Debian / SH4

This activity was presented at LinuxCon Japan 2010.

1. Debian/SH4

1.1 Introduction

We, the Renesas Linux/SuperH development team, have posted Board Support Packages (BSPs) on this Website. The BSPs utilize the Debian project (http://www.debian.org/). Although there are Linux distributions targeting the SuperH family of processors, Debian was selected as our default distribution for SuperH processors for the following reasons:

- Community-based development

  The Debian project is primarily a volunteer-based organization developed through the efforts of a worldwide network of developers and contributors. Unlike commercial projects, the Debian project can involve anyone in its community-based Linux distribution development.
- Support of many software products

  The Debian project provides more than 20,000 software components as Debian packages. The Debian packages rank at the top in terms of quantity among the crowd of Linux distributions.
- Consolidated packaging and easy updating

  Numerous Debian software products are individually packaged, and managed by a package management system called "dpkg." The Debian system can be easily updated by virtue of the dependence between software packages and package updating functionality.
- Collaboration with upstream developers

  The Debian project has declared the Debian Social Contract with the free software community. One provision of the Debian Social Contract states: "We will give back to the free software community." Based on this provision, the Debian project communicates all problems found in Debian and the fixes to solve them to the upstream authors of works included in the Debian system, in order to let them share the information.
- Support and accommodation of various architectures

  Debian supports many types of architecture, including ten CPU architectures and two kernels. The Debian system also has a feature to distribute those software packages based on the same version of source code that can be run separately on individual CPU architectures. There is also additional CPU architectures and kernels currently being ported through the use of a porting infrastructure named "Debian Ports." Porting for Debian/SH4 is also being done by using Debian Ports.
- Support of embedded applications

  The Debian project has a subproject, named the "Embedded Debian (Emdebian)" project, that supports cross compilers and package cross-compiling for development of Debian distributions for embedded devices and systems.

The port of Debian to the SuperH family processors has not yet been officially released by the Debian project. We are now working porting to Debian for the SuperH processor family in collaboration with several Debian developers by using Debian Ports, which is the infrastructure to develop new ports of Debian. http://www.debian.org/ports/

This article describes the present status and usages of Debian/SH4.

1.2 SuperH Model To Be Supported by Debian

The SuperH family includes several processor types such as the SH2, SH3, and SH4. They vary in their support for various functions. Therefore, it is difficult to support all SuperH processors by using Debian (since some cannot be supported by Debian

due to their functionality). The current CPU model being worked on in Debian is the SH4 (little endian) because the SH4 is the present mainstream model among SuperH family processors and little-endian CPUs are popular among users. We call this Debian project "Debian/SH4."

