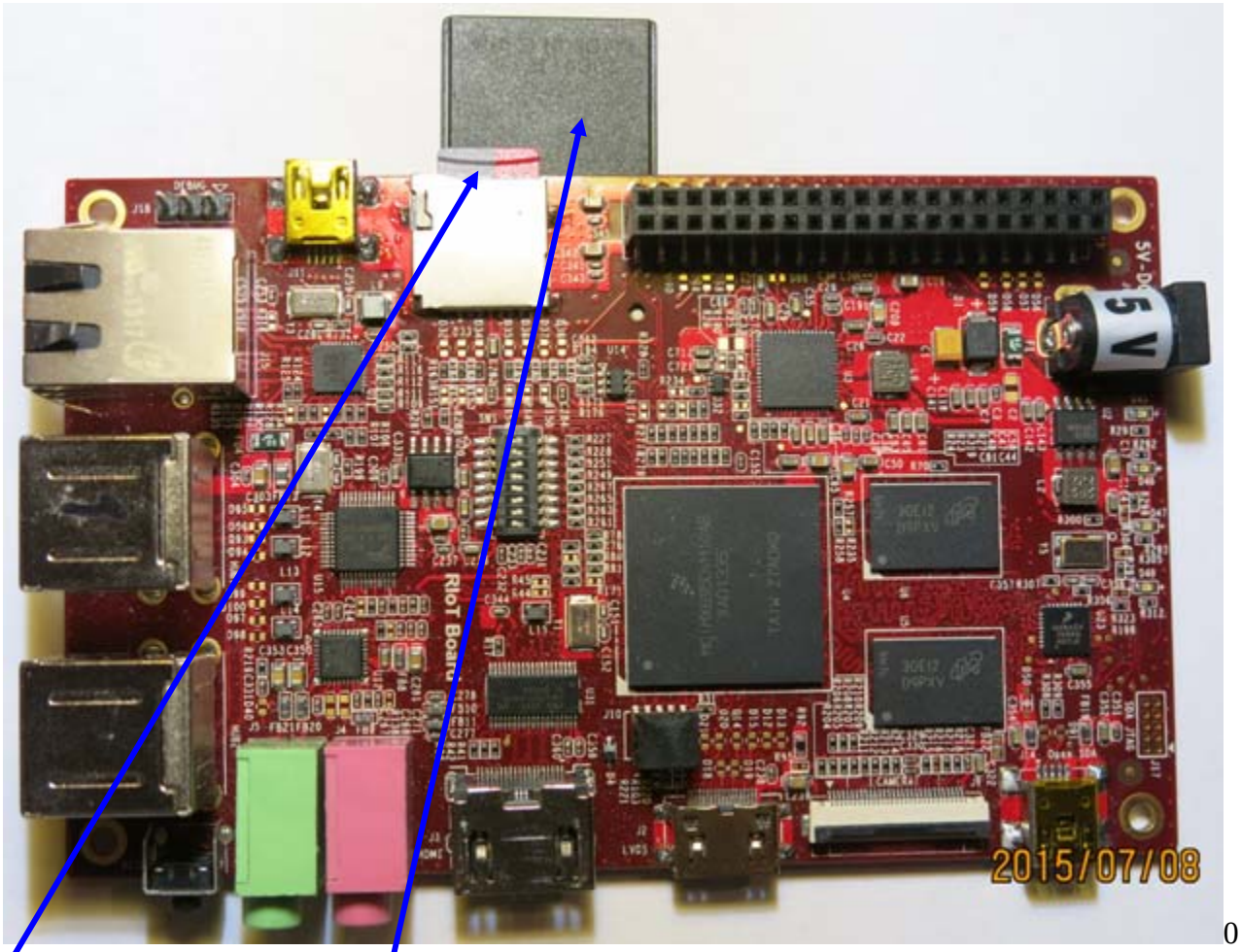
Linux for RIoTboard

Duy-Ky Nguyen **2015-06-01**



0

uSDcard is a lot better than mini SDcard being off board more than half of its size

