NXP Semiconductors

Document Number:
LS1043A/LS1046A ASK ASK
7.0.0

Release Notes

LS1043A/LS1046A ASK 7.0.0

Contents

1. Introduction

This document provides Release Notes for the LS1043/LS1046 ASK Software Package.

1.1. Purpose and scope

This release allows building of the LS1043A/LS1046A ASK and loading of the LS1043A/LS1046A binaries on RDB and RGW platforms.
This includes following:
Code release

- Sources downloaded from Open Source web sites
- LS1043A/LS1046A Openwrt ASK source patches and
  Makefiles

Reference release binaries

- Linux BSP for LS1043A/LS1046A
- Reset Configuration Word (RCW).
- Boot loader code (uBoot)
- Ramfs file systems Integrated with kernel image (.itb).
- JFFS2 file system bootable from NOR Flash for LS1043A RDB.
- UBIFS file system bootable from NOR Flash for LS1043A RDB and
- UBIFS file system bootable from NAND flash for LS1043A RGW

Available for LS1043A RDB/RGW platform and LS1046 RDB platform

1.2 Release Highlights

This release is similar to ASK 1.2.1 feature wise, except Multicast and NAT-PT, with the added feature of supporting unlimited connections and improved performance.
The Openwrt and Kernel versions used are as below:

- LS1043A/LS1046A BSP support on Kernel Version 4.1.35 synced with SDK version 1703
- Latest Openwrt version used is Chaos Calmer

1.1.1. New in this release

1. More than 50K connections (Complete External Hash)
2. IpSec (IPv4 and IPv6) offload
3. OpenSSL (Crypto only) offload
4. Improved performance
5. DPAA firmware integrated with Linux kernel image. (i.e Flashing of DPAA firmware is not required. It automatically gets loaded with uimage or itb)

2. Release notes for boot loader

This section contains the release notes for boot loader.

2.1. U-Boot

The U-Boot code basis is v2016.09. The following LS1043A/LS1046A components are part of U-Boot:

- I2C
- QSPI
- PCIe
- Ethernet Driver for DPAA.
- USB 3.0
- USB 2.0
- Alternate Bank support
- Linux Boot using SD card, TFTP, Y-modem
- Linux boot from QSPI

3. Release notes for Linux BSP

The Linux code basis for this release is v4.1.35. The following LS1043A/LS1046A components are part of this release.

- BSP for Linux, with following support
  - PLL/Clock and board specific initializations.
  - Interrupt controller.
  - DUART Driver.
  - Ethernet Driver for DPAA(FMAN).
  - CMM, FCI, and CDX control driver support.
  - PCIe Host Driver.
  - QSPI Driver
  - PCIe WiFi driver support (ath9k) for QCA Mini PCIe Module.
  - PCIe WiFi driver support Quantena Mini PCIe Module.
  - Jffs2 file system support for LS1043 RDB.
  - UBIFS file system support for LS1043 RDB and RGW.

4. Release notes for fast path

The following components are part of this release:

- CMM application for fast path control
- Fast path support for IPv4, IPv6, IPv4/NAT
- PPPoE
- VLAN
- Q-in-Q
- QoS for egress
- L2 Bridging (Automatic Mode)
- WiFi Fast Path
- IPSEC
- Tunnels (4o6, 6o4)
- Statistics for connections
- Interface statistics
- Supports more than 50K connections
- DPAA Firmware is integrated with Linux image.

5. Release notes for Openwrt ASK

The ASK is based on Openwrt Chaos Calmer 15.05 branch, and is prepared with the following toolchain, library, and binutils:

- GCC 4.8-linaro arm
- Glibc 2.19
- Binutils 2.24.3

The following packages are part of the ASK:

- Busybox 1.23.2
- Cmm
- Cdx
- Dpa_app
- Dnsmasq 2.73 (DHCP and DNS server)
- Dropbear 2015.67
- e2fsprogs_1.42.12
- Ethtool 3.7
- Fci
- Fmc
- fmlib
- fdisk_2.25.2-4
- Fstools_2015-05-24
- haserl_0.9.32
- hostapd_2015-03-25-1
- i2c-tools 3.1.2
- ip 4.0
- Iptables 1.4.21
- Ip6tables 1.4.21
- jshn 2015-06-14
- jsonfilter 2014-06-19
- libcli 1.9.4
- libgcc_4.8-linaro
- libgdbm_1.10

- libip4tc_1.4.21
- libip6tc_1.4.21
- libjson-c_0.12
- libjson-script_2015-06-14
- libmnl_1.0.3-2
- libnetfilter-conntrack_1.0.4
- libnl-tiny_0.1
- libpcap_1.5.3
- libpthread_2.19
- librt_2.19
- libstdcpp_4.8-linaro
- libubox_2015-06-14
- libubus_2015-05-25
- libuci_2015-04-09
- libxml2_2.9.2
- libxtables_1.4.21
- mbedtls-1.3.12
- mmiotool_1.00
- mtd-utils_1.5.0
- mount-utils
- net-tools-mii-tool_1.60
- Netifd-2015-08-25
- odhcp6c_2015-07-13
- odhcpd_2015-09-01
- perf_4.1.35
- ppp_2.4.7
- procd_2015-08-16
- sysstat_10.1.2
- strace_4.8-1
- tcpdump_4.5.1-4
- trace-cmd_v2.4.2-1
- uci_2015-08-27
- ubox_2015-07-14
- ubus_2015-05-25
- ubi-utils_1.5.1-2
- vsftpd_3.0.2
- web
- zlib_1.2.8

6. Building LS1043A/LS1046A OpenWrt ASK

6.1. LS1043A-RDB

To build the image using source released and the dl tar balls provided in the CR release directory, follow the below steps:

1. Checkout the release src and dl tar balls from CR folder in the release directory
2. Un-tar the src-openwrt-ls1043a_7.0.0.tar.bz2

   ➢ **`tar -jxvf src-openwrt-ls1043a_7.0.0.tar.bz2`**

3. Un-tar the dl-openwrt-ls1043a_7.0.0.tar.bz2

   ➢ **`tar -jxvf dl-openwrt-ls1043a_7.0.0.tar.bz2`**

4. Access the src directory src-openwrt-ls1043a_7.0.0
5. Create the soft link to the dl directory

   - **`ln -s ../dl dl`**

6. Copy the RDB build config file `config-ls1043a-rdb` from config directory

   - **`cp config/config-ls1043a-rdb .config`**

7. Execute the `make` command to compile the build to generate the image

   - **`make V=99`** (to see the full compilation logs) or `make` (without compilation logs)

After successful compilation, you can find the following useful images

1. **openwrt-ls1043a-hgw-lsrdb.itb** is the ITB image for Linux in bin/ls1043a-glibc directory
2. **u-boot-ls1043ardb.bin** is the u-boot image generated in bin/ls1043a-glibc/uboot-layerscape-ls1043ardbnor
3. **rcw_1600_gic4k.bin** is the default rcw image generated in bin/ls1043a-glibc/rcw/ls1043ardb/ RR_FQPP_1455
4. **openwrt-ls1043a-root.ubi_nor** is the ubifs root filesystem image in bin/ls1043a-glibc directory
5. **openwrt-ls1043a-root.jffs2-128k** is the jffs2 root filesystem image in bin/ls1043a-glibc directory
6. **openwrt-ls1043a-hgw-lsrdb-uImage** is the Linux Kernel Image generated in bin/ls1043a-glibc directory
7. **openwrt-ls1043a-hgw-lsrdb.dtb** is the dtb image generated in bin/ls1043a-glibc directory
8. **fsl_fman_ucode_r2.bin** is the firmware image generated in bin/ls1043a-glibc directory
9. **ls1043_r2.h** is the firmware header file generated in bin/ls1043a-glibc directory to integrate with Linux image

6.2. LS1043A-RGW

To build the image using source released and the dl tar balls provided in the CR release directory, follow the below steps:

1. Checkout the release src and dl tar balls from CR folder in the release directory
2. Un-tar the src-openwrt-ls1043a_7.0.0.tar.bz2

   - **`tar -jxvf src-openwrt-ls1043a_7.0.0.tar.bz2`**

3. Un-tar the dl-openwrt-ls1043a_7.0.0.tar.bz2

   - **`tar -jxvf dl-openwrt-ls1043a_7.0.0.tar.bz2`**

4. Access the src directory src-openwrt-ls1043a_7.0.0
5. Create the soft link to the dl directory

   - **`ln -s ../dl dl`**

6. Copy the RDB build config file `config-ls1043a-rdb` from config directory

   - **`cp config/config-ls1043a-rgw .config`**

7. Execute the `make` command to compile the build to generate the image

   - **`make V=99`** (to see the full compilation logs) or `make` (without compilation logs)

After successful compilation, you can find the following useful images

1. **openwrt-ls1043a-hgw-lsrgw.itb** is the ITB image for Linux in bin/ls1043a-glibc directory
2. **u-boot-ls1043ardb.bin** is the u-boot image generated in bin/ls1043a-glibc/uboot-layerscape-ls1043argw
3. **openwrt-ls1043a-root.ubi_nand** is the ubifs root filesystem image in bin/ls1043a-glibc directory
4. **openwrt-ls1043a-hgw-lsrgw-uImage** is the Linux Kernel Image generated in bin/ls1043a-glibc directory
5. **openwrt-ls1043a-hgw-lsrgw.dtb** is the dtb image generated in bin/ls1043a-glibc directory
6. **fsl_fman_ucode_r2.bin** is the firmware image generated in bin/ls1043a-glibc directory
7. **ls1043_r2.h** is the firmware header file generated in bin/ls1043a-glibc directory to integrate with Linux image