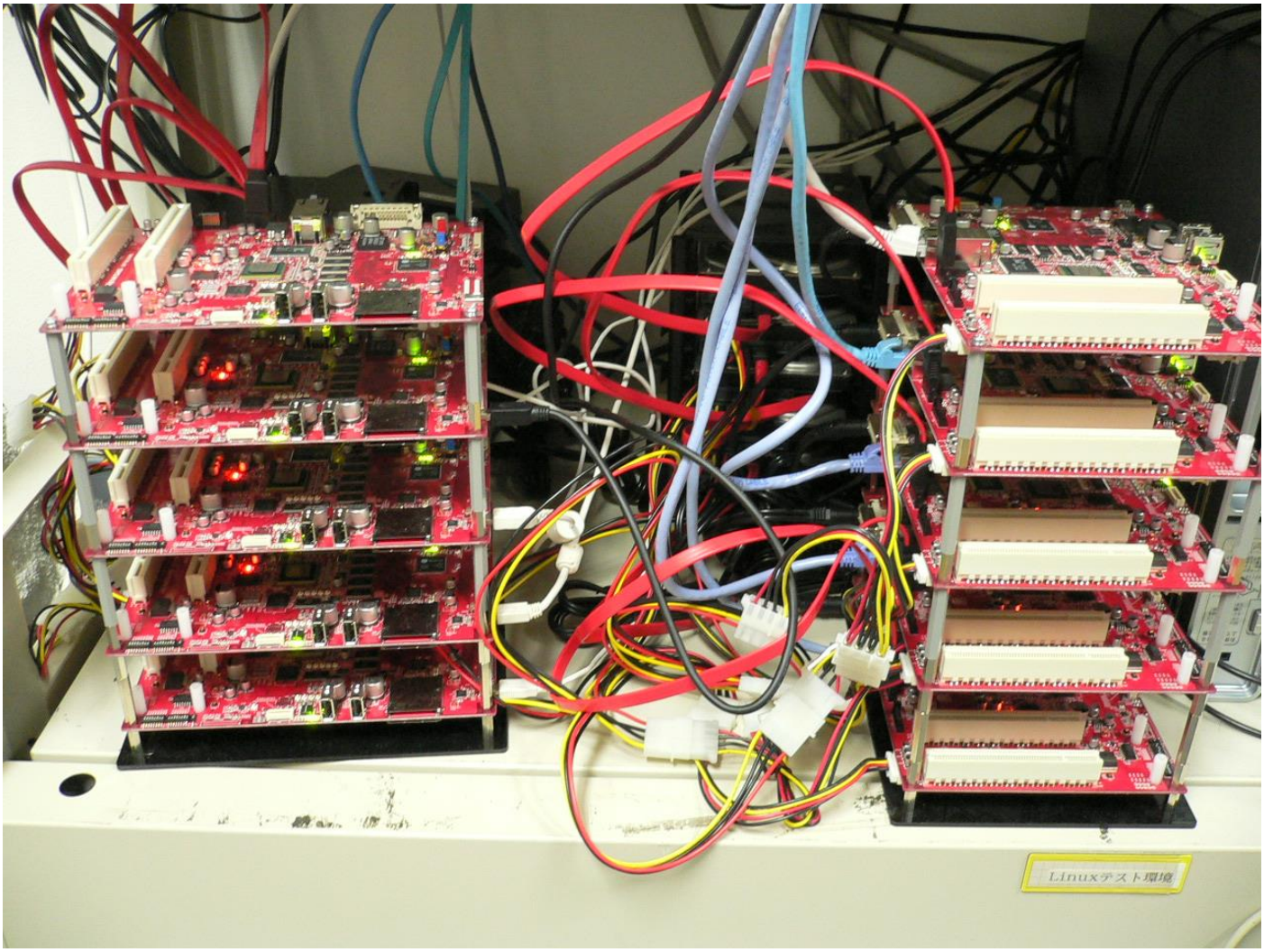
The SH4 is classified into two types: the SH4 (m4) and SH4A (m4a). Debian/SH4 is intended to support both types. Technically, Debian/SH4 is compiled only on the SH4 (m4). This is because the SH4 and SH4A are compatible in terms of instructions. The Debian/SH4 compiled on the SH4 (m4) can also run on the SH4A (m4a). However, you must be aware of the reverse usage. A Debian/SH4 system built on the SH4A (m4a) cannot run on the SH4 (m4). If you try to run such a system on the SH4 (m4), problems will be caused by the incompatibility between some CPU instructions, even though the system appears to run normally. You must also be careful not to run on the SH4 (m4) any Debian/SH4 system built on the SH4A (m4a). Debian/SH4 uses the default compiler provided by Debian for compilation control (the same compiler is used to compile Debian packages).

1.3 Debian/SH4 Support System

To support Debian/SH4, we are managing machines to build packages inside our project. In particular, we are building packages for Debian/SH4 using ten sh7785Icr boards on which the SH7785 is mounted.

These machines are linked to the database located in Debian Ports, and automatically compile Debian software. When a new version of a package related to Debian/SH4 is uploaded to the Debian project, the package and its update information are added to the database. Our machines automatically obtain the updated information and compile the package for Debian/SH4. For details on this automatic system, refer to "Outline of operation of the autobuilder network" on the Debian Project Website.

http://www.debian.org/devel/buildd/operation



1.4 Progress of Debian/SH4

The Debian/SH4 project began in June 2009. The present status of its progress is as follows (as of May, 2010):

Item	No. of packages	Percentage
Total No. of packages	8436	100%
Compiled packages	7766	92.55%
Packages failed in compiling	126	0.35%
Packages to be compiled	491	7.2%

About 8,500 packages must be compiled for each type of architecture. About 7,800 packages (or about 93% of those required) have already been compiled and are available for use. Available packages include "gcc" and "binutils" (needed to compile software), programming language packages "perl," "python," and "ruby," and such GUI tool kits as "Gtk" and "Qt."

