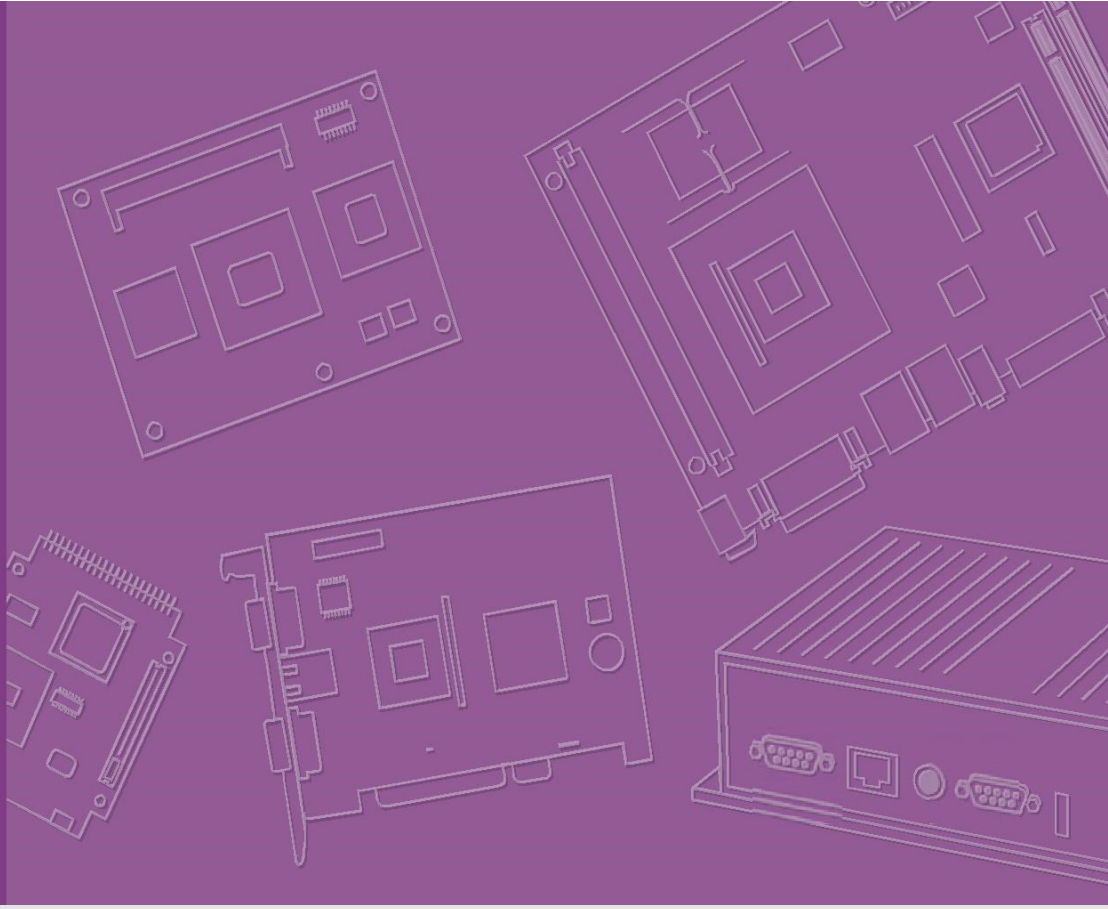

User Guide

Yocto Linux

Board Support Package Ver.5
For i.MX6 series

ADVANTECH
Enabling an Intelligent Planet

Table of Contents

Chapter 1

Getting Started

1. Getting Started

1.1 Prerequisites

All operations in this guide are based on Ubuntu 12.04 LTS 64bit only. First please install Ubuntu 12.04 LTS 64bit* with minimum 2GB memory.

*** ubuntu-12.04.1-desktop-amd64.iso**

1.1.1 To install required packages

Please login and perform the following commands:

```
sudo apt-get install ssh
sudo apt-get install ia32-libs libx11-dev:i386 libreadline6-dev:i386 \
   libgl1-mesa-glx:i386 zlib1g-dev:i386 uuid-dev:i386 liblzo2-dev:i386 \
   libncurses5-dev:i386
sudo apt-get install \
   bison build-essential ccache dpkg flex gcc g++ gettext intltool \
   libarchive-zip-perl libfreetype6-dev libdbus-glib-1-dev liborbit2-dev \
   libxml2-dev libx11-dev libgtk2.0-dev liblzo2-2 libtool m4 \
   patch rpm tcl uboot-mkimage uuid zlib1g zlib1g-dev \
   git gnupg flex bison gperf build-essential  zip \
   curl libc6-dev libncurses5-dev x11proto-core-dev libx11-dev:i386 \
   libreadline6-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev g++-multilib \
   mingw32 tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386 \
   gcc-4.6 g++-4.6 cpp-4.6 gcc-4.6-multilib uuid-dev liblzo2-dev \
   uboot-mkimage libarchive-zip-perl \
   wget git-core unzip texinfo gawk diffstat build-essential chrpath \
   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
sudo apt-get install libncurses5-dev:i386 liblzo2-dev:i386 uuid-dev:i386
sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/libGL.so
tar zcvf ~/usr_lib_i386-linux-gnu_for_Building_Android_KK.tar.gz \
     /usr/lib/i386-linux-gnu/{libuuid.a,libuuid.so,liblzo2.so,liblzo2.a}
sudo apt-get install uuid-dev liblzo2-dev
sudo tar zxvf ~/usr_lib_i386-linux-gnu_for_Building_Android_KK.tar.gz -C /
```

