ESP-9010
Debian Live System
Building Environment Guide

Revision v0.1.0

Advantech					
Initiated by		Job Title		Signature	
Approved by		Job Title		Signature	

Revision History

Version	Date(dd/mm/yyyy)	Modified by	Status
0.1.0	28/10/2016	Terry	Initial version

Content

1. DEBIAN LIVE SYSTEM

ESP-9010 LMP OS uses Debian Live system as the firmware skeleton. The output firmware image contains Debian maintained stable packages. The system supports boots from USB stick or hard drive.

1.1 Build Environment

ESP-9010 LMP OS is developed on **Debian 8.1.0** (jessie) **x86_64** PC host system or virtual machine (VM). The Linux kernel is **version 3.16.0.4.** To install Debian Live system, please execute "*sudo apt-get install live-build*" in the host system, described at Chap 2. The Debian Live system includes several sub-projects. Below table lists the versions of major ones used in ESP-9010 LMP OS system.

Debian Live Project	Version
live-build	4.0.3-1
live-boot	4.0.2-1
live-config	4.0.4-1

Table 1 Live Project Version

1.2 Live System Configuration

The Debian Live system is a collection of scripts that, in build time, download official packages from Debian main section to construct the firmware image for i386 or x86_64 architecture target. The scripts are separated into three parts: (1) live-build: scripts used to build customized Debian Live system, (2) live-boot: scripts used in early boot phase via the hooker of initramfs, (3) live-config: scripts used in rest boot process after live-boot. The details of all parameters of Debian Live system is out of the scope of this document. Please reference the manual page of "*lb_config*", "*live-boot*", and "*live-config*" as well as the user guide from the Debian web site.

Powered by Debian's rich package support, it is easy to add new package by updating the configuration file without bothering to build each package from scratch for most of cases. Below is the configuration file of Debian Live system for ESP-9010 LMP OS.

```
#!/bin/sh


lb config noauto \
        --architectures amd64 \
        --bootappend-live "boot=live components live-getty persistence noeject nopat ip=frommedia console=ttyS1,115200n81i
acpi_enforce_resources=lax pcie_aspm=off quiet" \
        --mirror-bootstrap ftp://debian.csie.ntu.edu.tw/pub/debian/ \
        --mirror-chroot ftp://debian.csie.ntu.edu.tw/pub/debian/ \
        --mirror-chroot-security ftp://debian.csie.ntu.edu.tw/pub/debian-security/ \
```

```
--mirror-binary ftp://debian.csie.ntu.edu.tw/pub/debian/ \
--mirror-binary-security ftp://debian.csie.ntu.edu.tw/pub/debian-security/ \
"${@}"
```

Table 2 Live System Configuration

Except for Debian default installed packages, the extra packages are listed in "<workspace>/config/package-lists/". See below table for the details.

```
live-boot live-config live-config-systemd openipmi pciutils ipmitool less memtester dmidecode bridge-utils i2c-tools lm-sensors usbutils openssh-server strace tree acpi-support ethtool acpidump acpi acpitool tcpdump ncurses-term lrzsz screen tshark irqbalance iasl setserial expect ntpdate vim hdparm bzip2 unzip python file p7zip-full gawk tftp-hpa psmisc libc6-i386 lib32gcc1 ecryptfs-utils parted
```

Table 3 Live System Package List

If user wants to add new package, he or she may want to navigate all available packages first by executing "apt-cache dumpavail" in the host system.

In order to ensure the download speed, user could modify the "--mirror" prefixed options by choosing the distribution mirror sites in your nearby. The worldwide Debian mirror sites are listed in the web site of http://www.debian.org/mirror/list.

1.3 Source Code Structure

The LMP OS source tree is structured as below table.

Note: the entries marked in blue are files and directories generated in build time and the red ones are the important configuration files and packages the customers need to reference.