Meanwhile, about 120 packages could not be compiled successfully or are being ported to SH4. There are also packages yet to be compiled because the target software does not support SH4 or has other problems. We will solve such problems step by step in cooperation with the Debian community.

For details on the progress of Debian/SH4, refer to the following Websites:

- Simplified status of building
  http://buildd.debian-ports.org/stats/sh4.txt
- Status of building
  http://buildd.debian-ports.org/status/architecture.php?a=sh4
- Graph for the status of building
  http://buildd.debian-ports.org/stats/graph-big.png

1.5 Schedule on Release

As mentioned above, Debian/SH4 is not yet a target of the official Debian release. We are now developing Debian/SH4 so that it will be released with the next Debian release (Debian Squeeze +1, which is an updated version of Debian Squeeze).

2. How To Use Debian/SH4

Debian/SH4 is still under development, but can be used as a Debian system. This chapter explains the basic methods of using Debian. Assume that the operations described below are performed on Debian/SH4 and the machine with Debian/SH4 installed is connected to the Internet. Note that the symbol "$" in the command line indicates the console, so you need not enter this symbol.

2.1 Updating the Repository

To obtain the latest information about Debian/SH4 packages, you must update local information on the repository that manages Debian packages. To update the information, execute the apt-get command as shown below. This apt-get command obtains update information from the repository.

```
root@ecovec:~# apt-get update
Get:1 http://ftp.jp.debian.org unstable Release.gpg [835B]
Get:2 http://ftp.jp.debian.org unstable Release [104kB]
Get:3 http://ftp.jp.debian.org unstable/main Sources/DiffIndex [2038B]
Get:4 http://ftp.debian-ports.org unstable Release.gpg [835B]
Get:5 http://ftp.debian-ports.org unreleased Release.gpg [835B]
Get:6 http://incoming.debian-ports.org unstable Release.gpg [835B]
Get:7 http://ftp.jp.debian.org unstable/main Sources [3950kB]
Get:8 http://ftp.debian-ports.org unstable Release [17.9kB]
Get:9 http://incoming.debian-ports.org unstable Release [7968B]
Get:10 http://ftp.debian-ports.org unreleased Release [19.9kB]
Ign http://incoming.debian-ports.org unstable/main Packages/DiffIndex
Get:11 http://incoming.debian-ports.org unstable/main Packages [16.7kB]
Ign http://ftp.debian-ports.org unstable/main Packages/DiffIndex
Ign http://ftp.debian-ports.org unreleased/main Packages/DiffIndex
Get:12 http://ftp.debian-ports.org unstable/main Packages [6473kB]
Get:13 http://ftp.debian-ports.org unreleased/main Packages [14.2kB]
Fetched 10.6MB in 1min 56s (91.2kB/s)
Reading package lists... Done
```

2.2 Updating the Root File System

To update the root file system, execute the apt-get command as shown below.

```
root@ecovec:~# sudo apt-get upgrade
```

This apt-get command updates the root file system. Note that the root file system may be destroyed if any updated Debian package has a problem. Therefore, be very careful when updating the root file system.

2.3 Searching Debian Packages

You may have trouble finding the packages you desire among the numerous packages provided by the Debian project. This section briefly explains how to search for desired packages efficiently.

2.3.1 Searching for a Package with a Package Name or Description

Each Debian package has a package description as part of it's package information. To find a package or obtain information about a package, you can search for the package by using it's package name or keyword found in its package description. For searching, use the search option of the apt-cache command. For example, execute the apt-cache command as shown below to search for packages related to the character string "hello".

```
root@ecovec:~# apt-cache search hello
gnome-games - games for the GNOME desktop
gpe-othello - othello board game for GPE
grhino - othello/reversi boardgame
gtkboard - many board games in one program
hello - The classic greeting, and a good example
hello-debhelper - The classic greeting, and a good example
jester - board game similar to Othello
quarry - Board games Go, Amazons, and Reversi (a.k.a. Othello)
vdr-plugin-examples - Plugins for vdr to show some possible features
xprint-utils - utilities for Xprint, the X11 print system
junior-system - Debian Jr. System tools
python-flask - micro web framework based on Werkzeug, Jinja2 and good intentions
gettext-doc - Documentation for GNU gettext
gnat-gps-doc - The GNAT Programming System - documentation
grhino-data - othello/reversi boardgame - data-files
libluabind-examples - luabind c++ binding for lua: example files
liblog-tracemessages-perl - Perl module to allow for trace messages in Perl code
libscalar-properties-perl - perl module to add run-time properties on scalar variables
python-pyparsing - Python parsing module
texlive-games - TeX Live: Games typesetting
```