6.3. LS1046A-RDB

To build the image using source released and the dl tar balls provided in the CR release directory, follow the below steps:

1. Checkout the release src and dl tar balls from CR folder in the release directory
2. Un-tar the src-openwrt-ls1046a_7.0.0.tar.bz2

   - **`tar -jxvf src-openwrt-ls1046a_7.0.0.tar.bz2`**

3. Un-tar the dl-openwrt-ls1046a_7.0.0.tar.bz2

   - **`tar -jxvf dl-openwrt-ls1046a_7.0.0.tar.bz2`**

4. Access the src directory src-openwrt-ls1046a_7.0.0
5. Create the soft link to the dl directory

   - **`ln -s ../dl dl`**

6. Copy the RDB build config file `config-ls1046a-rdb` from config directory

   - **`cp config/config-ls1046a-rdb .config`**

7. Execute the `make` command to compile the build to generate the image

   - **`make V=99`** (to see the full compilation logs) or `make` (without compilation logs)

After successful compilation, you can find the following useful images

1. **openwrt-ls1046a-hgw-lsrdb.itb** is the ITB image for Linux in bin/ls1046a-glibc directory
2. **u-boot-ls1046ardb.bin** is the u-boot image generated in bin/ls1043a-glibc/uboot-layerscape-ls1046ardb
3. **rcw_1600_qspiboot_swap.bin** is the default rcw image generated in bin/ls1046a-glibc/rcw/ls1046ardb/RR_FFSSPPPH_1133_5559
4. **fsl_fman_ucode_r2.bin** is the firmware image generated in bin/ls1046a-glibc directory
5. **ls1043_r2.h** is the firmware header file generated in bin/ls1046a-glibc directory to integrate with Linux image

7. Release images

The release contains the following images.

7.1. LS1043ARDB

- CR contains the src and dl tar balls.
    - src-openwrt-ls1043a_7.0.0.tar.bz2
    - dl-openwrt-ls1043a_7.0.0.tar.bz2
- RSR contains the following release images:
    1. itb-openwrt-ls1043a_7.0.0-ls1043a-rdb
    2. openwrt-rootfs-ls1043a_7.0.0-ls1043a-rdb.cpio.gz
    3. openwrt-rootfs-ls1043a_7.0.0-ls1043a-rdb.tar.gz
    4. u-boot-2016.01-v201601.02.1-ls1043a-rdb.bin
    5. uImage-openwrt-ls1043a_7.0.0-ls1043a-rdb
    6. dtb-openwrt-ls1043a_7.0.0-ls1043a-rdb
    7. ubi-nor-openwrt-ls1043a_7.0.0-ls1043a-rdb
    8. ubi-nand-openwrt-ls1043a_7.0.0-ls1043a-rdb
    9. jffs2-128k-openwrt-ls1043a_7.0.0-ls1043a-rdb
    10. fsl_fman_ucode_r2.bin
    11. ls1043_r2.h
    12. Inside rcw/ls1043ardb/RR_FQPP_1455 directory,
        rcw_1200.bin
        rcw_1500_qetdm.bin
        rcw_1600_gic4k.bin (Default)
        rcw_uefi_1400.bin
        rcw_1400.bin
        rcw_1500_sben.bin
        rcw_1600_qetdm.bin
        rcw_uefi_1500.bin
        rcw_1500.bin
        rcw_1600.bin
        rcw_1600_sben.bin
        rcw_uefi_1600.bin

7.2. LS1043ARGW

- CR contains the src and dl tar balls.
    - src-openwrt-ls1043a_7.0.0.tar.bz2
    - dl-openwrt-ls1043a_7.0.0.tar.bz2
- RSR contains the following release images:
    1. itb-openwrt-ls1043a_7.0.0-ls1043a-rgw
    2. openwrt-rootfs-ls1043a_7.0.0-ls1043a-rgw.cpio.gz
    3. openwrt-rootfs-ls1043a_7.0.0-ls1043a-rgw.tar.gz
    4. u-boot-2016.01-v201601.02.1-ls1043a-rgw.bin
    5. uImage-openwrt-ls1043a_7.0.0-ls1043a-rgw
    6. dtb-openwrt-ls1043a_7.0.0-ls1043a-rgw
    7. ubi-nand-openwrt-ls1043a_7.0.0-ls1043a-rgw
    8. fsl_fman_ucode_r2.bin
    9. ls1043_r2.h

7.3. LS1046ARDB

- CR contains the src and dl tar balls.
    - src-openwrt-ls1046a_7.0.0.tar.bz2
    - dl-openwrt-ls1046a_7.0.0.tar.bz2
- RSR contains the following release images:
    1. itb-openwrt-ls1046a_7.0.0-ls1046a-rdb
    2. u-boot-2016.01-v201601.02.1-ls1046a-rdb.bin
    3. jffs2-128k-openwrt-ls1046a_7.0.0-ls1046a-rdb
    4. fsl_fman_ucode_r2.bin
    5. ls1043_r2.h
    6. Inside rcw/ls1046ardb/RR_FFSSPPPH_1133_5559 directory,
       rcw_1600_qspiboot_swap.bin (default)
       rcw_1800_qspiboot_swap.bin

8. Guidelines for flashing or updating flash partitions

8.1. LS1043A-RDB

8.1.1. NOR Flash Layout

Start Address	*End Address*	*Image*	*Size*
0x060000000	*0x0600FFFFF*	*RCW + PBI*	*1MB*
0x060100000	*0x0601FFFFF*	*U Boot*	*1MB*
0x060200000	*0x0602FFFFF*	*U Boot Env*	*1MB*
0x060300000	*0x0610FFFFF*	*FMan ucode*	*14MB*
0x061100000	*0x61EFFFFF*	*Kernel ITB*	*47MB*

8.1.2. U-boot Environment Variables

The following is the **default u-boot** environment configuration

baudrate=115200
bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200
bootcmd=cp.b $kernel_start $kernel_load $kernel_size && bootm $kernel_load
bootdelay=10
console=ttyAMA0,38400n8
eth1addr=00:04:9F:03:D3:14
eth2addr=00:04:9F:03:D3:15
eth3addr=00:04:9F:03:D3:16
eth4addr=00:04:9F:03:D3:17
eth5addr=00:04:9F:03:D3:18
eth6addr=00:04:9F:03:D3:19
ethact=FM1@DTSEC1
ethaddr=00:04:9F:03:D3:13
ethprime=e1000#0

fdt_high=0xffffffffffffffff
fdtcontroladdr=ffbc7738
fileaddr=a0000000
filesize=c48a97
fman_ucode=60300000
hwconfig=fsl_ddr:bank_intlv=auto
initrd_high=0xffffffffffffffff
kernel_addr=0x100000
kernel_load=0xa0000000
kernel_size=0x2800000
kernel_start=0x61100000
loadaddr=0x80100000
mtdids=nor0=60000000.nor
ramdisk_addr=0x800000
ramdisk_size=0x2000000
scsidevs=0
stderr=serial
stdin=serial
stdout=serial

NOTE-I: *Default env setting sets bootargs for RAM based file system*.

8.1.3. Changing the Env Variable

Any env variable can be changed using following command:

setenv <Variable Name> <Variable Value>

Ex:

setenv bootargs ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200

8.1.4. Set the IP address

Change the IP address (in environment variable) using following command:

=>setenv serverip 192.168.5.124; setenv ipaddr 192.168.5.136;setenv ethact FM1@DTSEC1

Connect the ethernet cable to DTSEC1 port of LS1043A RDB to your Local PC

8.1.5. Flashing & Booting pre-built RamDisk ASK Images using TFTP

NOTE-1: *This section only explains the example of flashing RAM based filesystem. For Flash Based file system, please refer section JFFS2 Read-Write File System , Flash & Boot pre-built JFFS2 ASK Images using TFTP ,UBIFS Read-Write File System and Flashing & Booting pre-built UBIFS ASK Images using TFTP*

NOTE-2: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB .*

8.1.5.1. Flash RCW

```
=> tftp 0xa0000000 rcw_1600_gic4k.bin
=> protect off 60000000 +$filesize
=> erase 60000000 +$filesize
=> cp.b a0000000 60000000 $filesize
=> protect on 60000000 +$filesize
```

8.1.5.2. Flash U-boot

```
=> tftp 0xa0000000 u-boot-2016.01-v201601.02.1-ls1043a-rdb.bin
=> protect off 60100000 +$filesize
=> erase 60100000 +$filesize
=> cp.b a0000000 60100000 $filesize
=> protect on 60100000 +$filesize
```

8.1.5.3. Flash DPAA firmware

```
=> tftp 0xa0000000 fsl_fman_ucode_r2.bin
=> protect off $fman_ucode +$filesize
=> erase $fman_ucode +$filesize
=> cp.b a0000000 $fman_ucode $filesize
=> protect on $fman_ucode +$filesize
```

NOTE-1: -- *This step is not required if there is a valid firmware exists in flash. The 7.0.0 firmware is automatically loaded with Linux kernel.*

8.1.5.4. Flash Linux

```
=> tftp 0xa0000000 itb-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> erase $kernel_start +$filesize
=> cp.b 0xa0000000 $kernel_start $filesize
```

8.1.5.5. Booting Linux with auto-boot

Follow following steps to boot using auto boot:

1. Make sure all required image are flashed on the board-using step 8.1.5.1 to 8.1.5.4.
2. Make sure following env setting is saved:
   bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200
   bootcmd=cp.b $kernel_start $kernel_load $kernel_size && bootm $kernel_load