1. Get Started

1.1. Serial Port J18

Pin	J18	FTDI
1	UART2_TXD	TX
2	UART2_RXD	RX
3	GND	GND

1.2. Config Switch SW1

SW1	1	2	3	4	5	6	7	8
Serial Down Load	OFF	ON	x	x	x	x	x	x
Booting eMMC	ON	OFF	ON	ON	OFF	ON	ON	ON
Boot SD Card	ON	OFF	ON	OFF	OFF	ON	OFF	ON
Boot uSD Card	ON	OFF	ON	OFF	OFF	ON	ON	OFF

2. Flashing Images

Freescale has provided a tool **MfgTools** for flashing binary images : (1)u-boot, (2)kernel + device-tree and (3)root filesystem using Mini-USB J18 (next to Net port RJ45)

2.1. Windows Default Flashing using FreeScale MfgTools

2.1.1. cfg.ini

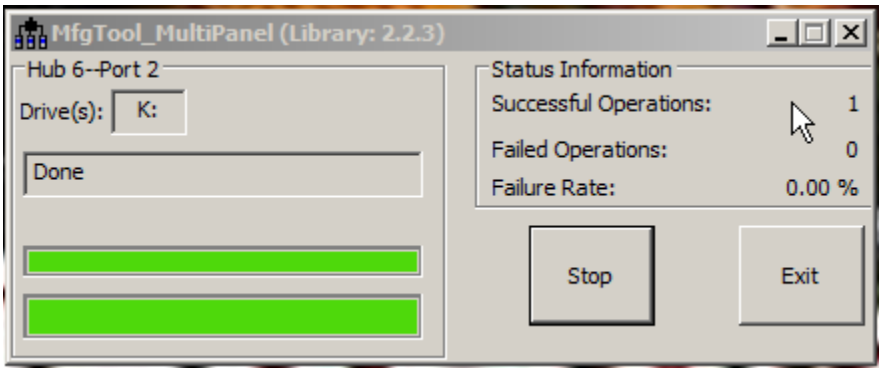
It requires a predefined structure to run

- File "cfg.ini"
  [LIST]
  name = i.MX6SOLO-DEBIAN-RIOT-eMMC
- Folder "profiles\MX6DL Linux Update\OS Firmware"
  - Main File "**ucl2.xml**" must have the name above "name = i.MX6SOLO-DEBIAN-RIOT-eMMC"
  - It has pre-compiled binaries for the tool to run as fully Linux system in flashing new images (1) uboot (2) kernel (3) root-filesystem RamFS
- Folder "profiles\MX6DL Linux Update\OS Firmware\files"
  Storage for new binaries images to be flashed by the tool.

2.1.2. ucl2.xml

It has

- one section of <CFG> with only 2 STATE :
  - BootStrap : to run target as a fully operational Linux/Android system to flash new images
  - Updater : to flash new images
- several section of <LIST> with detailed actions for 2 states above
  - BootStrap : boot u-boot, load kernel and initramfs into RAM and run from there
    - boot pre-compiled u-boot
    - load pre-compiled kernel into RAM
    - load pre-compiled root-filesystem RAMFS into RAM
  - Updater
    - Erase uboot
    - Send partition shell file and do partitioning
    - Send and do flashing NEW u-boot
    - Send "mkfs.ext4" file and do formatting toor-fs
    - Send and do flashing NEW root_FS
    - Send and do flashing NEW uboot boot.src
    - Un-Mount filesystem to save



2.2. User-Control on Linux

It's wise to do on virtual Linux machine as mistake could wipe out the system

2.2.1. Partition Using "FDISK" Tool

- Partition 1 : FAT of 8 M (type c)
- Partition 2 : Linux of the Rest (type 83)

2.2.2. Flashing U-Boot using "DD" Tool

```
# skip IN, seek OUT
printf ">> Erase u-boot segment \n"
sudo dd if=/dev/zero of=$DEV bs=512 seek=2 count=2000

printf ">> Write u-boot to sd card : uboot@0 -> sdcard@2*512 <uboot size>\n"
sudo dd if=$UBT of=$DEV bs=512 seek=2 conv=fsync
```

2.3. Flashing Binaries

Do format FAT and EXT_3 accordingly.