<workspace>	LMP source code work space
|_auto	Folder that contains the configuration for live-build
| |_build	Wrapper script of "lb build" command with customized settings
| |_clean	Wrapper script of "lb clean" command with customized settings
| |_config	Wrapper script of "lb config" command with customized settings
|_cache	Folder that contains the downloaded cache files
|_live-cache.tar.bz2	Tarball of live system cache files, preserved to accelerate build process
|_chroot	Filesystem in image
|_config	Folder that contains additional files to be put in firmware image
| |_common	Common configuration file for all stages
| |_bootstrap	Configuration file generated for live-build bootstrap stage
| |_chroot	Configuration file generated for live-build chroot stage
| |_binary	Configuration file generated for live-build binary stage
| |_source	Configuration file generated for live-build source stage

|_hooks	Scripts to update contents in binary and/or other stages
|_includes.binary	Customized bootstrap configuration files
|_includes.chroot	Additional files to be installed in target rootfs
|_package-lists	List of Debian-oriented package to be installed
|_packages.chroot	List of additional package (with deb suffix) to be installed
|_<misc>	
_Makefile	Entity Makefile to build LMP Live system
_binary	Folder that contains bootloader , kernel, initrd, and rootfs images
_ live-image-amd64	Firmware image to be programming to USB stick and/or hard drive
_ live-image-amd64.contents	List of binary files in binary folder
_build.log	Build time log of live-build
_ live-image-amd64.packages	List of package and version packed in firmware image
_chroot.packages.install	List of package installed in target
_chroot.packages.live	List of package installed in target for booting Live system
_persistence.conf	Define which directories to be persistent
_version	Version
_src	Folder that contains add-in packages and patch files
|_fastpath	Data Plane switch
|_fastpath2	Control Plane switch
|_ifupdown	Used to configure network interfaces based on scripts in /etc/network/
|_linux-kernel	Customized Linux kernel, configuration file, and patch file
|_sdk-xgs-robo	Broadcom SDK, configuration file, and patch files
|_sys-utils	The collection of Advantech-developed utilities
|_igb-driver	Intel gigabit Ethernet NIC adapter driver
|_memtest86+	Memory test utility
|_lfdk	Add lfdk utility (RU-like utility for Linux)
|_<misc>	
_<misc>	

Table 4 Source Code Structure

1.4 Build Firmware Image

Generate LMP firmware image from scratch:

```
$ cd <workspace>
$ make disclean                    Clean up all files of specific package
$ make all                         Build all add-in/customized packages
$ make imgclean                    Clean up LMP firmware image and related files
$ make image                       Generate LMP firmware image
```

Re-build specific package and generate new LMP firmware image:

```
$ cd <workspace>
$ make <package>                    Re-build specific package
$ make <package>.rebuild            Make disclean and make all
$ make <package>.install            Update package installation files in chroot
```

Configure Linux kernel parameters:

```
$ cd <workspace>
$ make linux-kernel config          Run Linux menuconfig
```

Note: There are more detailed make commands defined in <workspace>/Makefile.

1.5 Install Firmware Image to USB Stick or Hard Drive

The generated firmware image (live-image-amd64) contains a VFAT partition and the syslinux boot-loader. After user generates the firmware image, it is ready to be written to a USB stick or hard drive by:

```
$ cd <workspace>
$ make install DEVICE=/dev/sdX      Install LMP firmware image to hard drive
```

Note: <sdX> could be sda, sdb, or others, depending on the existence of other hard drives connected to mSTATA/STATA interface, or USB sockets.

Note: this operation will definitely overwrite the contents on the USB stick or hard drive.

1.6 Bootup Sequence

To enter to the BIOS, press DEL or F2. You can modify the bootup sequence and indicate to boot from specific devices (like mSATA1/mSATA2/USB). The configuration is saved in BIOS. BIOS will load the syslinux boot menu of the specific device. If specific device is broken, it can't load boot menu of the device. You can change Boot Option to load boot menu of another device.