3. Power cycle the board to boot it until the boot prompt.

8.1.5.6. Booting Linux without auto-boot

Follow following steps to boot Linux manually (this will be required if user wants to change the itb image and boot):

1. Press enter at u-boot prompt to stop autoboot.
2. Make sure following env setting is saved:

bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200
bootcmd=cp.b $kernel_start $kernel_load $kernel_size && bootm $kernel_load

3. Load new itb image to DDR using TFTP
   => tftp $kernel_load itb-openwrt-ls1043a_7.0.0-ls1043a-rdb
4. Boot Linux
   =>bootm $kernel_load

8.1.6. JFFS2 Read-Write File System

8.1.6.1. NOR Flash Layout for JFFS2 File System

For the Usage of JFFS2 Filesystem,the following is the NOR Flash Layout:

Start Address	*End Address*	*Image*	*Size*
0x060000000	*0x0600FFFFF*	*RCW + PBI*	*1MB*
0x060100000	*0x0601FFFFF*	*U Boot*	*1MB*
0x060200000	*0x0602FFFFF*	*U Boot Env*	*1MB*
0x060300000	*0x0610FFFFF*	*FMan ucode*	*14MB*
0x061100000	*0x61EFFFFF*	*uImage(Kernel)*	*14MB*
0x061F00000	*0x61FFFFFF*	*DTB*	*1MB*
0x062000000	*0x063FF_FFFF*	*Rootfs*	*32MB*

8.1.7. Flash & Boot pre-built JFFS2 ASK Images using TFTP

JFFS2 is the read/write file system that can used on QSPI Flash.

NOTE-1: *This section only explains the example of flashing JFFS2 based filesystem. For RAM Based file system, please refer Flashing & Booting pre-built RamDisk ASK Images using TFTP*

NOTE-2: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB .*

8.1.7.1. Flash RCW

Refer Flash RCW for flashing RCW if not flashed already.

8.1.7.2. Flash U-boot

Refer Flash U-boot for flashing u-boot if not flashed already.

8.1.7.3. Flash DPAA Firmware

Refer Flash DPAA firmware if not flashed.

8.1.7.4. Flash uImage, DTB, Rootfs

Flash uImage :

=> tftp 0xa0000000 uImage-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x61100000 +$filesize
=> erase 0x61100000 +$filesize
=> cp.b 0xa0000000 0x61100000 $filesize
=> protect on 0x61100000 +$filesize

Flash Rootfs :

=> tftp 0xa0000000 jffs2-128k-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x62000000 +$filesize
=> erase 0x62000000 +$filesize
=> cp.b 0xa0000000 0x62000000 $filesize
=> protect on 0x62000000 +$filesize

Flash DTB

=> tftp 0xa0000000 dtb-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x61f00000 +$filesize
=> erase 0x61f00000 +$filesize
=> cp.b 0xa0000000 0x61f00000 $filesize
=> protect on 0x61f00000 +$filesize

8.1.7.5. U-boot Env Commands

For JFFS2 Filesystem to work, following are the UBOOT ENV Commands:

=>setenv bootargs 'console=ttyS0,115200 root=/dev/mtdblock6 mtdparts=60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs) rw rootfstype=jffs2 earlycon=uart8250,mmio,0x21c0500'

=>setenv bootcmd 'cp.b 0x61100000 0x8007ffc0 0xd00000;cp.b 0x61f00000 0x90000000 0x100000;bootm 0x8007ffc0 - 0x90000000'

=>setenv mtdparts '60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs)'

=>setenv mtdids 'nor0=60000000.nor'
=>save

8.1.7.6. Boot Linux with auto-boot

Follow the following steps to boot using auto boot:

1. Make sure all required images are flashed on the board-using step 8.6.1 to 8.6.4.
2. Make sure following env setting is saved:
   bootargs= console=ttyS0,115200 root=/dev/mtdblock6 mtdparts=60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs) rw rootfstype=jffs2 earlycon=uart8250,mmio,0x21c0500
   bootcmd= cp.b 0x61100000 0x8007ffc0 0xd00000;cp.b 0x61f00000 0x90000000 0x100000;bootm 0x8007ffc0 - 0x90000000
3. Power cycle the board to boot it until the boot prompt.

8.1.8. UBIFS Read-Write File System

8.1.8.1. NOR Flash Layout for UBIFS File System

For the Usage of UBIFS Filesystem, the following is the QSPI Flash Layout:

Start Address	*End Address*	*Image*	*Size*
0x060000000	*0x0600FFFFF*	*RCW + PBI*	*1MB*

0x060100000	*0x0601FFFFF*	*U Boot*	*1MB*
0x060200000	*0x0602FFFFF*	*U Boot Env*	*1MB*
0x060300000	*0x0610FFFFF*	*FMan ucode*	*14MB*
0x061100000	*0x61EFFFFF*	*uImage(Kernel)*	*14MB*
0x061F00000	*0x61FFFFFF*	*DTB*	*1MB*
0x062000000	*0x063FF_FFFF*	*Rootfs*	*32MB*

8.1.9. Flashing & Booting pre-built UBIFS ASK Images using TFTP

UBIFS is the read/write file system that can used on QSPI Flash.

NOTE-1: *This section only explains the example of flashing JFFS2 based filesystem. For RAM Based file system, please refer Flashing & Booting pre-built RamDisk ASK Images using TFTP*

NOTE-2: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB*

.

8.1.9.1. Flash RCW

Refer Flash RCW for flashing RCW if not flashed already.

8.1.9.2. Flash U-boot

Refer Flash U-boot for flashing u-boot if not flashed already.

8.1.9.3. Flash DPAA Firmware

Refer Flash DPAA firmware if not flashed.

8.1.9.4. Flash uImage, DTB, Rootfs

Flash uImage :

```
=> tftp 0xa0000000  uImage-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x61100000 +$filesize
=> erase 0x61100000 +$filesize
=> cp.b 0xa0000000 0x61100000 $filesize
=> protect on 0x61100000 +$filesize
```

Flash Rootfs :

```
=> tftp 0xa0000000  ubi-nor-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x62000000 +0x2000000
=> erase 0x62000000 +0x2000000
=> cp.b 0xa0000000 0x62000000 $filesize
=> protect on 0x62000000 +$filesize
```

Flash DTB

```
=> tftp 0xa0000000  dtb-openwrt-ls1043a_7.0.0-ls1043a-rdb
=> protect off 0x61f00000 +$filesize
=> erase 0x61f00000 +$filesize
```

=> cp.b 0xa0000000 0x61f00000 $filesize
=> protect on 0x61f00000 +$filesize

8.1.9.5. Uboot Env Variables

- *setenv bootargs 'console=ttyS0,115200 mtdparts=60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs) rw rootfstype=ubifs root=ubi0:rootfs ubi.mtd=6 earlycon=uart8250,mmio,0x21c0500'*
- *setenv bootcmd 'cp.b 0x61100000 0x8007ffc0 0xd00000;cp.b 0x61f00000 0x90000000 0x100000;bootm 0x8007ffc0 - 0x90000000'*
- *setenv mtdparts '60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs)'*
- *setenv mtdids 'nor0=60000000.nor'*
- *save*

8.1.9.6. Boot Linux with auto-boot

Follow the following steps to boot using auto boot:

1. Make sure all required image are flashed on the board-using step 8.6.1 to 8.6.4.
2. Make sure following env setting is saved:
   bootargs=console=ttyS0,115200 mtdparts=60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs) rw rootfstype=ubifs root=ubi0:rootfs ubi.mtd=6 earlycon=uart8250,mmio,0x21c0500
   bootcmd= cp.b 0x61100000 0x8007ffc0 0xd00000;cp.b 0x61f00000 0x90000000 0x100000;bootm 0x8007ffc0 - 0x90000000
3. Power cycle the board to boot it until the boot prompt.

8.2. LS1043A-RGW

8.2.1. NOR Flash Layout

Start Address	*End Address*	*Description*
0x0000000	*0x011FFFF*	*RCW, SPL, U-Boot*
0x0120000	*0x013FFFF*	*FMAN ucode*

0x0140000	*0x015FFFF*	*QE ucode*
0x0160000	*0x315FFFF*	*FIT image*
0x3160000	*0x7FFFFFF*	*User space*

8.2.2. U-boot Environment Variables

The following is the **default u-boot** environment configuration

baudrate=115200
bootcmd=nand read $kernel_load $kernel_addr $kernel_size;bootm $kernel_load
bootdelay=10
console=ttyAMA0,38400n8
eth1addr=00:04:9F:04:60:bf
eth2addr=00:04:9F:04:60:c0
eth3addr=00:04:9F:04:60:c1
eth4addr=00:04:9F:04:60:c2
eth5addr=00:04:9F:04:60:c3
eth6addr=00:04:9F:04:60:c4
ethact=FM1@DTSEC1
ethaddr=00:04:9F:04:60:be
ethprime=FM1@DTSEC1
fdt_high=0xffffffffffffffff
fdtcontroladdr=ffc06600
fileaddr=a0000000
filesize=10a8a2f
fman_ucode=ffc27300
hwconfig=fsl_ddr:bank_intlv=auto
initrd_high=0xffffffffffffffff
kernel_addr=160000
kernel_load=0xa0000000
kernel_size=10a8a2f
kernel_start=0x61200000
loadaddr=0x80100000
qe_ucode=ffc172f0
ramdisk_addr=0x800000
ramdisk_size=0x2000000
stderr=serial
stdin=serial
stdout=serial

NOTE-I: Default env setting sets bootargs for RAM based file system.