Copy (1) "uImage" and (2) "imx6solo_RIoTboard.dtb" to FAT
Copt root filesystem to EXT_3

3. Root Filesystem

The most popular root filesystem builds have been evaluated (1) Yocto (2) BuildRoot (3) Debian

3.1. Yocto Linux

https://www.yoctoproject.org/downloads "poky-dizzy-12.0.2.tar.bz2"
It takes too long to build, so I tried an prebuild first to see if it's worth to wait. Unfortunately, I could NOT make Samba work on Yocto file-system using "apt-get" to install. I'm able to get it work on Ubuntu system using "apt-get" for Samba.

3.2. Build-Root (BR)

http://git.buildroot.net/buildroot/ : buildroot-2015.02.tar.bz2
This is my most favorite one for small footprint with bare minimum filesystem, BusyBox only. I select "Samba" in BR and it does not work for me. It's the latest Samba-4.

3.3. Debian Wheezy

Based on this article https://olimex.wordpress.com/2014/07/21/how-to-create-bare-minimum-debian-wheezy-rootfs-from-scratch/, I'm able to create a filesystem where Samba works properly after using "apt-get install"

```
# Host pkg
sudo apt-get install qemu-user-static debootstrap binfmt-support
# Debian version : wheezy
targetdir=rootfs
distro=wheezy
# first stage of Debian rootfs
mkdir $targetdir
```

```
sudo debootstrap --arch=armhf --foreign $distro $targetdir

# Copy pkg from Host to Traget for ChRoot to build
sudo cp /usr/bin/qemu-arm-static $targetdir/usr/bin/
sudo cp /etc/resolv.conf $targetdir/etc

## Do ChRoot
sudo chroot $targetdir

#############################################################

# Set Env_Var
distro=wheezy
export LANG=C

# Start stage 2
/debootstrap/debootstrap --second-stage

# Apt list
cat <<EOT > /etc/apt/sources.list
deb http://ftp.uk.debian.org/debian $distro main contrib non-free
deb-src http://ftp.uk.debian.org/debian $distro main contrib non-free
deb http://ftp.uk.debian.org/debian $distro-updates main contrib non-free
deb-src http://ftp.uk.debian.org/debian $distro-updates main contrib non-free
deb http://security.debian.org/debian-security $distro/updates main contrib non-free
deb-src http://security.debian.org/debian-security $distro/updates main contrib non-free
EOT

# Update Debian database
apt-get update

# Install Locales
apt-get install locales dialog
dpkg-reconfigure locales

# Install ntpdate
apt-get install ntpdate

# Install development : toolchain, make, ...
apt-get install build-essential

# Static IP
echo <<EOT >> /etc/network/interfaces
allow-hotplug eth0
# iface eth0 inet dhcp
iface eth0 inet static
       address 11.1.1.11
       netmask 255.255.0.0
       gateway 11.1.1.1
EOT

# Set hostname
echo riotb > /etc/hostname

## exit ChRoot
exit
```

3.4. System Build

There 2 packages under evaluation

- SVN-2591 with u-boot-2009.08 and linux-3.0.35
- SVN-2777 with u-boot-2013.04 and linux-3.10.17 with device-tree

The latter has been used due to it's an update one with device-tree used with the kernel.

The original package has been modified for the following features

- (1) U-boot, (2) kernel and (3) device-tree will be stored as raw binary in the top of un-partitioned flash of 8 MB
- All these 3 images will be updated over network. The existing implementation has u-boot stored as raw binary, but kernel & device-tree stored in FAT and PC tool is used for flashing
- Micro SD will be used as main storage while the existing implement uses mini SD
- NFS boot to support testing kernel , device-tree and drivers

