WIND RIVER

WR Linux 4.3 BCM956340 Test Report

VERSION 1.2

VERSION DATE FEB 9, 2015

WR Linux 4.3 BCM956340 Test Report

Version 1.1
Feb 9, 2015
Prepared For Huawei Corporation

Wind River Systems, Inc.
9855 Scranton Rd., Building 5
San Diego, CA 92121
858-824-3100 phone
858-824-3198 fax
www.windriver.com

REVISION HISTORY

Date	Version	By	Description of Change
3/26/2014	1.0	Zongjun Li/Guanghui Wang	Created
9/11/2014	1.1	Xiaozhen Chen/Guanghui Wang	Updated the release with RCPL26 and other patches. Added Section 8 Included Changes and Section 9 Known Issues
2/9/2015	1.2	Xiaozhen Chen/Guanghui Wang	Updated the release with RCPL28 and Remove 0011-ARM-7099-1-futex-preserve-oldval-in-SMP-__futex_atom.patch

Table of Contents

1 Introduction

1.1 Purpose and Scope

As part of its development effort, Wind River Services is creating a Wind River Linux 4.3 BSP for the BCM956340 reference board. To validate that effort, a series of acceptance tests will be performed. The purpose of this document is to enumerate the test cases, providing a description of the feature being tested, the test procedure, and the expected results.

As the tests are run, the output (evidence) will be collected into the BSP test report.

General information on configuring the hardware is not included here, but can be found in the BSP readme file, and the platform reference guide.

2 Applicable Documents

The following documents were referenced for the creation of this test plan:

No.	Document	Ver	Scope
1.			
2.			
3.			

3 Glossary of Terms

The table below represents an alphabetical list of abbreviations, acronyms, and terminology commonly used in reference to this product.

Term	Definition
UART	Universal Asynchronous Receiver-Transmitter (Serial Interface)
USB	Universal Serial Bus
I2C	Inter—Integrated Circuit

4 General Assumptions

The following assumptions were made as a part of the WR Linux 4.3 BCM956340 BSP acceptance test plan detailed in this document:

1) Wind River uses good faith efforts to provide efficient drivers but cannot guarantee a given performance.
2) Tests in the plan are run independently and no order of testing is implied.

5 Test Overview

5.1 Summary of Testing

The following table summarizes all of the tests to be performed.

Test	Description
Build (Small FS)	Tests that the BSP with small FS can be built correctly using the Wind River tools
Boot Login(Small FS)	Tests that the target can be booted up with the combination of kernel and small FS configurations
Build (Std FS)	Tests that the BSP with std FS can be built correctly using the Wind River tools
Boot Login(Std FS)	Tests that the target can be booted up with the combination of kernel and std FS configurations
UART	Tests that the BSP supports the UART ports
Ethernet	Tests that the BSP supports the gigabit Ethernet ports
I2C	Tests that the BSP supports the I2C bus and the connected I2C device
SPI NOR FLASH	Tests that the BSP supports the QSPI and the connected SPI NOR Flash
NAND Flash	Tests that the BSP supports the NAND Flash
GPIO	Tests that the BSP supports the GPIO ports
USB host	Tests that the BSP supports the USB host
PCIe Bus	Tests that the BSP correctly enumerates the PCIe bus and interfaces with addon cards plugged into the available slots

6 Test Procedure

6.1 Build (small FS)

Perform the following tests to verify that BSP could be built correctly

No.	Test	Expected Result
6.1.1	In an empty directory, install the BSP layer	
6.1.2	Using Wind River Workbench, configure a project using the following options: --with-layer=<path to BSP layer> --enable-board=bcm_helix4 --enable-kernel=standard --enable-rootfs=glibc_small --enable-bootimage=flash --with-rcpl-version=0028	BSP configure completes correctly
6.1.3	Using the project configured in the previous step, do a complete build from source: make all	BSP build completes correctly
6.1.4	Create another project directory and run the following command line to configure the BSP <WR_LINUX_INSTALL_DIR>/wrlinux-4/wrlinux/configure \ --with-layer=<path to BSP layer>\ --enable-board= bcm_helix4\ --enable-kernel=standard \ --enable-rootfs=glibc_small \ --enable-bootimage=flash \ --with-rcpl-version=0028	BSP configure completes correctly
6.1.5	Using the project configured in the previous step, do a complete build from source: make all	BSP build completes correctly
6.1.6	$PRJ/build/linux/scripts/mkuboot.sh -A arm -O linux -T kernel -C none -a 0x60008000 -e 0x60008000 -n 'Linux-2.6.34.15' -d $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/Image $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/uImage	A uImage is generated in $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/
6.1.7	Following the build-all, run the following to build a ramdisk file system make boot-image BOOTIMAGE_TYPE=flash BOOTIMAGE_RAM0SIZE=131072	$PRJ/export directory contains the newly built bcm_helix4-initrd.gz.uboot