8.2.3. Changing the Env Variable

Any env variable can be changed using following command:

setenv <Variable Name> <Variable Value>

Ex:

setenv bootargs ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200

8.2.4. Set the IP address

Change the IP address (in environment variable) using following command:

=>setenv serverip 192.168.5.124; setenv ipaddr 192.168.5.13;.setenv ethact FM1@DTSEC1

Connect the ethernet cable to DTSEC1 port of LS1043A RGW to your Local PC

8.2.5. Flashing & Booting pre-built RamDisk ASK Images using TFTP

NOTE-1: *This section only explains the example of flashing RAM based filesystem. For Flash Based file system, please refer section UBIFS Read-Write File System and Flashing & Booting pre-built UBIFS ASK Images using TFTP*

NOTE-2: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB*

8.2.5.1. Flash U-boot

```
=> tftp 0xa0000000 u-boot-2016.01-v201601.02.1-ls1043a-rgw.bin
Using FM1@DTSEC1 device
TFTP from server 192.168.1.1; our IP address is 192.168.1.10
Filename 'u-boot-nand.bin'.
Load address: 0xa0000000
Loading: #################################################
5.6 MiB/s
done
Bytes transferred = 684231 (a70c7 hex)
=> nand erase 0 $filesize
NAND erase: device 0 offset 0x0, size 0xa70c7
Erasing at 0xa0000 -- 100% complete.
OK
=> nand write $fileaddr 0 $filesize
NAND write: device 0 offset 0x0, size 0xa70c7
684231 bytes written: OK

Important: Before performing kernel flashing, reset the board.
=> reset
```

8.2.5.2. Flash DPAA firmware

```
=> tftp 0xa0000000 fsl_fman_ucode_r2.bin
=> protect off $fman_ucode +$filesize
=> erase $fman_ucode +$filesize
=> cp.b a0000000 $fman_ucode $filesize
=> protect on $fman_ucode +$filesize

=> tftp 0xa0000000 fsl_fman_ucode_ls1043_r1.0_108_4_9.bin
Using FM1@DTSEC1 device
```

```
TFTP from server 192.168.1.1; our IP address is 192.168.1.10
Filename 'fsl_fman_ucode_ls1043_r1.0_108_4_9.bin'.
Load address: 0xa0000000
Loading: ###
1.6 MiB/s
done
Bytes transferred = 37560 (92b8 hex)
=> nand erase 120000 $filesize
NAND erase: device 0 offset 0x120000, size 0x92b8
Erasing at 0x120000 -- 100% complete.
OK
=> nand write $fileaddr 120000 $filesize
NAND write: device 0 offset 0x120000, size 0x92b8
37560 bytes written: OK
```

NOTE-1: -- *This step is not required if there is a valid firmware exists in flash. The 7.0.0 firmware is automatically loaded with Linux kernel.*

8.2.5.3. Flash Linux

```
=> tftp 0xa0000000 itb-openwrt-ls1043a_7.0.0-ls1043a-rgw
Using FM1@DTSEC1 device
TFTP from server 192.168.1.1; our IP address is 192.168.1.10
Filename 'itb-openwrt-ls1043a_0.8-rc1-ls1043a-lsrgw'.
Load address: 0xa0000000
Loading: ################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
##
5.6 MiB/s
done
Bytes transferred = 13375439 (cc17cf hex)
=> nand erase 160000 $filesize
NAND erase: device 0 offset 0x160000, size 0xcc17cf
Erasing at 0xe20000 -- 100% complete.
OK
=> nand write $fileaddr 160000 $filesize
NAND write: device 0 offset 0x160000, size 0xcc17cf
13375439 bytes written: OK
=> setenv kernel_size $filesize
=> saveenv
Saving Environment to NAND...
```

Erasing NAND...
Erasing at 0x100000 -- 100% complete.
Writing to NAND... OK

8.2.5.4. Booting Linux with auto-boot

Follow following steps to boot using auto boot:

1. Make sure all required image are flashed on the board-using step 8.2.5.1 to 8.2.5.4
2. Make sure following env setting is saved:
   => setenv bootargs "console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500"
   => setenv kernel_addr 160000
   => setenv kernel_load 0xa0000000
   => setenv bootcmd "nand read '$kernel_load' '$kernel_addr' '$kernel_size';bootm '$kernel_load'"
   => saveenv
   Reboot the board to get the updated fman changes and board will boot with the updated image by default.
   => reset

3. Power cycle the board to boot it until the boot prompt.

8.2.5.5. Booting Linux without auto-boot

Follow following steps to boot Linux manually (this will be required if user wants to change the itb image and boot):

1. Press enter at u-boot prompt to stop autoboot.
2. Make sure following env setting is saved:
   bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500
   bootcmd=cp.b $kernel_start $kernel_load $kernel_size && bootm $kernel_load

3. Load new itb image to DDR using TFTP
   => tftp $kernel_load itb-openwrt-ls1043a_1.0.rc.0-ls1043a-rdb
4. Boot Linux
   =>bootm $kernel_load

8.2.6. UBIFS Read-Write File System

8.2.6.1. NAND Flash Layout for UBIFS File System

For the Usage of UBIFS Filesystem, the following is the NAND Flash Layout:

Start Address	*End Address*	*Image*	*Size*
0x0000000	*0x011FFFF*	*RCW+UBOOT*	*1152KB*
0x0120000	*0x013FFFF*	*FMAN*	*128KB*
0x0140000	*0x04FFFFF*	*RESERVED*	*3840K*
0x0500000	*0x12FFFFF*	*KERNEL*	*14MB*
0x1300000	*0x13FFFFF*	*DTB*	*1MB*

0x1400000	*0x7FFFFFF*	*ROOTFS*	*108MB*

8.2.7. Flashing & Booting pre-built UBIFS ASK Images using TFTP

UBIFS is the read/write file system that can used on QSPI Flash.

NOTE-1: *This section only explains the example of flashing UBIFS. For RAM Based file system, please refer* Flashing & Booting pre-built RamDisk ASK Images using TFTP

NOTE-2: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB*

8.2.7.1. Flash U-boot

Refer Flash U-boot for flashing u-boot if not flashed already.

8.2.7.2. Flash DPAA firmware

Refer Flash DPAA firmware for flashing DPAA firmware Image if not flashed already.

8.2.7.3. Flash uImage, DTB, Rootfs

Setting U-boot environment commands to flash uImage, dtb and Rootfs

Set the below Uboot Variables

```
=>setenv bootargs 'console=ttyS0,115200
mtdparts=7e800000.flash:1152K(Uboot)ro,128K(Fman),3840K(Reserved),14M(kernel),1M(dtb),108M(rootfs) rw
rootfstype=ubifs root=ubi0:rootfs ubi.mtd=5 earlycon=uart8250,mmio,0x21c0500'
=>setenv bootcmd 'nand read  0x8007ffc0 0x500000 0xe00000;nand read 0x90000000 0x1300000
0x100000;bootm 0x8007ffc0 - 0x90000000'
=>setenv mtdparts '7e800000.flash:1.2M(Uboot)ro,1M(Fman),1.8M(Reserved),14M(kernel),1M(dtb),108M(rootfs)'
setenv mtdids 'nand0: 7e800000.flash'
=>setenv rootfs_file 'ubi-nand-openwrt-ls1043a_7.0.0-rc1-ls1043a-rgw'
=>setenv dtb_file 'dtb-openwrt-ls1043a_7.0.0-rc1-ls1043a-rgw'
=>setenv uimage_file 'uImage-openwrt-ls1043a_7.0.0-rc1-ls1043a-rgw'
=>setenv update_kernel 'tftp 0xa0000000 $uimage_file;nand erase 0x500000 0xe00000;nand write 0xa0000000
0x500000 $filesize'
=>setenv update_dtb 'tftp 0xa0000000 $dtb_file;nand erase 0x1300000 0x100000;nand write 0xa0000000
0x1300000 $filesize'
=>setenv update_rootfs 'tftp 0xa0000000 $rootfs_file;nand erase 0x1400000 0x6C00000;nand write 0xa0000000
0x1400000 $filesize'
=>saveenv
```

Flash uImage :

Run the following command to update the kernel after setting the uboot variables as above

```
=> run update_kernel
```

Flash Rootfs :

Run the following command to update the kernel after setting the uboot variables as above

```
=> run update_rootfs
```

Flash DTB

Run the following command to update the kernel after setting the uboot variables as above

=> run update_dtb

8.2.7.4. Uboot Env Variables

- *setenv bootargs 'console=ttyS0,115200 root=/dev/mtdblock6 mtdparts=60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs) rw rootfstype=jffs2 earlycon=uart8250,mmio,0x21c0500'*
- *setenv bootcmd 'cp.b 0x61100000 0x8007ffc0 0xb00000;cp.b 0x61f00000 0x90000000 0x100000;bootm 0x8007ffc0 - 0x90000000'*
- *setenv mtdparts '60000000.nor:1M(RCW)ro,1M(u-boot)ro,1M(env),14M(fman)ro,14M(kernel),1M(dtb),32M(rootfs)'*
- *setenv mtdids 'nor0=60000000.nor'*
- *save*

8.2.7.5. Boot Linux with auto-boot

Follow the following steps to boot using auto boot:

1. Make sure all required image are flashed on the board-using step 8.6.1 to 8.6.4.
2. Make sure following env setting is saved:
   bootargs= bootargs 'console=ttyS0,115200 mtdparts=7e800000.flash:1152K(Uboot)ro,128K(Fman),3840K(Reserved),14M(kernel),1M(dtb),108M(rootfs) rw rootfstype=ubifs root=ubi0:rootfs ubi.mtd=5
   bootcmd= 'nand read 0x8007ffc0 0x500000 0xe00000;nand read 0x90000000 0x1300000 0x100000;bootm 0x8007ffc0 - 0x90000000'
3. Power cycle the board to boot it until the boot prompt.