Boot Option:

mSATA1: P0: SQF-SHMM1-32G-S7C

mSATA2: P1: SQF-SHMM2-32G-S9C

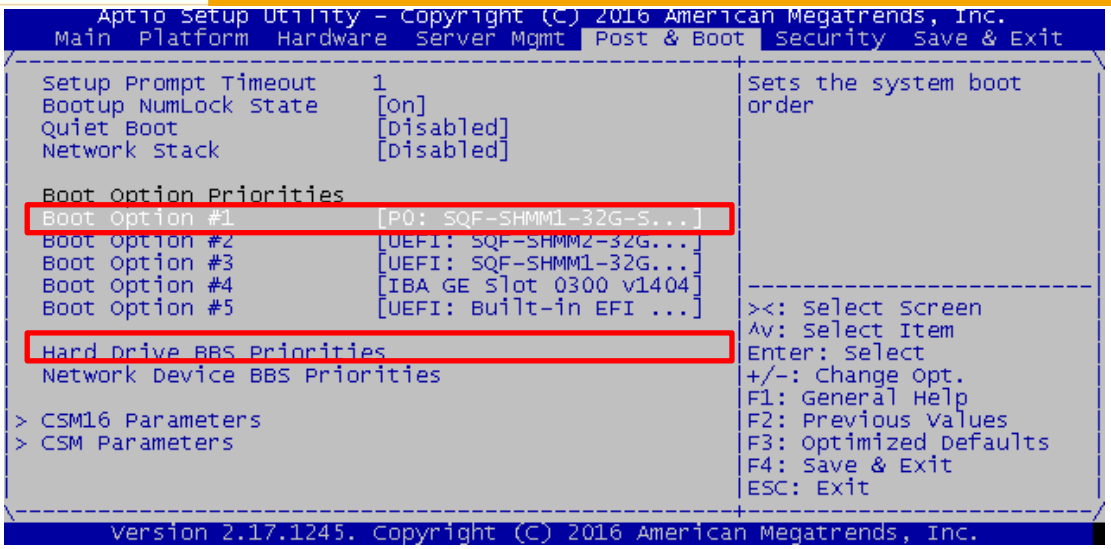


Figure 1 BIOS Boot Option

1.7 Syslinux Boot Menu

The bootstrap of Debian live system is syslinux that provides multi-boot capability. It is a collection of boot loaders capable of booting from different devices. Below figure is the snapshot of LMP boot menu customized for ESP-9010.

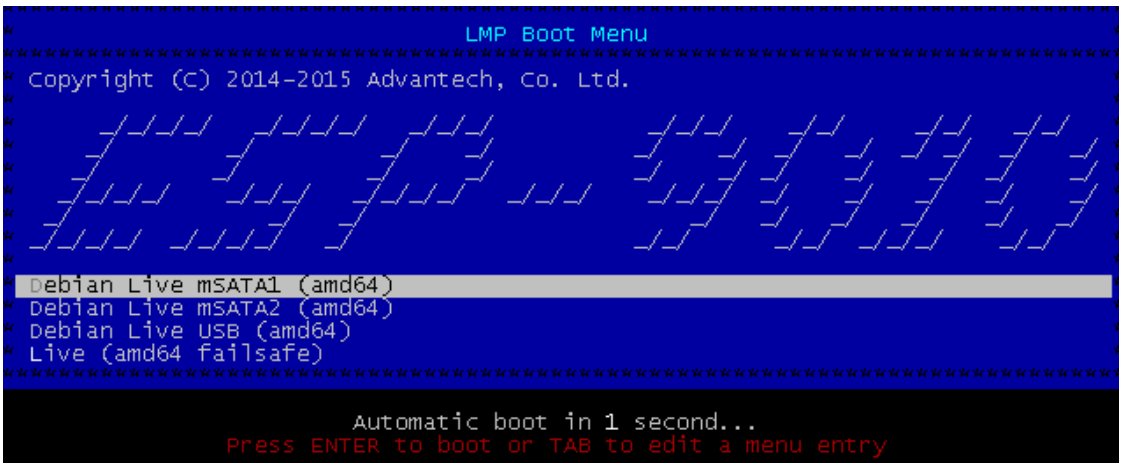


Figure 2 LMP Boot Menu

The menu is separated into four parts:

- Title: this is “LMP Boot Menu”.
- Banner: Advantech copyright and product fancy banner.
- Boot options:
  - “Debian Live mSATA1 (amd64)”: normal booting. Loading Linux Kernel from mSATA1. (Default)
  - “Debian Live mSATA2 (amd64)”: normal booting. Loading Linux Kernel from mSATA2.
  - “Debian Live USB (amd64)”: normal booting. Loading Linux Kernel from USB.
  - “Live (amd64 failsafe)”: fail-safe booting.
- Message bar:

- How long to wait until booting “Debian live (amd64)” automatically”. The default value is 3 seconds. The timer could be cancelled by user input. Once user does this, he/she has to manually select boot option or press TAB key to adjust boot parameters.

If it is required to configure syslinux parameters, title, banner, or boot menu, please update the configuration files or add new ones under “<workspace>/config/includes.binary/syslinux”.

The selected boot option is available only at the current time. If you would like to set mSATA2 to be default booting, please refer to **chap 6.6 Boot from mSATA1/mSATA2** in quick start guide.

1.8 How to Start Build Firmware Image

The steps of development setting:

Step 1. Debian Environment Prepare (Chap 2)

Step 2. If needed, VirtualBox Environment Prepare (Chap 3)

Step 3. Modify Script for live build (Chap 4)

Step 4. Change to gcc version to v4.8 due to fastpath compatibility
#sudo update-alternatives --config gcc

Step 5. Start to do live build image. If building successfully, firmware image (live-image-amd64) will be generated at root of LMP source code work space.
make all
make image

Step 6. Install firmware image to USB Stick or Hard Drive
For example, if usb stick or hard drive is mounted at /dev/sdc
make install DEVICE=/dev/sdc

2. DEBIAN ENVIRONMENT PREPARE

It describes which packages are needed to install for image building.

Update source.list

- $ sudo vim /etc/apt/sources.list

  # add two source download list
  deb http://http.debian.net/debian jessie main contrib non-free
  deb-src http://http.debian.net/debian jessie main contrib non-free

- $ sudo apt-get update

Enable sudo privilege without passwd for image building

- open a terminal
- enter "su" then enter root password
- apt-get install sudo
- enter "visudo"
- add a new line "<account> ALL(ALL:ALL) NOPASSWD:ALL"
  - user ALL=(ALL:ALL) NOPASSWD:ALL

Install packages

- mandatory packages
  - sudo apt-get install build-essential linux-headers-`uname -r` (for building debian linux package)
  - sudo apt-get build-dep linux (for building debian linux package)
  - sudo apt-get install live-build live-boot live-config (for debian live build)
  - sudo apt-get install libncurses5-dev (for linux menuconfig)
  - sudo apt-get install libc6-dev-i386 (for 32-bit packages, e.g. memtest86+)
  - sudo apt-get install fakeroot kernel-package (for building linux kernel related Debian packages)
  - sudo apt-get install iconx
  - sudo apt-get install noweb
  - sudo apt-get install nowebm
  - sudo apt-get install usbutils (for USB device ID mapping)
  - sudo apt-get install zlib1g-dev libftdi-dev libusb-dev libftdi1 libpci-dev (for flashrom)
  - sudo apt-get install ecryptfs-utils

    - sudo apt-get install doxygen
    - sudo apt-get install autoconf
    - sudo apt-get install libssl-dev
- download and install manually, if can't do apt-get install
    - wget http://ftp.us.debian.org/debian/pool/main/i/icon/iconx_9.4.3-4.2_amd64.deb
    - wget http://ftp.us.debian.org/debian/pool/main/n/noweb/noweb_2.11b-9_amd64.deb
    - wget http://ftp.us.debian.org/debian/pool/main/n/noweb/nowebm_2.11b-9_all.deb
- optional packages
    - sudo apt-get install vim (for editor)
    - sudo apt-get install tree (for directory navigation)
    - sudo apt-get install git-core gitg subversion (for version control)
    - sudo apt-get install gdb (for debugging)
    - sudo apt-get install php5-cli php5-json php5-gd php-fpdf (for Phoronix test suite)
    - sudo apt-get install squashfs-tools (for mksquashfs)
    - sudo apt-get install yum
    - sudo apt-get install mock (for sudo without password)
    - sudo apt-get install python-pip (for python colorlog)
    - sudo pip install --upgade colorlog (for python colorlog)

