To our customers,

Old Company Name in Catalogs and Other Documents

   On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SH7670 SHLinux –
Technical Guide

Introduction

This document describes the Linux features specific to the SH7670 CPU and the RSK+ development board. The SH7670 RSK+ BSP is distributed as a compressed tar archive. During this document, you will create a tool chain, and build a working environment for the SH7670 RSK+. This document assumes the reader has a basic knowledge of Linux.

Requirements

- To work with the SH7670 RSK+ BSP, you will need a Windows Host environment, and a GNU/Linux Development Environment.

- The Renesas High Performance Embedded Workshop (HEW) runs in a Windows environment, and needs to be installed to facilitate the downloading of images to the RSK+ board using a Renesas E10A.

- To build and develop on the µClinux environment, you will need a Linux development machine. This can be a second physical machine, or a virtual machine sharing the Windows environment.

On a clean Ubuntu distribution, packages must be added to support this build process.
You will need to enter the following commands, or ensure the following packages are installed if you use a different distribution.

```
sudo apt-get install gcc
sudo apt-get install build-essential
sudo apt-get install ncurses-dev
sudo apt-get install mtd-tools
```
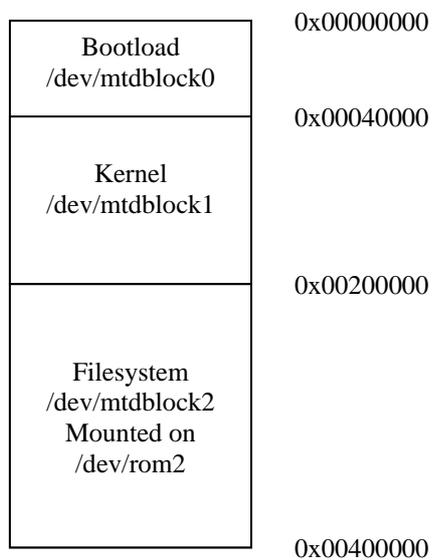
The BSP was compiled with gcc4.0 though gcc >4.0 should work but hasn't been tested.

The Linux development environment will require at least 1.0 GB of free space.

This document is based on work completed using Ubuntu 8.04 default install with the above packages as the development environment, and an SH7670 RSK+ development board (issue 1). Other Linux distributions may be used, but please be aware of slight differences that may occur in command usage and package management.

SH7670 Flash Memory Partition Map

The following partition mappings have been used in building the SH7670 RSK+ BSP.
These FLASH partitions are mapped by the MTD driver to the RSK+ Flash Device.

Bootload /dev/mtdblock0	0x00000000
Kernel /dev/mtdblock1	0x00040000
Filesystem /dev/mtdblock2 Mounted on /dev/rom2	0x00200000
	0x00400000

The above diagram shows the physical addresses of the partitions, and the logical naming of the partitions in the /dev subsystem.

SH7670 SHLinux – Technical Guide

SH7670 RSK+ BSP Project Structure

To get started, extract the compressed archive to your home directory or desired location in the Linux environment.

```
cd ~
tar xvf  /media/cdrom/sh7670-uClinux-RC4.tar.gz
```

Note:
The release component suffix (RC4 in this case) is the issue number of the BSP.

The SH7670 BSP consists of the following locations that you need to be aware of when developing:

Build Scripts	Master copies of configuration files, project make-files for cross-compilation and built material.
\configs \target \stage \sh7670-images	- Configuration file location - Built toolchain location - Build location for packages - Final location for shsh7670-linux.bin and sh7670-filesystem.bin
Tool chain	The following packages are built to create a tool chain for the SH7670 by the build scripts.
binutils-2.17 elf2flt gcc-3.4.6 uClibc-0.9.27-sh2	- Assembler and Utilities required for the GNU C Compiler - FLT libraries and Linker scripts to run applications in a flat memory model. - The GNU Compiler - C Libraries
Source Packages	
Bootstrap2rsk linux-2.6.19-sh7670 init busybox-1.5.1 protofs fbv-1.0b WebFE	- A basic functionality bootstrap which displays the splash screen and loads Linux. - The µClinux source tree for the SH7670 board and drivers. - Provides initial initialisation and bootscripts for userspace. - Main utilities including the Shell and Web Server. - Contains the filesystem structure, utilities and images. - An image viewer to display images on the QVGA display. - A demonstration Web based front end that can control the QVGA
Extras fmtool bmprle zips	 - Flash Memory Tool Scripts for use with HEW E10A Support - Sample application for converting a BMP file into the special RLE encoded C array - A container folder that holds the compressed packages.
Output build\sh7670-images	 - sh7670-linux.abs.bin (kernel file + bootstrap) - sh7670-filesystem.bin (filesystem file size 2MB) - sh-uclinux-full.bin (combined file using full-image script size 4MB)

Build Scripts

The build directory should be the base directory for compilation of the SH7670 BSP and tool chain. The Makefile in the build directory will build the tool chain, followed by the BSP the first time it is run.
You will need to know your account password for the **sudo** command.
Execute the following in the projects root to begin:

```
cd build
make
sudo make protofs-install
make full-image
```

The first build takes around 45 minutes. The build scripts will extract the required packages from compressed files stored in the zips directory to create the entire SH7670 Tool chain, and BSP.

The Build directory has been created such that the top level makefile can control the entire build process. The following actions are supported.

Make	Will build any updated packages.
make clean	Cleans the BSP but will not remove the tool chain.
make distclean	Cleans both BSP and tool chain, requiring a complete rebuild.

You can also build specific targets in the BSP as long as there is a prebuilt tool chain; for example:

```
make linux
make fbv-1.0b
make busybox
make WebFE
```

Rebuilding any package which is installed to the filesystem will require a re-make of the filesystem image using:

```
sudo make protofs-install
```
Root access is required to create the device nodes in the /dev directory.

```
make full-image
```
This script will make a 4MB file buy padding the kernel to 1792 KB and filesystem to fit 2MB .
A 4MB file is then assembled to file the Partition Map (above)

Name	Size	Offset
/dev/mtdblock0	256KB	0x0
/dev/mtdblock1	1792KB	0x4000
/dev/mtdblock2	2048KB	0x20000

Whilst in the build directory, you can call the package makefile directly to access more targets. Use the make -f option for this.

```
make -f Makefile.linux-sh7670 build
make -f Makefile.linux-sh7670 configure
```

If you wish to add new packages to be built for the SH7670 RSK+ µClinux distribution, it is recommended that you create a directory to contain the source code and build, and create a makefile in the build directory that uses the environment provided.
You can copy the existing makefiles for examples of creating your own packages.

Toolchain

The SH7670 RSK+ toolchain consists of the following packages.

    binutils-2.17
    elf2flt-sh2
    gcc-3.4.6
    uClibc-0.9.27-sh2

Currently the toolchain is configured by default to support the C language, and is targeted to the SH2 processor with Big Endian (MSB).

Elf2flt libraries are installed into the toolchain to provide flat executable support for userspace programs. This is used by passing the "-Wl,-elf2flt" flags at compilation and linking.

The toolchain when built is put into the target directory under build. This destination is defined by the make variable $(PREFIX).

Bootstrap

The bootstrap directory contains the source for a basic bootloader for the SH7670 RSK+. This bootloader initialises the CPU and board registers, the bus state controller, SDRAM, flash and serial UART for operation.

The serial UART is currently configured to run at 19200 baud with 8 Bits, no Parity, 1 Stop Bit and no Flow Control. (57600,n,8,1)

The bootloader will perform some early checks to test the current bus width mode the board has been set to. In order to support the QVGA screen, the SH7670 RSK+ board must be set in 16 bit bus width by setting SW4 bit 10 to the off position. If this is incorrectly set, a warning will be displayed on the serial output indicating a change needs to be made.

The bootloader is also responsible for initialising the QVGA display, loading a splash image, and updating the QVGA. Due to space constraints, the boot splash image is stored as a custom type RLE image with a simplistic extraction algorithm.