6.2 Boot Login(small FS)

Perform the following tests to verify that reference platform boots correctly with BSP. Boot methods include ramdisk and NFS.

No.	Test	Expected Result
6.2.1	Setup TFTP server on the host system. Copy the export/bcm_helix4-uImage-WR4.3.0.0_standard file to the tftpboot directory and change its name to uImage. Copy the export/ bcm_helix4-initrd.gz.uboot file to the tftpboot directory and change its name to ramdisk	
6.2.2	In the uboot, setup ramdisk boot, for example, Setenv bootargs 'root=/dev/ram rw console=ttyS0,115200’	
6.2.3	Power on the target system. In the uboot,input the commands as following: tftp 0x62000000 uImage tftp 0x65000000 ramdisk bootm 62000000 65000000	Board should download kernel, ramdisk from the host. Kernel will boot up.
6.2.4	Set up an empty directory on the host for use as an NFS file system	
6.2.5	Unarchive the file system into the directory using the file export/bcm_helix4-standard-glibc_small-dist.tar.bz2	
6.2.6	Setup the host NFS export file and daemons	
6.2.7	In the uboot, setup NFS boot, for example, setenv bootargs 'root=/dev/nfs rw, nfsroot=128.224.156.74:/opt/nfs/arm ip=dhcp console=ttyS0,115200’	
6.2.8	Power on the target system. In the uboot,input the commands as following: tftp 0x62000000 bootm 62000000;	Board should download kernel from the tftp server, and then use the NFS file system as the root FS. Kernel will boot up

6.3 Build (std FS)

Perform the following tests to verify that BSP could be built correctly

No.	Test	Expected Result
6.3.1	In a empty directory, install the BSP layer	
6.3.2	Using Wind River Workbench, configure a project using the following options: --with-layer=<path to BSP layer> --enable-board=bcm_helix4 --enable-kernel=standard --enable-rootfs=glibc_std --enable-bootimage=flash --with-rcpl-version=0028	BSP configure completes correctly
6.3.3	Using the project configured in the previous step, do a complete build from source: make all	BSP build completes correctly
6.3.4	Create another project directory and run the following command line to configure the BSP <WR_LINUX_INSTALL_DIR>/wrlinux-4/wrlinux/configure \ --with-layer=<path to BSP layer>\ --enable-board= bcm_helix4\ --enable-kernel=standard \ --enable-rootfs=glibc_std \ --enable-bootimage=flash \ --with-rcpl-version=0028	BSP configure completes correctly
6.3.5	Using the project configured in the previous step, do a complete build from source: make all	BSP build completes correctly
6.3.6	Run make –C build linux.menuconfig and change CONFIG_AUDIT to be disabled. Then make –C build linux.rebuild; mak fs; $PRJ/build/linux/scripts/mkuboot.sh -A arm -O linux -T kernel -C none -a 0x60008000 -e 0x60008000 -n 'Linux-2.6.34.15' -d $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/Image $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/uImage	A uImage is generated in $PRJ/build/linux-bcm_helix4-standard-build/arch/arm/boot/

6.4 Boot Login(std FS)

Perform the following tests to verify that reference platform boots correctly with BSP. Since the std FS image is very large, we use it via NFS.