Install packages for gcc4.8

- sudo apt-get install gcc-4.8-multilib
- sudo apt-get install g++-4.8-multilib
- Reference
    - http://askubuntu.com/questions/453681/gcc-wont-link-with-m32

Change gcc from 4.9 to 4.8

The gcc 4.8 is used to build code due to fastpath compatibility.

- sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 30
- sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 40
- sudo update-alternatives --config gcc

Remove unused/out-of-date packages

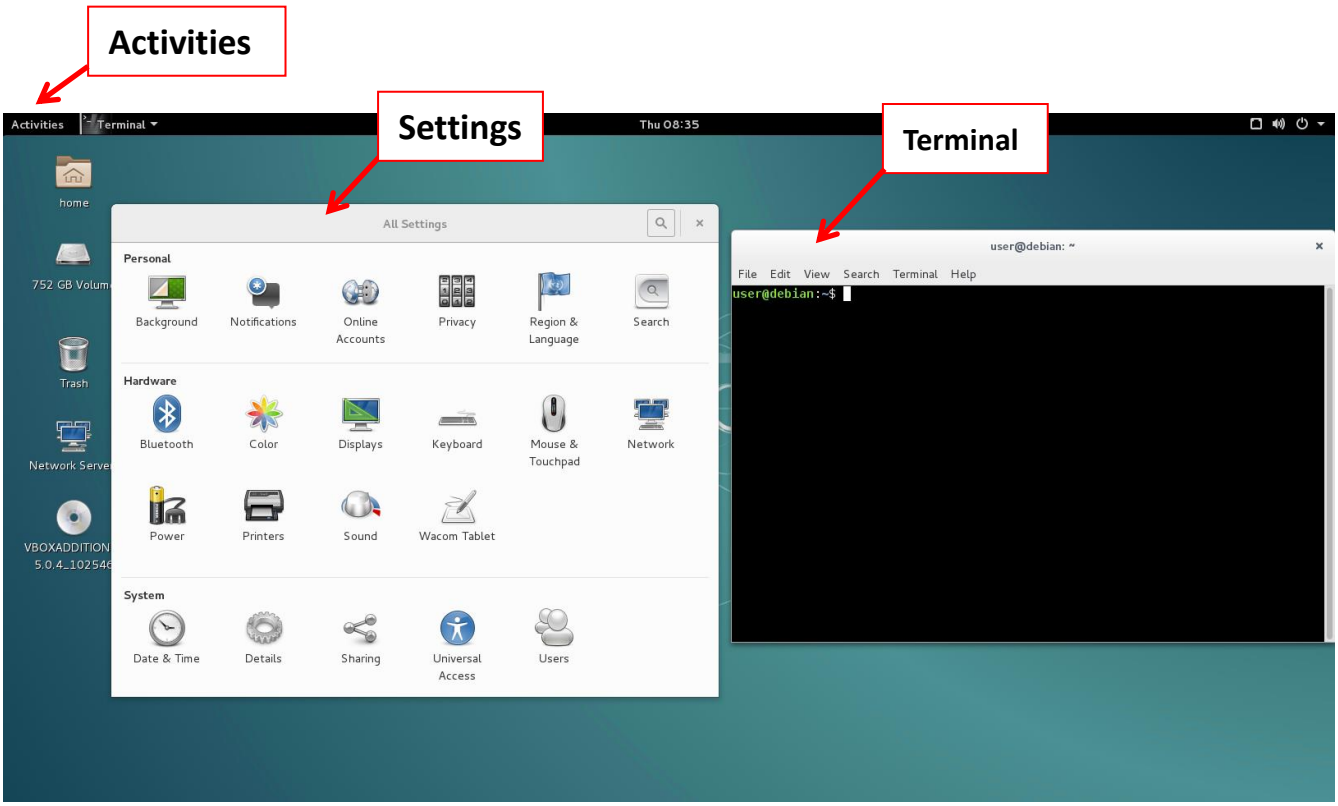
- open a terminal
- remove out-of-date packages
    - sudo apt-get autoclean
    - sudo apt-get autoremove
- remove old linux kerenels
    - dpkg --get-selections | grep linux