You can specify multiple keywords by delimiting them with a space.

```
root@ecovec:~# apt-cache search hello example
hello - The classic greeting, and a good example
hello-debhelper - The classic greeting, and a good example
vdr-plugin-examples - Plugins for vdr to show some possible features
junior-system - Debian Jr. System tools
gettext-doc - Documentation for GNU gettext
libluabind-examples - luabind c++ binding for lua: example files
libscalar-properties-perl - perl module to add run-time properties on scalar variables
python-pyparsing - Python parsing module
```

2.3.2 Listing Currently Installed Packages

When you want to output the packages currently installed in the root file system in a list, execute the dpkg -l command as shown below.

```
root@ecovec:~# dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Cfg-files/Unpacked/Failed-cfg/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  adduser        3.110          add and remove users and groups
ii  alsa-base      1.0.21+dfsg-2  ALSA driver configuration files
ii  alsa-utils     1.0.21-1       ALSA utilities
ii  apt            0.7.21         Advanced front-end for dpkg
ii  apt-utils      0.7.21         APT utility programs
ii  aptitude       0.4.11.11-1    terminal-based package manager
(snip)
```

2.3.3 Searching for Packages with a Web Browser

When you are not using the Debian/SH4 machine but using a personal computer connected to the Internet, you can use the Web services provided by the Debian project to search for packages via a Web browser.

- Service to search package directories and package contents

  http://www.debian.org/distrib/packages#search_packages

- Debian Package Tracking System

  https://packages.qa.debian.org/common/index.html

2.4 Installing a Debian Package

This section explains how to install a Debian package in the root file system by using an example of installing the hello package you found through the package search described above.

First, try to execute the program "hello" included in the hello package from the command line. Because the hello package is not yet installed, execution fails with the error message "command not found" being output.

```
root@ecovec:~# hello
-bash: hello: command not found
```

Next, install the hello package. To install a Debian package, use the apt-get command with the install option specified. Execute the apt-get command for the hello package. This command outputs log data and installs the hello package as shown below.

```
root@ecovec:~# apt-get install hello
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  hello
0 upgraded, 1 newly installed, 0 to remove and 140 not upgraded.
Need to get 0B/58.3kB of archives.
After this operation, 659kB of additional disk space will be used.
Selecting previously deselected package hello.
(Reading database ... 12312 files and directories currently installed.)
Unpacking hello (from .../archives/hello_2.5-1_sh4.deb) ...
Processing triggers for man-db ...
fopen: Permission denied
Processing triggers for install-info ...
Setting up hello (2.5-1) ...
```

After installing the hello package, try to execute the hello program again. When the hello package has been installed normally, the following message is output:

```
root@ecovec:~# hello
Hello, world!
```

2.5 Obtaining Source Code for Debian Packages

This section explains how to obtain the source code for Debian packages.

First, check that the /etc/apt/sources.list file includes the line shown below. If not, add the line to the file.

```
root@ecovec:~# cat /etc/apt/sources.list
deb http://ftp.debian-ports.org/debian/ unstable main
deb http://ftp.debian-ports.org/debian/ unreleased main
deb http://incoming.debian-ports.org/buildd/ unstable main

# get source
deb-src http://ftp.jp.debian.org/debian unstable main
```