3.5. Tool-Chain : arm-fsl-linux-gnueabi-4.6

https://github.com/embest-tech/fsl-linaro-toolchain

3.6. U-Boot : u-boot.imx

```
make distclean
make mx6solo_RIoTboard_config
```

```
#define CONFIG_EXTRA_ENV_SETTINGS \
       "script=boot.scr\0" \
       "fdt_addr=0x18000000\0" \
       "boot_fdt=try\0" \
       "ip_dyn=yes\0" \
       "console=ttymxc1\0" \
       "fdt_high=0xffffffff\0"     \
       "initrd_high=0xffffffff\0" \
       "mmc dev 1\0" \
       "mmcpart=1\0" \
       "mmcroot=mmcblk1p2\0" \
       "smp=nosmp\0"\
       "bootargs=console=$console,115200 $smp " \
       "video=mxcfb0:dev=hdmi,1280x720M@60,bpp=32 " \
       "video=mxcfb1:off fbmem=10M\0" \
       "mmcargs=setenv bootargs $bootargs root=$mmcroot rootwait rw consoleblank=0\0" \
       "loadbootscript= fatload mmc ${mmcdev}:${mmcpart} $l{oadaddr} ${script};\0" \
       "bootscript=echo Running bootscript from mmc ...; source\0" \
       "ubtfile=RIOTB/_UBT_\0" \
       "bootfile=RIOTB/_zIMG_\0" \
       "fdtfile=RIOTB/_DTB_\0" \
       "fdtBlk=800\0" \
       "lnxBlk=900\0" \
       "mmc.ubt=mmc rescan; mmc dev 1; tftp $loadaddr $ubtfile; mmc write $loadaddr 2 400\0" \
       "mmc.dtb=mmc rescan; mmc dev 1; tftp $loadaddr $fdtfile; mmc write $loadaddr $fdtBlk 100\0"
\
       "mmc.lnx=mmc rescan; mmc dev 1; tftp $loadaddr $bootfile; mmc write $loadaddr $lnxBlk
2c00\0" \
       "mmc.mfs=mmc rescan; mmc dev 1; mmc read 11000000 $fdtBlk 100; mmc read 12000000 $lnxBlk
2c00\0" \
```

```
       "mmc.boot=run mmcargs; run mmc.mfs; bootz 12000000 - 11000000\0" \
       "nfs.opts=nolock\0" \
       "nfs.root=/home/dkn/_NFS/_RFS\0" \
       "nfs.args=setenv bootargs console=${console},${baudrate} ${smp} root=/dev/nfs " \
             "nfsroot=${serverip}:${nfs.root},${nfs.opts} rw ip=dhcp ${morebootargs}\0" \
       "nfs.boot=tftp ${loadaddr} ${bootfile}; tftp ${fdt_addr} ${fdtfile}; run nfs.args; bootz
${loadaddr} - ${fdt_addr}\0" \
       "bootcmd=run mmc.boot\0" \
```

```
#!/bin/bash

printf ">> Host Pkgs\n"

sudo apt-get install wget git-core unzip texinfo libsdl1.2-dev \
gawk diffstat build-essential chrpath

sudo apt-get install sed cvs subversion coreutils texi2html \
docbook-utils python-pysqlite2 help2man make gcc g++ desktop-file-utils \
libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf automake groff curl lzop asciidoc xterm

printf ">> Target pkgs\n"

git clone git://github.com/embest-tech/fsl-linaro-toolchain.git

curl https://raw.githubusercontent.com/android/tools_repo/master/repo > ~/bin/repo

chmod a+x ~/bin/repo
```

3.7. Linux Kernel : zImage

make distclean
make imx_v7_defconfig

Config

- Turn off IPv6
- Turn on NFS

3.8. BusyBox : busybox-1.22

Cross-Compile : arm-fsl-linux-gnueabi-
Static Build
Turn on on all

/usr/sbin/tftp -> busybox
/usr/sbin/httpd -> busybox

3.9. Net Config : /etc/network/interfaces

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
auto eth0
allow-hotplug eth0
```

```
iface eth0 inet dhcp
```

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
auto eth0
allow-hotplug eth0
# iface eth0 inet dhcp
iface eth0 inet static
    address 10.0.0.20
    network 10.0.0.0
    netmask 255.255.255.0
    broadcast 10.0.0.255
    gateway 10.0.0.1
```

3.10. Samba

```
apt-get install samba samba-common samba-common-bin
#/etc/init.d/samba status

smbpasswd –a root

/etc/samba/smb.conf

// # append
[root_FS]
       path = /
       writeable = yes
```

3.11. SSH

On target RIoTboard

```
“apt-get install openssh-server”
```

/etc/rc.local : service ssh start

On Cygwin

```
ssh-copy-id root@ssh-server-IP
```

ssh-keygen -t rsa

Enter to skip SSH passwd ==> No paaswd required using "ssh root@11.1.1.11:".