8.3. LS1046A-RDB

8.3.1. Flash Usage

LS1046ARDB has 2 QSPI flash connected over QSPI controller. Only one QSPI flash is available at a time depending upon the board switch settings. These switch settings can also be overridden by CPLD commands.

To protect the default U-Boot in flash0 (aka bank0), it is a convention employed by NXP to deploy work images into the flash1 (aka bank4), and then switch to the flash1 (aka bank4) for testing. Switching to the flash1 (aka bank4) can be done in software using CPLD commands and effectively swaps the flash0 (aka bank0) with the flash1 (aka bank4). This protects flash1 and keeps the board bootable under all circumstances.

To determine the current bank, refer to the following U-Boot log

```
U-Boot 2016.01 (Aug 19 2016 - 16:59:27 +0800)

SoC:  LS1046E (0x87070010)
Clock Configuration:
       CPU0(A72):1600 MHz  CPU1(A72):1600 MHz  CPU2(A72):1600 MHz
       CPU3(A72):1600 MHz
       Bus:      600  MHz  DDR:      2100 MT/s  FMAN:     700  MHz
Reset Configuration Word (RCW):
       00000000: 0c150010 0e000000 00000000 00000000
       00000010: 11335559 40005012 40025000 c1000000
       00000020: 00000000 00000000 00000000 00238800
       00000030: 20124000 00003101 00000096 00000001
I2C:   ready
Model: LS1046A RDB Board
Board: LS1046ARDB, boot from QSPI vBank 0
CPLD:  V2.2
PCBA:  V2.0
SERDES Reference Clocks:
SD1_CLK1 = 156.25MHZ, SD1_CLK2 = 100.00MHZ
DRAM:  Initializing DDR....using SPD
Detected UDIMM 18ASF1G72AZ-2G3B1
8 GiB (DDR4, 64-bit, CL=15, ECC on)
       DDR Chip-Select Interleaving Mode: CS0+CS1
SEC0: RNG instantiated
PPA Firmware: Version 0.2
Using SERDES1 Protocol: 4403 (0x1133)
Using SERDES2 Protocol: 21849 (0x5559)
```

Bank switching can be done in U-Boot using the following statements:

- Switch to QSPI bank 0(default) for RDB:
  ```
  =>cpld reset
  ```

- Switch to QSPI bank 4 for RDB:
  ```
  =>cpld reset altbank
  ```

8.3.2. QSPI Flash Layout

Start	*End Offset*	*Description*	*Size*
0x40000000	*0x400FFFFF*	*bank0 rcw + pbi*	*1 M*
0x40100000	*0x401FFFFF*	*bank0 U-Boot image*	*1 M*
0x40200000	*0x402FFFFF*	*bank0 U-Boot Env*	*1 M*
0x40300000	*0x403FFFFF*	*bank0 Fman ucode*	*1 M*

0x40400000	*0x404FFFFF*	*bank0 UEFI*	*1 M*
0x40500000	*0x406FFFFF*	*bank0 Primary Protected Application (PPA)*	*2 M*
0x40700000	*0x408FFFFF*	*Secure boot header + bootscript*	*2 M*
0x40900000	*0x40FFFFFF*	*bank0 reserved*	*7 M*
0x41000000	*0x43FFFFFF*	*bank0 FIT Image*	*48 M*
0x44000000	*0x440FFFFF*	*bank4 rcw + pbi*	*1 M*
0x44100000	*0x441FFFFF*	*bank4 U-Boot image*	*1 M*
0x44200000	*0x442FFFFF*	*bank4 U-Boot Env*	*1 M*
0x44300000	*0x443FFFFF*	*bank4 Fman ucode*	*1 M*
0x44400000	*0x444FFFFF*	*bank4 UEFI*	*1 M*
0x44500000	*0x446FFFFF*	*bank4 PPA*	*2 M*
0x44700000	*0x448FFFFF*	*Secure boot header + bootscript*	*2 M*
0x44900000	*0x44FFFFFF*	*bank4 reserved*	*7 M*
0x45000000	*0x47FFFFFF*	*bank4 FIT Image*	*48 M*

8.3.3. U-boot Environment Variables

The following is the default u-boot environment configuration

baudrate=115200
bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500, pci=nomsi, $othbootargs
bootcmd=env default bootargs;sf probe 0:0;sf read $kernel_load $kernel_start $kernel_size;bootm $kernel_load
bootdelay=10
console=ttyS0,115200
eth1addr=00:04:9f:04:a8:3f
eth2addr=00:04:9f:04:a8:3a
eth3addr=00:04:9f:04:a8:3b
eth4addr=00:04:9f:04:a8:3d
eth5addr=00:04:9f:04:a8:3c
eth6addr=00:00:00:00:00:01
eth7addr=00:00:00:00:00:02
ethact=FM1@DTSEC3
ethaddr=00:04:9f:04:a8:3e
ethprime=FM1@DTSEC3
fdt_high=0xffffffffffffffff
fdtcontroladdr=ffe0aa28
fman_ucode=ffe1c9f0

gatewayip=10.232.85.254
hwconfig=fsl_ddr:bank_intlv=auto
initrd_high=0xffffffffffffffff
ipaddr=10.232.85.122
kernel_addr=0x100000
kernel_load=0xa0000000
kernel_size=0x2800000
kernel_start=0x1000000
loadaddr=0x80100000
mtdparts=1550000.quadspi:1m(rcw),15m(u-boot),48m(kenel.itb);7e800000.flash:16m(nand_uboot),48m(nand_kernel),448m(nand_free)
othbootargs=mtdparts=1550000.quadspi:1m(rcw),15m(u-boot),48m(kenel.itb);7e800000.flash:16m(nand_uboot),48m(nand_kernel),448m(nand_free)
ramdisk_addr=0x800000
ramdisk_size=0x2000000
scsidevs=0
serverip=10.232.90.101
stderr=serial
stdin=serial
stdout=serial

Environment size: 1258/8188 bytes

NOTE-I: Default env setting sets bootargs for RAM based file system.

8.3.4. Changing the Env Variable

Any env variable can be changed using following command:

setenv <Variable Name> <Variable Value>

Ex:

setenv bootargs ttyS0,115200 root=/dev/ram0 earlycon=uart8250,0x21c0500,115200

8.3.5. Set the IP address

Change the IP address (in environment variable) using following command:

=>setenv serverip 192.168.5.124; setenv ipaddr 192.168.5.136.setenv ethact FM1@DTSEC1

Connect the ethernet cable to DTSEC1 port of LS1046A RDB to your Local PC

8.3.6. Flashing & Booting pre-built RamDisk ASK Images using TFTP

The following sections will share the instructions to update the images. For specific addresses, please refer to the QSPI Flash Memory Map as a reference. If the user intends to flash images at both banks at once; this can be done using the probe for respective bank e.g. if images needs to be updated for bank 0 “sf probe 0:0” can be used from bank 0 and if images needs to be updated for alternate bank from bank 0 “sf probe 0:1” can be used from bank 0. Once the images are flashed, respective images can be selected using correct “cpld reset” command.

NOTE-1: *For Image names in the below sections please refer LS1043ARDB for LS1043A-RDB, LS1043ARGW for LS1043ARGW and LS1046ARDB for LS1046ARDB*

8.3.6.1. Flash U-boot

By default, an existing U-Boot is run in QSPI bank 0 after the system is powered on or after a hard reset is performed. To flash U-Boot to the alternate bank, first switch to bank 0 by performing a hard reset or by typing reset. Then use the following commands to flash a new U-Boot into the alternate bank. Once the flashing of images completes switch to the alternate bank where the new U-Boot:

```
=>tftp 82000000 u-boot-2016.01-v201601.02.1-ls1046a-rdb.bin
=>sf probe 0:1
=>sf erase 100000 +$filesize
=>sf write 82000000 100000 $filesize
=>cpld reset altbank
```

8.3.6.2. Flash RCW

To program a new RCW, first switch to QSPI bank 0 by performing a hard reset or by typing reset. Next, load the new RCW to RAM by downloading it via TFTP and then copying it to flash offset 0x0. 0x0 is the offset address of RCW in the QSPI flash. Execute the following commands at the U-Boot prompt to program the RCW to QSPI flash and reset to alternate bank

```
=>tftp 82000000 rcw_1600_qspiboot_swap.bin
=>sf probe 0:1
=>sf erase 0 +$filesize
=>sf write 82000000 0 $filesize;
=>cpld reset altbank
```

8.3.6.3. Flash DPAA firmware

To program a new microcode, first switch to QSPI bank 0 by performing a hard reset or by typing cpld reset. Next, load the new microcode to RAM by downloading it via TFTP and then writing it to flash offset 0x300000. 0x300000 is the offset of ucode in the QSPI flash. Then execute the following commands at the U-Boot prompt to program the ucode to flash and reset to alternate bank.

```
=>tftp 83000000 fsl_fman_ucode_r2.bin
=>sf probe 0:1
=>sf erase 300000 +$filesize
=>sf write 83000000 300000 $filesize
=>cpld reset altbank
```

NOTE-1: -- ***This step is not required if there is a valid firmware exists in flash. The 7.0.0 firmware is automatically loaded with Linux kernel.***

8.3.6.4. Flash Linux

To program a new kernel itb, first switch to QSPI bank 0 by performing a hard reset or by typing cpld reset. Next, load the new itb to RAM by downloading it via TFTP and then writing it to flash offset 0x1000000. 0x1000000 is the offset of the kernel itb in the QSPI flash. Then execute the following commands at the U-Boot prompt to program the kernel itb to flash and reset to alternate bank.

```
=>tftp a0000000 itb-openwrt-ls1046a_7.0.0-ls1046a-rdb
=>sf probe 0:1
=>sf erase 1000000 +$filesize
=>sf write a0000000 1000000 $filesize
=>cpld reset altbank
```

8.3.6.5. Booting Linux with auto-boot

Follow following steps to boot using auto boot:

1. Make sure all required image are flashed on the board-using step 8.3.6.1 to 8.3.6.5.
2. Make sure following env setting is saved:
   bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500, pci=nomsi, $othbootargs
   bootcmd=env default bootargs;sf probe 0:0;sf read $kernel_load $kernel_start $kernel_size;bootm $kernel_load
3. Power cycle the board to boot it until the boot prompt.