Next, execute the apt-get commands as shown below. These commands obtain source code information from the package repository and install the dpkg-dev package.

```
root@ecovec:~# apt-get update
Hit http://ftp.jp.debian.org unstable Release.gpg
Hit http://ftp.jp.debian.org unstable Release
Hit http://ftp.jp.debian.org unstable/main Sources/DiffIndex
Hit http://ftp.debian-ports.org unstable Release.gpg
Hit http://ftp.debian-ports.org unreleased Release.gpg
Hit http://incoming.debian-ports.org unstable Release.gpg
Hit http://ftp.debian-ports.org unstable Release
Hit http://incoming.debian-ports.org unstable Release
Hit http://ftp.debian-ports.org unreleased Release
Ign http://incoming.debian-ports.org unstable/main Packages/DiffIndex
Ign http://ftp.debian-ports.org unstable/main Packages/DiffIndex
Hit http://incoming.debian-ports.org unstable/main Packages
Ign http://ftp.debian-ports.org unreleased/main Packages/DiffIndex
Hit http://ftp.debian-ports.org unstable/main Packages
Hit http://ftp.debian-ports.org unreleased/main Packages
Reading package lists... Done
```

```
root@ecovec:~# apt-get install dpkg-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libstdc++6-4.3-dev g++-4.3
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  build-essential
Suggested packages:
  debian-keyring
The following NEW packages will be installed:
  build-essential dpkg-dev
0 upgraded, 2 newly installed, 0 to remove and 130 not upgraded.
Need to get 7006B/415kB of archives.
After this operation, 762kB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://ftp.debian-ports.org unstable/main build-essential 11.5 [7006B]
Fetched 7006B in 1s (6476B/s)
Selecting previously deselected package dpkg-dev.
(Reading database ... 13113 files and directories currently installed.)
Unpacking dpkg-dev (from .../dpkg-dev_1.15.7.2_all.deb) ...
Selecting previously deselected package build-essential.
Unpacking build-essential (from .../build-essential_11.5_sh4.deb) ...
Processing triggers for man-db ...
fopen: Permission denied
Setting up dpkg-dev (1.15.7.2) ...
Setting up build-essential (11.5) ...
```

After obtaining the repository information and installing the dpkg-dev package, execute the apt-get command as shown below to obtain the source code. This apt-get command obtains the source code for the hello package.

```
root@ecovec:~# apt-get source hello
Reading package lists... Done
Building dependency tree
Reading state information... Done
Need to get 590kB of source archives.
Get:1 http://ftp.jp.debian.org unstable/main hello 2.5-1 (dsc) [1219B]
Get:2 http://ftp.jp.debian.org unstable/main hello 2.5-1 (tar) [583kB]
Get:3 http://ftp.jp.debian.org unstable/main hello 2.5-1 (diff) [5965B]
Fetched 590kB in 1s (458kB/s)
gpgv: directory `/root/.gnupg' created
gpgv: new configuration file `/root/.gnupg/gpg.conf' created
gpgv: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run
gpgv: keyblock resource `/root/.gnupg/trustedkeys.gpg': general error
gpgv: Signature made Sun Feb 14 17:26:43 2010 UTC using RSA key ID 9F1B8B32
gpgv: Can't check signature: public key not found
dpkg-source: warning: failed to verify signature on ./hello_2.5-1.dsc
dpkg-source: info: extracting hello in hello-2.5
dpkg-source: info: unpacking hello_2.5.orig.tar.gz
dpkg-source: info: applying hello_2.5-1.diff.gz
dpkg-source: info: upstream files that have been modified:
 hello-2.5/doc/Makefile.in
```

The source code is extracted into the directory where the command was executed and can be obtained as a hello-2.5 source package.

2.6 Compiling Debian Packages (Creating Binary Packages)

This section explains how to compile source code by using an example of compiling the hello source code. First, move to the hello-2.5 directory.

```
root@ecovec:~# cd hello-2.5/
root@ecovec:~/hello-2.5#
```

Next, execute the command shown below. This command installs the package required to compile the hello source package.

```
root@ecovec:~/hello-2.5# apt-get build-dep hello
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 130 not upgraded.
```

Also execute the command shown below to install the devscripts package, which is an auxiliary package used to create a package.

```
root@ecovec:~/hello-2.5# apt-get install devscripts
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libstdc++6-4.3-dev diffstat g++-4.3 libapt-pkg-perl libipc-run-perl
  libio-pty-perl
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  apt apt-utils aptitude at autopoint ca-certificates conkeror
  conkeror-spawn-process-helper dbus dctrl-tools debhelper debian-keyring
  defoma dictionaries-common diffstat dput equivs exim4 exim4-base
(snip)
```

After completing all the operations above, execute the command shown below.

```
root@ecovec:~/hello-2.5# debuild -us -uc
 dpkg-buildpackage -rfakeroot -D -us -uc
dpkg-buildpackage: warning: using a gain-root-command while being root
dpkg-buildpackage: export CPPFLAGS from dpkg-buildflags (origin: vendor):
dpkg-buildpackage: export CFLAGS from dpkg-buildflags (origin: vendor): -g -O2
dpkg-buildpackage: export CXXFLAGS from dpkg-buildflags (origin: vendor): -g -O2
dpkg-buildpackage: export FFLAGS from dpkg-buildflags (origin: vendor): -g -O2
dpkg-buildpackage: export LDFLAGS from dpkg-buildflags (origin: vendor):
dpkg-buildpackage: source package hello
dpkg-buildpackage: source version 2.5-1
(snip)
```