No.	Test	Expected Result
6.4.1	Setup TFTP server on the host system. Copy the export/bcm_helix4-uImage-WR4.3.0.0_standard file to the tftpboot directory and change its name to uImage.	
6.4.2	Set up an empty directory on the host for use as an NFS file system	
6.4.3	Unarchive the file system into the directory using the file export/bcm_helix4-standard-glibc_std-dist.tar.bz2	
6.4.4	Setup the host NFS export file and daemons	
6.4.5	In the uboot, setup NFS boot, for example, setenv bootargs 'root=/dev/nfs rw, nfsroot=128.224.156.74:/opt/nfs/arm ip=dhcp console=ttyS0,115200’	
6.4.6	Power on the target system. In the uboot, input the commands as following: tftp 0x62000000 bootm 62000000;	Board should download kernel from the tftp server, and then use the NFS file system as the root FS. Kernel will boot up

6.5 UART

Perform the following tests to verify that UART ports operate correctly:

No.	Test	Expected Result
6.5.1	Connect a serial cable between the host and the Serengeti board UART0. In the Uboot,using console=ttyS0,${baudrate} to boot the kernel	Using UART0, the kernel can be loged in.
6.5.2	Connect a serial cable between the host and the Serengeti board UART0. In the Uboot,using console=ttyS1,${baudrate} to boot the kernel Connect the serial cable between the host and the Serengeti board UART1.	Using UART1, the kernel can be loged in.

6.6 Ethernet

Perform the following tests to verify that reference platform performs correctly:

No.	Test	Expected Result
6.6.1	Connect appropriate cable between host and the eth0 on the target.	
6.6.2	Configure eth0 with an appropriate IP address	
6.6.3	Run “ping <host ip> -s 65500”	No packet loss
6.6.4	From host, run “ping <target ip> -s 65500”	No packet loss
6.6.5	Connect appropriate cable between host and the eth1 on the target.	
6.6.6	Configure eth1 with an appropriate IP address And stop eth0.	
6.6.7	Run “ping <host ip> -s 65500”	No packet loss
6.6.8	From host, run “ping <target ip> -s 65500”	No packet loss

6.7 I2C

Perform the following tests to verify that reference platform supports I2C correctly.

6.7.1	Write a test program for I2c READ. For example:	EEPROM(device address is 0x50) connected to the I2C bus can be read

```
int i2c_read(int fd, int offset, int *value)
{
    int ret = 0;
    struct i2c_smbus_ioctl_data ioctl_data = {0};
    union i2c_smbus_data data;

    /* read from offset */
    data.byte = 0;
    ioctl_data.read_write = I2C_SMBUS_READ;
    ioctl_data.command = offset;
    ioctl_data.size = I2C_SMBUS_BYTE_DATA;
    ioctl_data.data= &data;

    ret = ioctl(fd, I2C_SMBUS, &ioctl_data);
    if (ret < 0)
    {
        printf("read, I2C_SMBUS failed: %d.\n", errno);
    }
    else
    {
        *value = data.byte;
    }

    return ret;
}
```

6.7.2	Write a test program for I2c write. For example:	EEPROM(device address is 0x50) connected to the I2C bus can be written

```
int i2c_write(int fd, int offset, int value)
{
    int ret = 0;
    struct i2c_smbus_ioctl_data ioctl_data = {0};
    union i2c_smbus_data data;

    /* write at offset with value */
    data.byte = value;
    ioctl_data.read_write = I2C_SMBUS_WRITE;
    ioctl_data.command = offset;
    ioctl_data.size = I2C_SMBUS_BYTE_DATA;
    ioctl_data.data= &data;

    ret = ioctl(fd, I2C_SMBUS, &ioctl_data);
    if (ret < 0)
    {
        printf("write, I2C_SMBUS failed: %d.\n", errno);
    }

    return ret;
}
```

6.7.3	Wrtie a value to the EEPROM and then read out	The value read out equals the value written to the EEPROM
6.7.4		
6.7.5	Write the EEPROM by PROC fs. For example:	

```
cd /proc/iproc-i2c/iproc-i2c0
# echo 0 80 32 49 > iproc-i2c-dbg

Command can execute slow, please wait...

Write OK.
Wrote 0x31 at addr 32
```

6.7.6	Read the EEPROM by PROC fs. For example: # echo 1 80 32 0 > iproc-i2c-dbg Command can execute slow, please wait... Read OK. --------Value read at 32 = 0x31	The value read out equals the value written to the EEPROM

6.8 SPI NOR Flash

Perform the following tests to verify that reference platform supports SPI NOR flash correctly and the SPI works correctly.