8.3.6.6. Booting Linux without auto-boot

Follow following steps to boot Linux manually (this will be required if user wants to change the itb image and boot):

5. Press enter at u-boot prompt to stop autoboot.
6. Make sure following env setting is saved:
   bootargs=console=ttyS0,115200 root=/dev/ram0 earlycon=uart8250,mmio,0x21c0500, pci=nomsi, $othbootargs

   bootcmd=env default bootargs;sf probe 0:0;sf read $kernel_load $kernel_start $kernel_size;bootm $kernel_load

7. Load new itb image to DDR using TFTP
   => tftp $kernel_load itb-openwrt-ls1046a_7.0.0-ls1046a-rdb

8. Boot Linux
   =>bootm $kernel_load

9. Guideline for WiFi drivers

9.1. Quantenna PCIe 11ac 4x4

1. Connect Quantenna Mini PCIe WiFi module to the mini PCIe slot on the RDB.
2. Boot the system with UBIFS/JFFS as rootfs.

3. Interface with name “host0” can be seen on successful loading of the driver.

4. Run following command for interface UP:

ifconfig host0 <ip address> netmask <ip mask> up e.g. "ifconfig host0 192.168.2.1 netmask 255.255.255.0 up"

5. Refer quantenna documentation to configure AP and STA.

9.2. Setting up WiFi fast path

To enable WiFi fast path following steps needs to be done.

1. Configure/modify cmm config file to fast forward the wifi interface. Following lines should be present in /etc/config/fastforward file for the each wifi interface needs to be fast forward.

   #WiFi fastforward configuration for Quantenna driver
   config wifi_fastforward
   option ifname host0
   option direct_path_rx 0

2. After modifying /etc/config/fastforward reboot the system.

3. For wifi fast-forwarding in bridge mode set following, default configuration is route mode.

   root@OpenWrt:/# echo 1 > /sys/class/vwd/vwd0/vwd_bridge_hook_enable

4. For wifi fast-forwarding in route mode set following.

   root@OpenWrt:/# echo 1 > /sys/class/vwd/vwd0 /vwd_route_hook_enable

5. Now the wifi traffic will be fast-forwarded.

10. Egress QOS Configuration

Below is the example configuration with 100mbps port shaper and queue 0 to 8 are enabled. In this configuration queue 9 is not enabled so, this traffic goes through low priority queue (0).

Iptables –A POSTROUTING –t mangle –j CONNMARK –p udp --dport 1000 -s 192.168.3.10 --set-mark 0
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1001 -s 192.168.3.10 --set-mark 1
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1002 -s 192.168.3.10 --set-mark 2
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1003 -s 192.168.3.10 --set-mark 3
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1004 -s 192.168.3.10 --set-mark 4
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1005 -s 192.168.3.10 --set-mark 5
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1006 -s 192.168.3.10 --set-mark 6
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1007 -s 192.168.3.10 --set-mark 7
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1008 -s 192.168.3.10 --set-mark 8
iptables -A POSTROUTING -t mangle -j CONNMARK -p udp --dport 1009 -s 192.168.3.10 --set-mark 9

cmm -c set qm interface eth1 scheduler 0 queue 8 queue 7 queue 6 queue 5 queue 4 queue 3 queue 2 queue 1 queue 0
cmm -c set qm interface eth1 shaper port on rate 100000 ifg 200 bucket_size 4096
cmm -c set qm interface eth1 qos on

11. NPT-V6 configuration

LAN IP: 2001::1/64; WAN IP: 2006::1/64
Host1 IP : 2001::2/64; Host2 IP: 2006::2/64

ip6tables -t mangle -A POSTROUTING -s 2001::/64 -o eth6 -j SNPT --src-pfx 2001::/64 --dst-pfx 1111::/64
ip6tables -t mangle -A PREROUTING -d 1111::/64 -i eth6 -j DNPT --src-pfx 1111::/64 --dst-pfx 2001::/64

11.1. Usage

V6-Host1---------[LAN] DUT[WAN]----------v6-Host2

- When traffic from Host1 to Host2 flows, DUT translates the source IP of the Packet to 1111::ef0:0:0:2 as per the ip6tables mangle rule(POSTROUTING).
- When traffic from Host2 destined to 1111::ef0:0:0:2 flows, the traffic is forwarded to Host1 based on the ip6tables mangle rule(PREROUTING).
- All this is done in the fastpath. These connections should be seen in ‘show connections’ & ‘query v6connections’ of cmm.

12. Firewall Configuration

In the current release by default no firewall rules are configured including NAT. To enable NAT following firewall rules needs to be configured.

iptables -P INPUT DROP
iptables -A INPUT –i eth2 -j ACCEPT
iptables -P FORWARD DROP
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o eth2 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i eth2 -o eth0 -j ACCEPT

13. Known issues

Know issues in this release are:

1. Port-lockup crash is observed at high load for small size packets up to 400 bytes when traffic is pumped between 1G and 10G ports in LS1043 RDB and RGW. No issue is observed with 10G ports on LS1046.
2. Port-lockup crash is observed with collisions in Flow table.
3. PPPoE client is not connecting consistently with server in case of PPP0E-v6
4. Multiple SSIDs are not supported and eth0 interface should be up while testing wifi offload.
5. Multicast Fast path is not supported.
6. NAT-PT fast path is not supported.
7. IPSEC is not supported with IPv6 data on IPv4 VPN tunnel and IPv4 data on IPv6 tunnel.
8. An error is observed while configuring NULL encryption.
9. A crash is observed while viewing entries in auto bridge module.
10. A crash is observed when wrong username and password entered while doing FTP to the DUT.

14. Appendix A (Performance Numbers)

14.1. LS1046 RDB

14.1.1. Fast Path Performance

Following tests are done with Spirent Test center.
LAN => eth4@10G
WAN => eth5@10G

14.1.1.1. IPv4 UDP Routing (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
64	4664179	3,170
128	4664179	5,523
256	4464285	9,933
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

WAN to LAN		
Frame Size	**Fps**	**Mbps**
64	4734848	3,177
128	4734848	5,618
256	4464285	10,000
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.2. IPv4 UDP Routing (Bi-Directional)

Frame Size	**Fps**	**Mbps**
64	6127450	4,140
128	6067960	7,217
256	6009614	13,282
390	5733944	18,865
512	4699248	20,000
1024	2394636	20,000
1280	1923076	20,000

1518	1623376	20,000

14.1.1.3. IPv4 NAT UDP Routing (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
66	4664179	3,209
128	4595588	5,459
256	4528985	10,000
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

WAN to LAN		
Frame Size	**Fps**	**Mbps**
66	4595588	3,200
128	4595588	5,488
256	4464285	9,898
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.4. IPv4 NAT UDP Routing (Bi-Directional)

Frame Size	**Fps**	**Mbps**
66	5681818	3,937
128	5681818	6,721
256	5530972	12,304
390	5296610	17,500
420	5681818	20,000
512	5681818	20,000
1024	4699248	20,000
1280	2394636	20,000
1518	1923076	20,000

14.1.1.5. IPv4 NAT UDP Routing with 32768 connections (Bi-Directional)

Frame Size	Fps	Mbps
66	5787036	3,980
128	5787036	6,846
256	5733944	12,756
390	5681818	18,773
420	5681818	20,000
512	5681818	20,000
1024	4699248	20,000
1280	2394636	20,000
1518	1923076	20,000

14.1.1.6. IPv4 Fast Path IMIX Bi-directional stream

Frame Size	Weight	Passed Rate	FPS	Throughput (Mbps)
66	7	77.565%	5040322	15,513
594	4			
1518	1			

14.1.1.7. IPv4 VLAN UDP Routing (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
70	3511235	2,550
128	3551136	4,199
256	3472222	7,741
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

WAN to LAN		
Frame Size	**Fps**	**Mbps**
70	3591954	2,586
128	3591954	4,290
256	3551136	7,912
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000

1280	961538	10,000
1518	811688	10,000

14.1.1.8. IPv4 VLAN UDP Routing (Bi-Directional)

Frame Size	Fps	Mbps
70	4194630	3,034
128	4194630	4,970
256	4166666	9,238
390	4139072	13,575
512	4111842	17,497
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.9. IPv4 VLAN UDP Routing with 1024 connections (Bi-Directional)

Frame Size	Fps	Mbps
70	4222972	3,056
128	4222972	5,023
256	4222972	9,319
390	4194630	13,829
512	4194630	17,843
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.10. IPv4 PPPoE UDP Routing (Uni-Directional)