In several minutes, a hello binary package will be generated in a directory one level higher than that of the current directory.

```
root@ecovec:~/hello-2.5# ls ../*.deb
../hello_2.5-1_i386.deb
```

3. Advanced Usage of Debian/SH4

On one hand, Debian packages are very easy to use. On the other, if Debian packages are installed on a reference board with an embedded CPU (e.g., a SuperH processor), the root file system will be too large to load into flash ROM or other small-sized ROM.

The Debian project is implementing the Emdebian project to provide Debian for embedded devices. Emdebian Grip (a product of the Emdebian project) allows you to reduce the size of the root file system while taking advantage of Debian packages. This chapter explains how to use Emdebian Grip and how to further reduce the size of the root file system.

http://www.emdebian.org/

3.1 Emdebian Grip

Emdebian Grip converts existing Debian packages into those for embedded devices. You (users) need not perform conversion, but can obtain converted packages from the Emdebian Website. Original Debian packages and converted packages are compatible, and any of them can be used as standard Debian packages. Given this compatibility, a root file system can contain both the packages converted by Emdebian Grip and their original packages together.

A Debian package converted from an original Debian package has a version different from that of the original package. The package version number of a converted package has the suffix "emX." The "X" in "emX" indicates the number of times the package has been built. The suffix is usually "em1". Table 1 shows the difference in package version between a converted apt package and its original apt package.

Table 1 Versions of Debian apt Package Before and After Conversion

-	Version
Example of version before conversion	apt_0.7.25.3_sh4.deb
Example of version after conversion	apt_0.7.25.3em1_sh4.deb

If you need to convert a package, use the em_autogrip command provided in the emdebian-grip-server package. You (users) might not have any opportunity to use the command because most Debian packages have already been converted. For further information, refer to the relevant page on the Emdebian Website.

http://linux.codehelp.co.uk/emdebian/man/em_autogrip.html

```
$ em_autogrip -b /target/repository/path -s apt
```

Figure 1 Example of the em_autogrip Command

3.1.1 How To Use Emdebian Grip

To use the packages converted by Emdebian Grip, you must change the repository path called "apt-line." The apt-line is specified in the /etc/apt/sources.list file. Use an editor to change the apt-line as shown in figure 2.

```
Before the change: deb http://ftp.debian-ports.org/debian/ unstable main
After the change: deb http://www.emdebian.org/grip/ unstable main
```