6.8.1	Write a small file to the mtd, For example, dd if=./a.txt of=/dev/mtdblock1 bs=1024 count=4	Write successfully
6.8.2	Read the file from the mtd, For example, dd if=/dev/mtdblock1 of=./b.txt bs=1024 count=4	Read successfully
6.8.3	Compare a.txt and b.txt	b.txt is as same as a.txt
6.8.4	Write a big file to the mtd, For example, dd if=./c.txt of=/dev/mtdblock1 bs=1024 count=64	Write successfully
6.8.5	Read the file from the mtd, For example, dd if=/dev/mtdblock1 of=./d.txt bs=1024 count=64	Read successfully
6.8.6	Compare c.txt and d.txt	d.txt is as same as c.txt

6.9 NAND flash

Perform the following tests to verify that reference platform supports NAND flash correctly:

No.	Test	Expected Result
6.9.1	Write a small file to the mtd, For example, dd if=./a.txt of=/dev/mtdblock7 bs=1024 count=4	Write successfully
6.9.2	Read the file from the mtd, For example, dd if=/dev/mtdblock7 of=./b.txt bs=1024 count=4	Read successfully
6.9.3	Compare a.txt and b.txt	b.txt is as same.as a.txt
6.9.4	Write a big file to the mtd, For example, dd if=./c.txt of=/dev/mtdblock7 bs=1024 count=3072	Write successfully
6.9.5	Read the file from the mtd, For example, dd if=/dev/mtdblock7 of=./d.txt bs=1024 count=3072	Read successfully
6.9.6	Compare c.txt and d.txt	d.txt is as same.as c.txt

6.10 GPIO

Perform the following tests to verify that reference platform supports GPIO correctly.

6.10.1	Using Sysfs to verify gpio.For example: cd /sys/class/gpio/ echo 6 > export	
6.10.2	Set gpio direction, for example: cd gpio6 echo out > direction	
6.10.3	Read gpio direction, for example: cat direction	The direction showed by 'cat ' command is same as what you wanted.
6.10.4	Set gpio value, for example: echo 1 > value	
6.10.5	Read gpio value, for example: cat value	The value showed by 'cat ' command is same as what you wanted.
6.10.6	Set gpio value, for example: echo 0 > value	
6.10.7	Read gpio value, for example: cat value	The value showed by 'cat ' command is same as what you wanted.

6.11 USB Host

Perform the following tests to verify that reference platform performs correctly:

No.	Test	Expected Result
6.11.1	Insert a USB disk into USB(host)	
6.11.2	Run "mount /dev/sda1 /mnt/usb1"	Files in the usb disk can be seen..
6.11.3	Do some read/write operations on the usb disk and verify that the operations are successful	The operations are successful
6.11.4	Run "umount /dev/sda1"	'Umount' operation is succeesful.

6.12 PCIE

Perform the following tests to verify that PCI bus is enumerated, and appropriate drivers are loaded, and data flows to and from PCI bus are right. Because there is no suitable slot on BCM956340 reference board for us to insert a network card, we assume that on HUAWEI's board, we can insert a PCIe NIC, and the following tests can be executed on HUAEI's board.

No.	Test	Expected Result
6.12.1	Shutdown and power off target system	
6.12.2	Place a PCIe network card in empty PCIe slot.	
6.12.3	Power on and boot target	
6.12.4	Run “ifconfig –a”	Additional Ethernet devices should be listed
6.12.5	Connect appropriate cable between host and target board PCIe network card port	
6.12.6	Configure ethX with appropriate IP address	
6.12.7	Run “ping <host ip> -s 65500”	No packet loss

7 Test Report

7.1 Test Result Matrix

Discussion of test configuration

Test	Name	Result	Comments
6.1	Build (small FS)	Successful	
0	Boot Login(small FS)	Successful	
0	Build (std FS)	Successful	
0	Boot Login(std FS)	Successful	
6.5	UART	Successful	
0	Ethernet	Successful	
0	I2C	Successful	
6.8	SPI NOR FLASH	Successful	
6.9	NAND flash	Successful	
0	GPIO	Successful	
6.11	USB Host	Successful	
0	PCIE	Not Applicable	There is no suitable slot in the reference board, so the tests should be executed with HuaWei's board.

8 Included Changes

Accumulated changes compared to last release:

1. Updated with RCPL28.
2. 0011-ARM-7099-1-futex-preserve-oldval-in-SMP-__futex_atom.patch is removed from our layer because this patch has been included in WRLinux RCPL28.

9 Known Issues

There is no known issues.