LAN to WAN			
Frame Size	Fps	Clear Through put (Mbps)	Clear Through put Mbps
74	3633720	2,759	2,965
128	3676470	4,349	4,588
256	3676470	7,926	8,353
390	2976190	9,844	9,952
512	2314814	9,907	10,000
1024	1188212	9,952	10,000
1280	955657	9,962	10,000
1518	813802	9,958	10,000

WAN to LAN			
Frame	Fps	Clear	Clear

Size		Through put (Mbps)	Through put Mbps
74	3906250	2,938	2,938
128	3906250	4,625	4,625
256	3810975	8,415	8,415
390	3033980	9,951	9,951
512	2349624	10,000	10,000
1024	1197318	10,000	10,000
1280	961538	10,000	10,000
1518	815926	10,000	10,000

14.1.1.11. IPv4 PPPoE UDP Routing (Bi-Directional)

Frame Size	Fps	Clear Through put (Mbps)	Clear Through put Mbps
74	4464284	3,357	3,643
128	4464284	5,286	5,571
256	4432624	9,787	10,071
390	4401408	14,437	14,718
512	4280820	18,219	18,493
1024	2376424	19,848	20,000
1280	1911314	19,878	20,000
1518	1627604	19,922	20,000

14.1.1.12. IPv4 UDP Bridging(Bi-Directional)

Frame Size	Fps	Mbps
64	8333332	5,625
128	8012820	9,546
256	7183908	15,995
390	6067960	20,000
512	4699248	20,000
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.13. IPv4 UDP Bridging (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
64	5482456	3,696
128	5296610	6,262
256	4464285	10,000

390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.14. IPv4 VLAN UDP Bridging(Uni-Directional)

Frame Size	Fps	Mbps
70	3511235	2,550
128	3551136	4,199
256	3472222	7,741
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.15. IPv4 VLAN UDP Bridging(Bi-Directional)

Frame Size	Fps	Mbps
70	4194630	3,034
128	4194630	4,970
256	4166666	9,238
390	4139072	13,575
512	4111842	17,497
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.16. IPv6 UDP Routing (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
86	4734848	4,064
128	4664179	5,547
256	4464285	10,000
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

WAN to LAN		
Frame Size	**Fps**	**Mbps**
86	4664179	3,995
128	4664179	5,515
256	4464285	10,000
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.17. IPv6 UDP Routing (Bi-Directional)

Frame Size	Fps	Mbps
86	6377550	5,416
128	6377550	7,581
256	6188118	13,771
390	5733944	18,895
512	4699248	20,000
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.18. IPv6 UDP Routing with 32768 connections(Bi-Directional)

Frame Size	Fps	Mbps
86	6578946	5,626
128	6578946	7,854
256	6578946	14,507
390	6067960	20,000
512	4699248	20,000
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.19. IPv6 Fast Path IMIX Bi-directional stream

Frame Size	Weight	Passed Rate	FPS	Throughput (Mbps)
86	7	80.000%	5081300	16,000
594	4			
86	7			

14.1.1.20. IPv6 VLAN UDP Routing (Uni-Directional)

LAN to WAN		
Frame Size	**Fps**	**Mbps**
128	3720238	4,401
256	3633720	8,105
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

WAN to LAN		
Frame Size	**Fps**	**Mbps**
128	3720238	4,430
256	3676470	8,115
390	3033980	10,000
512	2349624	10,000
1024	1197318	10,000
1280	961538	10,000
1518	811688	10,000

14.1.1.21. IPv6 VLAN UDP Routing (Bi-Directional)

Frame Size	Fps	Mbps
128	4370628	5,202
256	4340276	9,634
390	4310344	14,135
512	4280820	18,280
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.22. IPv6 VLAN UDP Routing with 1024 connections (Bi-Directional)

Frame Size	Fps	Mbps
128	2111486	2501
256	4370628	9,707
390	4370628	14,332
512	4340276	18,584
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.1.23. IPv6 UDP Bridging(Bi-Directional)

Frame Size	Fps	Mbps
86	8116882	6,925
128	8012820	9,538
256	7102272	15,823
390	5896226	19,423
512	4699248	20,000
1024	2394636	20,000
1280	1923076	20,000
1518	1623376	20,000

14.1.2. Slow Path (To the board) performance

Slow path performance measured with iperf3 tool between PC and DUT on 10G interface.

14.1.2.1. IPv4 TCP performance.

On DUT following configuration is done.

Iperf3 command used on sender side.
#iperf3 -c <IP_address> -R

Iperf3 command used on receiver side.
#iperf3 -s -i1

TCP	Gbps
Tx	6.31
Rx	6.50

14.1.2.2. IPv4 UDP Performance

Iperf3 command used on sender side.
#iperf3 -u -c IP_address -b10G

Iperf3 command used on receiver side.
#iperf3 -s -i1

UDP	Gbps
Tx	2.88
Rx	2.72

14.1.3. Quantenna mPCIe 11ac 4x4 WiFi Performance with WiFi offload.

Tests are done with IXA setup using dual 1G Eethernets ports pumping 600Mbps on each interface with 1500 byte packets.

14.1.3.1. Routing (fast-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	645+645 =1290	9.2	505+505 =1010	10.4

14.1.3.2. Routing (slow-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	610+605 =1215	17.2 %	505+502 =1007	36.6 %

14.1.3.3. Bridging (fast-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	560+560 =1120	9.2	485+485= 970	10.4

14.1.3.4. Bridging (slow-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	560+560 =1120	14.3	485+485= 970	12.8

14.1.3.1. Slow path (to the board) performance

Slow path performance measured with iperf3 tool between PC and DUT on WIFI interface.

IPv4 TCP performance.

On DUT following configuration is done.

Iperf3 command used on sender side.
#iperf3 -c <IP_address> -R

Iperf3 command used on receiver side.
#iperf3 -s -i1

TCP	Mbps	CPU %
Wifi Tx	420	7.8%
Wifi Rx	900	18.7%

IPv4 UDP Performance

Iperf3 command used on sender side.
#iperf3 -u -c IP_address -u –b1G

Iperf3 command used on receiver side.
#iperf3 -s -i1

UDP	Mbps	CPU %
Wifi Tx	232	2.3%
Wifi Rx	955	16.6%

14.1.4. IPSEC Tunnel (IPv4 Performance through IPv4 ipsec tunnel)

3DES/SHA-1				
Clear packet	Encrypted Packet	Frames Per Second	Clear Thruput (Mbps)	Encrypted Thruput (Mbps)
64	114	2256316	1516	2419
128	178	1888216	2236	2991
256	306	1245018	2749	3247
390	442	932834	3060	3448
512	562	747606	3182	3481
1024	1074	416110	3475	3642
1280	1330	340412	3540	3676
1518	1608	NA	NA	NA

AES128/MD5				
Clear packet	Encrypted Packet	Frames Per Second	Clear Thruput (Mbps)	Encrypted Thruput (Mbps)
64	122	2272726	1527	2582
128	186	2055920	2434	3388
256	314	1558602	3441	4165
390	458	1257544	4125	4809
512	570	1066552	4539	5034
1024	1082	654450	5466	5770
1280	1338	547764	5697	5951
1518	1616	NA	NA	NA

Note-1: The throughput results for 1518B is NA – Not Available, becos Fragmentation/Re-Assembly is not yet supported in this release.

Note-2: Please refer SVT report for performance results for other algorithms

14.2. LS1043 RGW

14.2.1. Fast Path Performance

Following tests are done with Spirent Test center.
LAN => eth0@1G, eth1@1G, eth2@1G, eth3@1G
WAN => eth4@10G

Note-1 : *Unable to measure Performance results for small frame sizes due to **Port Lockup/FMD crash**. Observed **Port Lockup/FMD crash** issue even for 390B frame size for VLAN & Bridging.*

14.2.1.1. IPv4 UDP Routing (Bi-Directional)

Frame Size	Mbps
64	-
128	-
256	-
390	7764
512	8000
1024	8000
1280	8000
1518	8000

14.2.1.2. IPv4 NAT UDP Routing (Bi-Directional)

Frame Size	Mbps
66	-
128	-
256	-
390	7400
512	8000
1024	8000
1280	8000
1518	8000

14.2.1.3. IPv4 VLAN Routing (Bi-Directional)

Frame Size	Mbps
66	-
128	-

256	-
390	-
512	8000
1024	8000
1280	8000
1518	8000

14.2.1.4. IPv6 Routing (Bi-Directional)

Frame Size	Mbps
66	-
128	-
256	-
390	7600
512	8000
1024	8000
1280	8000
1518	8000

14.2.1.5. IPv4/IPv6 L2 bridging (Bi-Directional)

Frame Size	Mbps
66	-
128	-
256	-
390	-
512	8000
1024	8000
1280	8000
1518	8000

14.2.1.6. IPv4 PPPoE NAT UDP Routing (Bi-Directional)

Frame Size	Mbps
66	-
128	-
256	-
390	7400
512	8000
1024	8000
1280	8000
1518	8000

14.2.2. Quantenna mPCIe 11ac 4x4 WiFi Performance with WiFi offload.

Tests are done with IXA setup using dual 1G ethernets ports pumping 600Mbps on each interface with 1500 byte size packets.

14.2.2.1. Routing (fast-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	635+630 = 1265	10.4 %	542+532 = 1074	10.5 %

14.2.2.2. Routing (slow-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	610+605 = 1215	25.5 %	545+542 = 1087	28.9 %

14.2.2.3. Bridging (fast-path)

Transport Protocol	WiFi Receiver		WiFi Transmitter	
	Mbps	CPU %	Mbps	CPU %
UDP	555+550 = 1105	8.4	480+485 = 965	10.4

14.2.2.4. Bridging (slow-path)

Transport Protocol	WiFi Receiver	WiFi Transmitter

	Mbps	CPU %	Mbps	CPU %
UDP	550+560 = 1110	16.9	475+485 = 960	13.9

15. Appendix B – Supported Feature List

Note: Features with yellow highlight are NOT supported in this release

15.1. General Purpose Processor(GPP) Networking

The General Purpose Processor (GPP) networking features provides the support of the Linux operating system on the GPP. Following table lists the GPP networking features.

