- *Question*: Why do I get the following message when I try to build or run something in Scratchbox?

```
mmap: permission denied
```

  Answer: This is a problem in Ubuntu 8.04. It can be resolved by reducing the `mmap_min_addr` to 4096. As root enter:

```
echo 4096 >/proc/sys/vm/mmap_min_addr
```

- *Question*: Why while I am logged into Scratchbox, I cannot see my home folder?

  Answer: This is because while you are in Scratchbox, you cannot see any folders outside of the Scratchbox tree since Scratchbox login script changes the root folder to /scratchbox. Therefore, while in Scratchbox this is the highest place you can get in the folders hierarchy. While outside scratchbox you can copy files from you real home directory to your scratchbox home directory; for example:

  ```
  cp ~/... /scratchbox/users/$USER/home/$USER
  ```

- *Question*: Can I wrap the ARM Compiler with another version of glibc CodeSourcery for Scratchbox?

  Answer: Yes you can just pass the path of the CodeSourcery version that contains the glibc version that you want to use to the pluginbuilder command.

- *Question*: Why do I get the following message when I try to login to Scratchbox?

  ```
  ERROR: Scratchbox is not properly set up!
  ```

  Answer: Scratchbox has not started its services yet. Please enter as root while logged out of Scratchbox:

  ```
  /scratchbox/sbin/sbox_ctl start
  ```

- *Question*: How can I add more users to Scratchbox?

  Answer: You can add more users in Scratchbox by entering the following as a root:

  ```
  /scratchbox/sbin/sbox_adduser <username>
  ```

  Note *You can only add users that already exist in your Linux host box.*

- *Question*: Can I use sudo instead of switching to root?

  Answer: The whole procedure should work with sudo, but perhaps specific cases might need to be carried out in a slightly different way. The outline procedure was tested only with switching to root though.

- *Question*: Why do I get this message when I execute run_me_first.sh?

  ```
  Host kernel has vdso support (which is incompatible with SB)
  ```

  Answer: You can fix this as root with either:

  ```
  echo 0 > /proc/sys/vm/vdso_enabled
  ```
  or
  ```
  add 'vdso=0' to the kernel parameters
  ```

- *Question*: Can I automate the sourcing of `setup-rvct` ?

  Answer: Yes, you can create a `.bashrc` file in your home folder that contains the following line of code:

  ```
  source /scratchbox/compilers/<COMPILER NAME>/setup-rvct
  ```

  which will be executed every time you log into Scratchbox.

- *Question*: Why do I get the following link error when I am using glibc from CodeSourcery 2009q1-203 ?

```
Fatal error: L6029U: Relocation #REL:0 in crtn.o(.ARM.exidx.init) is
wrt invalid/missing symbol.
```

Answer: This is happening due to wrong symbols relocation in `crtn.o` provided by that glibc. There is a workaround implemented in ARM Compiler version 4.0 build 650, that downgrades this error to a warning. This enables the link to complete with GNU objects that contain invalid relocations to NULL. In this case you will get the following warning:

```
Warning: L6029E: Relocation #REL:0 in crtn.o(.ARM.exidx.init) is wrt
invalid/missing symbol.
Warning: L6029E: Relocation #REL:0 in crtn.o(.ARM.exidx.fini) is wrt
invalid/missing symbol.
```

5 Appendix – References

Please refer to the following pages for more information.

- http://linux.onarm.com/index.php/From_scratch
  - ARM's Linux Internet Platform (ALIP) website; instructions of building ALIP from scratch.

- http://www.scratchbox.org
  - Scratchbox website

- http://www.arm.com/support/newsgroup
  - ARM's Technical support's newsgroup