An example utility called 'bmprle' has been provided for converting BMP files into the RLE encoding format, as a C Array. The image is compiled into image.c along with the extraction algorithms. These can be changed and updated to support new images or other forms of image compression if the developer wishes.

The final task of the bootloader is to pass control to the decompression routines of the Linux compressor which takes responsibility of extracting Linux to the correct location in SDRAM.
To build the bootstrap individually and create sh7670-linux.bin file in the build/sh7670-images directory use the following command in the build directory:

    make bootstrap

Linux Kernel

The SH7670 Linux Kernel has a preconfigured default configuration to ensure a working binary can be built.
To restore this configuration and then configure the Kernel, type the following in the build directory;

    make -f Makefile.linux-2.6.17-sh7670 default_config
    make -f Makefile.linux-2.6.17-sh7670 configure
    make -f Makefile.linux-2.6.17-sh7670
When the new kernel has been built, re-link it into the sh7670-linux.bin by rebuilding the bootstrap as above.

Init and Booting scripts

When the SH7670 Board boots; Linux will execute /sbin/init. This program is contained in the "init" directory to be built and installed by the build scripts.

Also contained in "init" is an rc script file which is executed by init. This script is responsible for calling BusyBox to install on a fresh file system at first boot and setting up ethernet and boot parameters.

Customise this file to change boot parameters or configure the boot process.

BusyBox

BusyBox 1.5.1 has been built to provide the SH7670 with the core functionality of a Linux Console based Operating System. In the SH7670 BSP BusyBox has been split into separate components to improve performance of the running Linux console.

No Shared Library support.

Currently the BusyBox binary executables are not able to share common code. This leads to large run time of applications when performing copy operations to instantiate a new process. One of BusyBox's features is to build all of the executables into a single file. However this adds to the quantity of data to copy on a process creation. To overcome this and improve performance, the BusyBox utilities have been split into several executables containing related utilities. Currently there are four BusyBox categories:
Shellinit   (MSH Shell)
Coreutils   (ls, cp, mv, ...)
netutils   (ifconfig, ping, ...)
miscutils   (insmod, mount, ...)

Fork and Vfork on µClinux:

When adding utilities or changing source on the SH7670, is it important to remember that the fork() system call is not supported, and where possible the vfork() call should be used instead to allow process execution.

When building BusyBox, several options have been built into the makefile to accommodate the split files of the SH BusyBox. The following commands can be issued in the build directory:

To perform a complete build of all of the BusyBox utilities
make -f Makefile.busybox-1.5.1

or to configure and make individual binaries,
make -f Makefile.busybox-1.5.1 coreutils.config
make -f Makefile.busybox-1.5.1 coreutils.busybox

where coreutils is one of:
shellinit coreutils netutils miscutils

Installing on the board:

As part of the boot process provided by /etc/rc, all executable binaries in the /busybox directory will be called with the '--install –s' options at bootup. This facilitates the first time install of the symlinks and keeps them up to date if new binaries are added with more functionality.
When the BusyBox binaries are rebuilt they can be copied directly onto the flash filesystem into /busybox. The boot scripts will update the symlinks on the next power cycle of the board. The boot scripts will not however remove symlinks if applets are removed from the utilities or moved.

Web Front End (WebFE)

The demonstration web front end of the SH7670 RSK+ board is served by a skeletal web server called httpd. The web server is an optional part of a BusyBox build and is configured into the BusyBox object – '/busybox/netutils'. The server is invoked via the symbolic link '/sbin/httpd' at system startup from the configuration file '/etc/rc'. The BusyBox httpd server takes only one parameter - the pathname of a directory to use as the root of its website. The default is '/www'.
 Structure of the demonstration website under /www:

Index.html redirects the web browser to the cgi-bin/renesas.cgi script. This script performs the input parsing, control, and outputs a formatted HTML page.

images/   stores the images for rendering the web interface

cgi-bin/   - hosts programs to be made available for execution by a web browser. Programs may be executables compiled for the SH2 architecture or shell scripts to be executed by /bin/sh. In this demonstration, the cgi-bin directory hosts a single shell script 'renesas.cgi' optimised for execution by the lightweight default shell, /bin/sh.