GPP Networking Features

Feature	Description	Notes
PPP over Ethernet (PPPoE)	Client and server protocol that establishes a peer-to-peer relationship, thus allowing service providers to monitor each session. • Point-to-Point protocol • Compression Control Protocol (CCP) • IP Control Protocol (IPCP) • Authentication Protocols • PAP • CHAP • MS-CHAPv1 • MS-CHAPv2	• RFC 2516 • RFC 1661 • RFC 1962 • RFC 1332 • • RFC 1334 • RFC 1994 • RFC 2433 • RFC 2759
Client/Server	Protocols facilitate file transfer and upgrade capabilities for the system firmware. • DHCP • HTTP • FTP/TFTP • Telnet/SSH • UPnP IGD Daemon • Samba Server support	• RFC 2616. • RFCs 959. TFTP support available from U-Boot only. • RFCs 854, 4251.

Feature	Description	Notes
TCP/IP Protocol Suite	Includes the following TCP/IP protocol support: • IPv4 • TCP • UDP • ICMP • ARP • Static Routing • IGMP • Private Network Address Allocation • DHCP Server • DHCP Client • DNS Server • DNS Proxy • NAT • Private Network • IPv6 • Base Protocol • Neighbor Discovery • Auto-configuration • Internet Control Message Protocol • Addressing • Multicast Addressing • Multicast Listener Discovery (MLDv2) • Assign Numbers of IPv4 Protocol Field and IPv6 Next Header Fields • DHCP • IPv6/IPv4 Dual Stack • IPv4 over IPv6 Tunneling • Multicast Listener Discovery (MLDv2) snooping • Multicast Receive and Transmit • Control module for Fast Packet Processing • Jumbo Framework	• RFC 791 • RFC 793 • RFC 768 • RFC 792 • RFC 826 • RFC 2236 • RFC 1918 • RFCs 2131, 2132, 1533 • RFC 2132 • RFCs 1035, 1101 • RFCs 1035, 1101 • RFCs 3022, 1812 • RFC 2460 • RFC 2461 • RFC 2462 • RFC 2463 • RFC 2373 • RFC 2375 • RFC 3810 • RFC 1700 • RFC 2132
Open SSL	OpenSSL version 1.0.2.d. The features that are accelerated through the accelerator engine are: • AES, DES, 3DES • SHA1, SHA-256, MD5 • TRNG • Public Key	

Feature	Description	Notes
IP Security (IPSEC)	Suite of protocols for securing Internet Protocol (IP) communications by authenticating and/or encrypting each IP packet in a data stream. IPsec also includes protocols for cryptographic key establishment. • Mode-Config • Xauth • AES certified • Security architecture for the Internet Protocol • IP Security Document Roadmap 2411 • Connection type • Tunnel • Transport • Pass-through • Key management • Manual • Automatic Internet Key Exchange (IKE) • Gateway authentication • ISAKMP • Manual and aggressive modes • Diffie-Hellman algorithms: Group 1 (768-bits), Group 2 (1024-bits), and Group 5 (1536-bits) on GPP • Diffie-Hellman algorithms: MODP 1 (768-bits), MODP 2 (1024-bits), MODP 5 (1536-bits) on GPP • Pre-shared secret key • Public Key Infrastructure (PKI) • Public-Key Cryptography Standards (PKCS) • IP protocols • ESP - IP Encapsulation Security Payload • AH - IP Authentication Header • Encryption • AES, 256-bit • 3DES, 168-bit • DES, 56-bit • NULL • VPN Hardware acceleration • Authentication • MD5, HMAC-MD5-96 within ESP and AH • SHA-1, HMAC-SHA-1-96 within ESP and AH • Interoperability: Windows XP, Linux • Manual Connections • VPN pass-through (IPsec, PPTP, L2TP)	• RFC 2411 • • • • • • • RFCs 2401, 2412, 2393 • RFC 2411 • • • • • RFC 2409 • • • • RFCs 2407, 2408 • • • • • • • • • • • • RFC 2406 • RFC 2402 • • • • • • • • RFC 2403 • RFC 2404

Feature	Description	Notes
Firewall Security	The firewall security features provided as part of the kernel are a subset of a full featured firewall implementation. • Stateful Packet Inspection Firewall • Event Logging • Flexible log settings, configurable by web-based management • Log file persistence after power failure or system reboot • Authentication attempt log • Session log • ALG support for FTP/SIP	
Bridging	The Ethernet bridging configuration utility is used to set up, maintain, and inspect the Ethernet bridge configuration in the Linux kernel. • Ethernet WAN and LAN transparent bridging	The 802.1D MAC bridging kernel module runs at L2 layer in protocol stack. Linux bridge code implements a subset of ANSI/IEEE 802.1D.
	• MAC Discovery	IEEE 802.1D (GARP, GMRP, RSTP protocols excluded).
WiFi Access	WiFi card driver providing WiFi access through PCI_ interface. Support off-the shelf WiFi PCI and mini-PCI cards. Data rate control, modules for support of WEP, WPA, and TKIP security protocols.	• IEEE 802.11 •
Quality of Service	Classful and Classless Queuing and Classifiers Types are kernel modules to manage the Quality of Service (QoS) in packet flows. • Classless Queuing • First In First Out (FIFO) • Classful Queuing • Priority (PRIO) • Hierarchical Token Bucket (HTB) • Hierarchical Fair Service Curve (HFSC) • Diff-Serv Marker (DS_MARK) according to MAC, IP, DSCP/ToS value, port, and application • Types of Classifiers • U32 Classifier • FW Classifier	

15.2. Forward Engine (FE) features.

Feature	Description	Notes
FPP PPPoE Relay	Support for PPPoE relay. With this the LAN side PPPoE client can setup a PPPoE link with the server in the WAN side.	RFC 2516 RFC 3817

FPP Generic Fragmentation and Reassembly	• Support fragmentation for IPv4, IPv6, IPSEC-IPv4, IPSEC-IPv6 traffic	RFC 2460 RFC 791
	• Support packet reassembling	• RFC 2473
TCP/IP Protocol: IPv4/ IPv6 Forwarding	• IPv4 • TCP • UDP • ICMP • ARP • Static Routing • Policy Routing • IGMP • Private Network Address Allocation • DHCP Server • DHCP Client • DNS Server • DNS Proxy • NAT • Private Network • IPv6 • Base Protocol • Neighbor Discovery • Auto-configuration • Internet Control Message Protocol • Addressing • Multicast Addressing • Multicast Listener Discovery (MLDv2) • Assign Numbers of IPv4 Protocol Field and IPv6 Next Header Fields • IPv6-to-IPv6 Network Prefix Translation (NPTv6, NAT66, or IPv6 NAT)	RFC 791 RFC 793 RFC 768 RFC 792 RFC 826 • • RFC 2236 RFC 1918 RFCs 2131, 2132, 1533 RFC 2132 RFCs 1035, 1101 RFCs 1035, 1101 RFCs 3022, 1812 • • RFC 2460 RFC 2461 RFC 2462 RFC 2463 RFC 2373 RFC 2375 RFC 3810 RFC 1700 RFC 6296
Multicast IPv4 Support	Support for IPv4 Multicast in FMAN. Multicast traffic going through the router will be fast forwarded.	RFC 4291 RFC 1112 RFC 3307 RFC 2375 RFC 2710
Unicast, Multicast IPv6	IP Version 6 Addressing Architecture Host Extensions for IP Multicasting Allocation guidelines for IPv6 multicast addresses IPv6 Multicast address assignment Multicast Listener Discovery (MLD) for IPv6 Multicast to Unicast conversion	RFC 4291 RFC 1112 RFC3307 RFC2375 RFC2710

NAT-PT	Support for IPv4/IPv6 protocol translation: UDP and TCP packets only NAPT port translation support ALG not supported "Standard" IPv4 / IPv6 routing (no translation) is supported for connections that don't match the NAT-PT rules	• RFC 2766
VLAN	• Support for unique MAC address to VLAN interfaces	802.1Q
L2 Bridging	IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Bridges. Auto Bridging Support in FMAN	IEEE 802.1D
Advanced QoS	Strict Priority Queuing Class Based Weighted Fair Queuing Token Bucket Rate Limiting Flow Shaping Exception Path Rate Limiting	• LS10xMA-specific feature
PPP over Ethernet (PPPoE)	Client and server protocol that establishes a peer-to-peer relationship, thus allowing service providers to monitor each session. • Point-to-Point protocol • Compression Control Protocol (CCP) • IP Control Protocol (IPCP) • Authentication Protocols: • PAP • CHAP • MS-CHAPv1 • MS-CHAPv2	RFC 2516 • RFC 1661 RFC 1962 RFC 1332 • RFC 1334 RFC 1994 RFC 2433 RFC 2759
IPSEC Offload	The IPSec data processing is offloaded to the FPP architecture, with split of control and data path processing between the GPP and the FPP.	
WiFi Routing/Bridging Offload	WiFi Routing/Bridging functionality is done in FPP to have more CPU headroom on GPP.	

How to Reach Us:
Home Page:
nxp.com

Web Support:
nxp.com/support

11 July 2017