OTHERWISE a long passwd, 6 chars at lest ???

Provide RB login passwd

ssh-copy-id root@ssh-server-IP

Connect from Cygwin **under .ssh, a MUST**

```
/home/DKN/.ssh/ssh root@RIotb_IP
```

3.12. VSFTPD

Install

```
apt-get install vsftpd
```

Config

```
# /etc/ vsftpd.conf
# If you connect from the internet with local users, you should enable TLS/SSL/FTPS
local_enable=YES
```

```
# Write permissions
write_enable=YES

# Make sure PORT transfer connections originate from port 20 (ftp-data).
connect_from_port_20=YES

ascii_upload_enable=YES
ascii_download_enable=YES
```

```
usermod –d /home/www dkn
```

Run under xinetd

```
# /etc/xinet.d/vsftpd
service ftp
{
        disable                 = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/vsftpd
        per_source              = 5
        instances               = 200
        banner_fail             = /etc/vsftpd.busy_banner
        log_on_success          += PID HOST DURATION
        log_on_failure          += HOST
}
```

3.13. SVN

apt-get install subversion

mkdir __SVN

svnadmin create __SVN

3.13.1. config : Comment out

```
anon-access = read

auth-access = write

password-db = passwd
```

3.13.2. passwd

```
dkn = ???
```

3.13.3. AutoStart

```
# /etc/rc.local
# ps aux | grep svn

sudo svnserve -d -r /home/__SVN
```

3.14. Web Server : BusyBox HTTPD & PHP

Install static BusyBox

```
/usr/sbin/httpd -> ../../bin/busybox
/usr/sbin/tftp -> ../../bin/busybox
```

```
httpd –c /etc/httpd.conf –h /home/www
```

3.14.1. PHP5-CGI PHP5-SQLITE

```
## /etc/httpd.conf

*.php:/usr/bin/php5-cgi
```

```
## /etc/php5/cgi/php.ini

cgi.force_redirect = 0
cgi.redirect_status_env ="yes";
```

```
// phpinfo.php

<?
        phpinfo();
?>
```

3.15. DynDns.com

apt-get install ddclient

username:password

4. Summary

The very HW MAC address (ethaddr in UBoot) is stored some where on the flash in the very first Linux run. So, we have to RE-FORMAT the whole flash to use a new HW MAC address; otherwise network port cannot be activated due to the conflict.

There’re 2 packages evaluated

- UBoot-2009 & Linux-3.0 without device tree
- UBoot-2013 & Linux-3.10 with device tree (*)

There’re 3 system builds evaluated

- Yocto (problem with Samba)
- BuildRoot (problem with Samba)
- Debian Wheezy (*)

There're 3 boot devices evaluated

- On-board flash
- Mini SD Card
- Micro SD Card (*)

There're 3 boot processes

- UBoot in top of flash as raw binary
- Kernel + DTB (Device Tree Binary) in flash as raw binaries or as FAT files
- Root Filesystem in Ext2 partition

So the option is available for kernel either as raw binary or as FAT file. The former has been preferred as it's a lot more convenient for development to update UBoot and kernel over network.

- U-boot is first flashed using "dd" tool on a host, then updated itself over network
- Kernel and DTB are flashed over network
- Root filesystem is flashed on a host

Flash partition

- Top 8M reserved for UBoot, kernel & DTB (raw binary)
- Next 8M for FAT, in case to store kernel & DTB
- The rest for Ext2

The new Linux system includes

- Samba server easy for file management from PC
- Web server using BusyBox HTTPD and dynamic IP via DynDns.com
- SVN server via DynDns.com
- BusyBox Telnet server via DynDns.com
- SSH-Server : Remote maintenance as "root"
- VSFTPD : File Trasnfer for updating Web page

```
Disk /dev/sdb: 63.9 GB, 63864569856 bytes
64 heads, 32 sectors/track, 60906 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000f22c2

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               9          17        8192+   c  W95 FAT32 (LBA)
Partition 1 does not end on cylinder boundary.
/dev/sdb2              17       60906    62351359+  83  Linux
```