USB Host and Function Controller

The SH7670 CPU has an internal USB Host and Peripheral Controller Module.
The USB module can only be operated in one mode at a time. To facilitate this, the drivers for the two USB modes have been built as separate loadable modules, and loading them is performed at boot time by the /etc/rc script. This script will default to load the USB Host Module Driver, unless SW3 is held during boot. If SW3 is held when the rc script executes, the board will be set in USB Function Controller mode.

USB Host:

The USB Host Module Driver can be found in the Linux configuration under Device Drivers -> USB Support -> R8A66597 RSK2+ SH7670 Host Controller Support

When a USB memory stick is inserted, the driver will detect it. You are then expected to mount the new device using the script "usbon", then the device will be mounted and available in the '/dev' structure as '/dev/sda1'.
To remove the device unmount it using the "usboff" script, then remove it.
Failure to do this might result in problems with the SCSI driver situated above the host driver.

USB Function Controller:

The USB Function Module Driver can be found in the Linux configuration under Drivers -> USB -> Gadget

The SH7670 RSK+ USB Function Controller driver can be found in the Linux configuration under Device Drivers -> USB Support -> USB Gadget Support -> R8A66597 RSK2+ SH7670 Gadget driver.
The driver can be found in the Linux configuration under Device Drivers -> USB Support -> USB Gadget Support -> R8A66597 RSK2+ SH7670 Gadget driver.
The driver is intended to present the board to a USB host as a Mass Storage Device. This is achieved in combination with the File Storage Gadget (g_file_storage.ko) and the USB function controller driver.
The file that is presented to the USB host can be any filesystem the host accepts, but it is recommended to make a FAT file system to enable compatibility with both Windows and Linux.

To create a new image file of a different size, follow the procedure below:

1)   dd if=/dev/zero of=massfile.img bs=1k count=750

Count can be adjusted to create a file of any size, in this case a 750Kb file will be generated. Make sure the file will fit on the target device.

2)   fdisk massfile.img

Enter the following sequence of commands to the menu.
x   : enter the expert menu.
c   : set the cylinders
1   : This is only a theoretical number as we don't have a physical drive - but one is the easiest number to manage
r   : return to main menu
n   : New partition
p   : Create a Primary Partition
1   : Partition 1
1   : First cylinder is 1
t   : Set the partition type
e   : W95 Fat 16 (LBA)
w   : Write and quit

3)   You can now use this file as a parameter to the 'g_file_storage' gadget. When first accessed Windows should detect that the drive is clean and asks you if you would like to format the drive.

4)   Please see /etc/rc for an example of how the mass storage device is loaded and how the modules are chosen at boot time to decide which mode the board will operate in.

Note:
By default both USB drivers (Host + Function) are pre-built into the BSP, you select which driver to load during the boot process (by pressing SW3). You must set the dip-switches (SW4) prior to booting the device, when switching the USB mode.

Known Observations

The Web Front End is not as responsive as the button response time. There is a great deal more processing being performed, through scripting and the /bin/sh shell. This could be improved in the future by taking a single binary approach, or solving the shared library issues.

The Frame Buffer Viewer application (FBV) uses 'mmap' to obtain a pointer to the frame buffer. This call has known limitations on the µClinux platform and so the FBV passes flags to the kernel that are unexpected. The kernel responds to this by outputting a "BUG" warning from nommu.c. This is harmless and can be ignored or hidden by selecting to hide BUG warnings in the kernel configuration.

The USB Pen Drive is not always loaded completely. It may be necessary upon insertion, to remove and re-insert the pen drive if it does not mount successfully. The drive is successfully mounted when the 'Web Front End' shows the "Unmount" option.

The MAC Address must be valid for the Web Application to function. By default the board is programmed with a valid MAC address.