Figure 2 Changing the apt-line to Emdebian Grip

After changing the apt-line, execute the apt-get update command to obtain repository information. Then, execute the apt-get upgrade command to switch the whole root file system to that of Emdebian Grip.

```
root@ecovec:~# apt-get update
Get:1 http://ftp.jp.debian.org unstable Release.gpg [835B]
Get:2 http://ftp.jp.debian.org unstable Release [104kB]
Get:3 http://incoming.debian-ports.org unstable Release.gpg [835B]
Get:4 http://ftp.debian-ports.org unreleased Release.gpg [835B]
Get:5 http://www.emdebian.org unstable Release.gpg [197B]
Get:6 http://ftp.debian-ports.org unreleased Release [19.9kB]
Get:7 http://incoming.debian-ports.org unstable Release [7968B]
Get:8 http://ftp.jp.debian.org unstable/main Sources/DiffIndex [2038B]
Get:9 http://www.emdebian.org unstable Release [32.9kB]
Ign http://www.emdebian.org unstable Release
Get:10 http://ftp.jp.debian.org unstable/main 2010-06-01-0902.49.pdiff [8554B]
Get:11 http://ftp.jp.debian.org unstable/main 2010-06-01-0902.49.pdiff [8554B]
Ign http://www.emdebian.org unstable/main Packages
Ign http://ftp.debian-ports.org unreleased/main Packages/DiffIndex
Get:12 http://ftp.jp.debian.org unstable/main 2010-06-01-0902.49.pdiff [8554B]
Ign http://incoming.debian-ports.org unstable/main Packages/DiffIndex
Ign http://www.emdebian.org unstable/main Packages
Hit http://ftp.debian-ports.org unreleased/main Packages
Get:13 http://incoming.debian-ports.org unstable/main Packages [22.4kB]
Get:14 http://www.emdebian.org unstable/main Packages [532kB]
Fetched 732kB in 24s (29.7kB/s)
Reading package lists... Done
```

```
root@ecovec:~# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
  iptables libc-bin libc6 libdpkg-perl libfreetype6 libjpeg62 libncurses5
  libpng12-0 ncurses-bin ocaml-interp zlib1g
The following packages will be upgraded:
  adduser alsa-base alsa-utils apt apt-utils aptitude at base-files
  base-passwd bash bsdmainutils bsdutils bzip2 ca-certificates coreutils cpio
  cpp cpp-4.3 cron dash dctrl-tools debconf debconf-i18n
  debian-archive-keyring debianutils defoma dhcp3-client dhcp3-common
  dictionaries-common diff diffutils dpkg e2fslibs e2fsprogs ed emdebian-grip
  emdebian-tdeb exim4 exim4-base exim4-config exim4-daemon-light fakeroot file
  findutils fontconfig fontconfig-config g++-4.3 gcc-4.3 gcc-4.3-base
  gettext-base gnupg gpgv grep groff-base gzip hicolor-icon-theme hostname
  html2text ifupdown initscripts insserv install-info iproute iputils-ping
  iso-codes less libacl1 libapr1 libaprutil1 libapt-pkg-perl libarchive1
  libasound2 libatk1.0-0 libatk1.0-data libattr1 libavahi-client3
  libavahi-common-data libavahi-common3 libblkid1 libboost-iostreams1.40.0
  libbsd0 libbz2-1.0 libcairo2 libclass-accessor-perl libcomerr2
  libconfig-auto-perl libconfig-inifiles-perl libcroco3 libcups2
(snip)
```

3.1.2 Comparison of Package Sizes Before and After Package Conversion

Let's look at how a package is changed through package conversion by Emdebian Grip.

Table 2 Package Sizes Before and After Conversion

-	Size	Package file name
Before conversion	1.8 MB	apt_0.7.25.3_sh4.deb
After conversion	638 KB	apt_0.7.25.3em1_sh4.deb

Table 2 shows an example of the change in package size by conversion of an apt package. This example indicates that the size of converted package is about one-third the original package size.

You can see that the package size is reduced. How is the package size reduced? Table 3 lists the number of directories and files in the converted package and original package. You can see that many directories and files are reduced by package conversion. Emdebian Grip eliminates the directories and files related to the main program, documents, and message catalogs.

Table 3 Numbers of Directories and Files Before and After Conversion

-	No. of directories	No. of files
Before conversion	136 directories	212 files
After conversion	37 directories	34 files

3.1.3 Comparison of System Sizes Before and After Switching to Emdebian Grip

Now, let's look at the whole system. Here, we only address the base system (i.e., the basic system of Debian), switch the base system to that of Emdebian Grip, and then compare the sizes of the Debian base system and Emdebian Grip base system. Assume that you changed the apt-line and then executed the apt-get upgrade command as described in section 3.1.1 to switch the whole system to the Emdebian Grip system. Table 4 lists the system sizes obtained before and after switching the systems. The system size becomes less than half. Note that the reduced size indicates the size of the root file system when not compressed. You can compress the root file system into a smaller size with a compression tool, e.g., the journaling flash file system (JFFS2).

Table 4 Total Sizes of Root File System Before and After System Switching

-	Size
Before switching	186 MB
After switching	79 MB

Next, let's compare the size of individual directories located in the root directory. Figure 3 shows the directory sizes obtained before and after switching the systems. You can see that switching the systems reduces the sizes of the /usr and /var directories.