Update PCI & ISB ID to description database

- sudo update-pciids
- sudo update-usbids

3. VIRTUALBOX ENVIRONMENT PREPARE

If you use VirtualBox (Virtual Machine) to develop and build code, the below steps will let development easier. Download VirtualBox from https://www.virtualbox.org/



Adjust system settings

- click "Activities" in top left corner then enter "Settings" and start below tunnings
- disable screen lock
  - click "Privacy->Screen Lock" and turn off "Automatic Screen Lock"
- disable notifications
  - click "Privacy->Screen Lock" and turn off "Show Notifications"
- disable blank screen
  - click "Power" and set "Blank screen" to "Never"
- disable auto start program
  - click "Details->Removable Media" and select "Never prompt or start programs on media insertion" checkbox
- enable auto login
  - click "Users->Unlock", enter root password then turn on "Automatic Login"
  - click "Users->Unlock" again
- turn off speaker
  - click "Sound" and turn off "Output volume"
- enable NTP

- click "Date and Time->Unlock", enter root password then turn on "Automatic Time Zone"

Install VirtualBox guest additions

- click "Device" in toolbar -> “Install Guest Additions" in top bar
- click “Cancel” button if auto-run window pops up
- open terminal then enter “sudo sh /media/cdrom/VBoxLinuxAdditions.run”
- enter "sudo eject /dev/sr<cdrom index>"

Enable application icon to desktop

- enable desktop icon
  - click "Activities" in top left corner then enter “Tweak Tool"
  - click “Desktop” in left sidebar and turn on “Icon on Desktop"
  - select/deselect icons you want to show on desktop

Adjust terminal preference

- open a terminal
- change to "white on black" theme
  - click "Edit->Profile Preferences" in terminal's top bar then click "Colors" tag
  - unselect "Use colors from system theme"
  - select "White on black" of "Built-in schemes"
- disable scrolling line limitation
  - click "Edit->Profile Preferences" in terminal's top bar then click "Scrolling" tag
  - deselect "Limit scrllback to" of "Scrollback"
- enable colorful shell prompt
  - uncomment "force_color_prompt=yes" in <home>/.bashrc
- enable colorful vim output
  - uncomment "syntax on" in /etc/vim/vimrc

Mount workspace disk

- mkdir /home/user/workspace
- chdir 777 /home/user/workspace
- add to /etc/fstab to auto-mount disk
  - /dev/sdb /home/user/workspace ext3 defaults,errors=remount-ro 0 0

4. MODIFY SCRIPT FOR LIVE BUILD

In Debian8.1, we must modify building script for live build.

1. Replace ${_BOOTLOADER} with SYSLINUX

#sudo vim /usr/lib/live/build/binary_hdd

dd if=chroot/usr/lib/${_BOOTLOADER}/mbr.bin of=${FREELO} bs=440 count=1

replaced by

dd if=chroot/usr/lib/SYSLINUX/mbr.bin of=${FREELO} bs=440 count=1

This is because the syslinux version in jessie expects mbr.bin to be found in /usr/lib/**SYSLINUX**/mbr.bin, but ${_BOOTLOADER} of Debian8.1 script is **syslinux.**

2. **Add "sync" to /usr/lib/live/build/binary_hdd**

```
if [ -n "${_SYSLINUX_INSTALLER}" ]
then
        case "${LB_BUILD_WITH_CHROOT}" in
                true)
                        Chroot chroot "${_SYSLINUX_INSTALLER}"
                        ;;
                false)
                        ${_SYSLINUX_INSTALLER}
                        ;;
        esac
fi

sync; sync; sync; sync; sync
umount chroot/binary.tmp
rmdir chroot/binary.tmp
```