Table 5 Comparison of the Sizes of Directories in the Root File System

Name	Before Switching	After switching
bin	5	5
boot	1	1
dev	1	1
etc	2	2
home	1	1
lib	9	9
media	1	1
mnt	1	1
opt	1	1
proc	1	1
root	1	1
sbin	4	4
selinux	1	1
srv	1	1
sys	1	1
tmp	1	1
usr	108	50
var	58	11

3.2 Summary of Emdebian Grip Usage

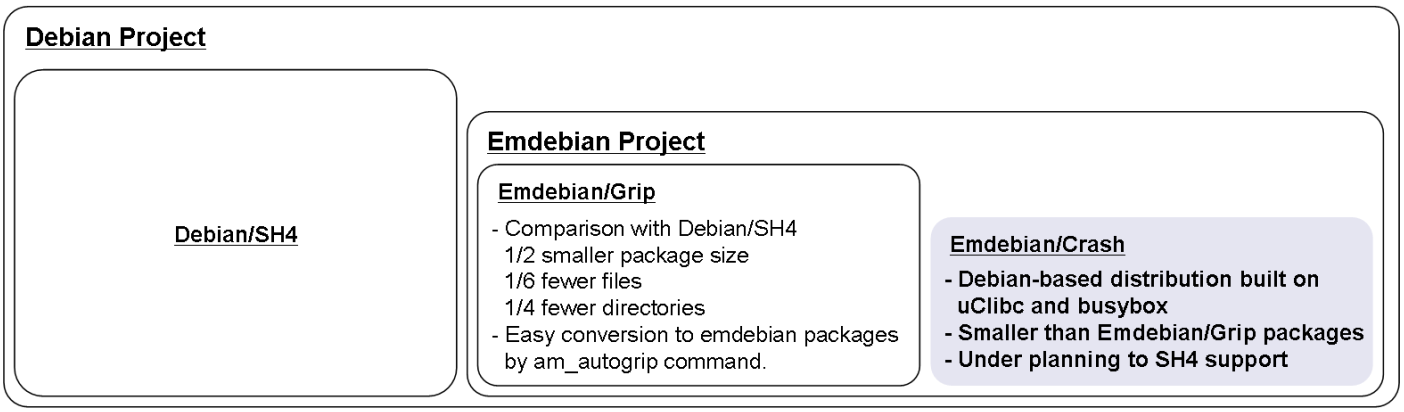
As described above, Emdebian Grip allows you to greatly reduce the size of the root file system. Furthermore, Emdebian Grip can reduce the system size while maintaining such features of a Debian package as the dependence between packages and ease of package installation. You can consider Emdebian Grip an option of the OS for processor boards on which compact flash ROM is mounted for embedded applications.

3.3 How To Further Reduce the Size of the Root File System

You can reduce the size of the root file system by using Emdebian Grip as described above. Some of you might want to further reduce the system size. This section suggests some other methods of reducing the size of the root file system.

3.3.1 Emdebian Crush

Emdebian Crush is a Debian-based distribution built on uClibc and busybox. Similar to Emdebian Grip, Emdebian Crush is being developed by the Emdebian project. Although Emdebian Crush currently supports only the armel architecture, we intend to make Emdebian Crush also available for Debian/SH4.



3.3.2 Other Tools

Other tools used to reduce the size of the root file system include openembedded, openwrt, buildroot, and Scratchbox, but are not explained here.

4. Native and Cross Compilers

Debian uses native compilers to compile binary packages for individual architectures. The default C complier is gcc, and multiple versions of gcc can be used to compile Debian packages. As of April 2010, Debian/SH4 supports gcc-4.3 and gcc-4.4 and includes both in its distribution.

We usually use cross compilers for developing software for embedded applications. Because the CPUs for embedded use are less powerful than those of desktop machines, we can compile the binary packages for target architectures much more quickly by running cross compilers on fast CPUs.

The Emdebian project supports the use of Debian products and tools for embedded devices and systems. The Emdebian project provides cross compilers compatible with the native compilers of Debian.

You can effectively use the cross compilers provided by the Emdebian project. However, we recommend that you use the cross compiler provided by CodeSourcery, which is known for its development of gcc and eglibc. Renesas requested CodeSourcery offer a cross compiler intended for the SH4, and has made the cross compiler available to users.

http://www.mentor.com/embedded-software/codesourcery

4.1 Compatibility of Debian and CodeSourcery Cross Compilers

We have applied a patch concerning the _fpcr_value to eglic in Debian/SH4 distribution. Note that eglic of Debian/SH4 is not compatible with binary packages that were built with a cross compiler, eglibc, or glibc included in other Debian distributions. The cross compiler provided by CodeSourcery, however, eliminates this incompatibility and can generate binary packages compatible with other Debian products. In the future, we will request the GNU project, which is the Debian's upstream project incorporate the said patch in order to resolve this compatibility issue.

http://www.gnu.org/