1.1.2 To install JDK

Please download "jdk-6u45-linux-x64.bin" manually, put it to directory ~/FILES/ and perform the following commands:

```
cd /usr/lib
sudo ~/FILES/jdk-6u45-linux-x64.bin
sudo mkdir jvm; cd jvm
sudo mv ../jdk1.6.0_45 .
cd jdk1.6.0_45/
sudo update-alternatives --install /usr/bin/java    java    /usr/lib/jvm/jdk1.6.0_45/jre/bin/java    2
sudo update-alternatives --install /usr/bin/javac   javac   /usr/lib/jvm/jdk1.6.0_45/bin/javac   2
sudo update-alternatives --install /usr/bin/jar     jar     /usr/lib/jvm/jdk1.6.0_45/bin/jar     2
sudo update-alternatives --install /usr/bin/javap   javap   /usr/lib/jvm/jdk1.6.0_45/bin/javap   2
sudo update-alternatives --install /usr/bin/javadoc javadoc /usr/lib/jvm/jdk1.6.0_45/bin/javadoc 2
sudo update-alternatives --config javap
sudo update-alternatives --config javadoc
sudo update-alternatives --config java
sudo update-alternatives --config javac
sudo update-alternatives --config jar
cd ~/
sudo sh -c "echo "JAVA_HOME=/usr/lib/jvm/jdk1.6.0_45" >> /etc/environment"
```

1.2 Conventions

${PREBUILT_IMAGE} : compressed prebuilt image (*.img.gz)

${BSP_TARBALL} : BSP tarball (*.tgz)

${BSP_HOME} : home directory of the BSP

${BDIR} : build directory (e.g. build-x11)

${MX6PROC} : i.MX6 Processor
mx6q for iMX6 Quad Core / Dual Core
mx6dl for iMX6 Dual Lite / Solo

${IMX6PROC} : i.MX6 Processor
imx6q / imx6dl

${BOARD} : available target boards list below
ubc220 / rom3420 / rom5420 / rom7420 / rsb4410 / ubcds31

${BOARD_REV} : board revision
a1 / a2 / b1

${MC}** : machine code combined with **${IMX6PROC}${BOARD}${BOARD_REV}
for example,
imx6dlubc220a1 for UBC-220 A1
imx6qrom3420a1 for ROM-3420 A1
imx6qrom5420b1 for ROM-5420 B1
imx6qrsb4410a2 for RSB-4410 A2

${MEM_SIZE} : memory size
1G / 2G / 512M

${SD_DEVICE} : device name of SD card in Ubuntu (e.g. /dev/sdf)

${SDCARD_IMAGE} : sdcard image built by bitbake (*.sdcard)

${UBOOT} :u-boot version(e.g. 2013.04-r0)

${KERNEL} : linux kernel version(e.g. 3.14.28-r0)

${POKY} : Yocto poky version(e.g. 1.7)

debug console / serial console
serial terminal program (e.g. minicom, putty, teraterm ...) that serial port is configured to 115200 8N1

terminal console
terminal program (e.g. gnome-terminal, xfce4-terminal ...)

1.3 Introducing BSP

The BSP is based on Yocto Project with Freescale enhanced features for i.MX6, plus specific target board features from Advantech Inc..

1.3.1 Naming Rule

The tarball/prebuilt image name is consist of the model name followed by "LB" or "LI" plus version number and released date.

For example, 7420A1LBV5080_2016-03-09.tgz which "7420A1" stands for RSB-**7420 A1**, "LB" is acronym of **L**inux **B**SP, "V5080" stands for **V**ersion **5**.**080**;.

For example, 7420A1LIV5080_DualQuad_2016-03-09.img.gz which "LI" is acronym for prebuilt **L**inux **I**mage, DualQuad means this image is fit for Dual Core/Quad Core.

1.3.2 BSP tarball

Unpack BSP tarball to home directory by performing the following command:

```
$ tar xvf ${BSP_TARBALL} -C ~/
```

(Every BSP with different version contains an unique folder, e.g. after unpacking 7420A1LBV5080_2016-03-09.tgz to home directory, the directory , ~/imx6LBV5080_2016-03-09 is the BSP's home folder.)

The description of some important folders list below:

sources/

meta-advantech/ : meta layer by Advantech

meta-fsl-*/ : meta layer by Freescale

fsl-setup-release.sh : to create one new build environment

setup-environment : to continue an exist build environment

1.3.3 Prebuilt image

Perform the following command to build one boot-up SD card

```
$ gunzip -c ${PREBUILT_IMAGE} | dd of=${SD_DEVICE} bs=1M
```

1.4 Build Instructions

1.4.1 To create one new build environment

1) Perform the following commands in terminal console

```
$ cd ${BSP_HOME}
$ MACHINE=${MC} source fsl-setup-release.sh -b ${BDIR} -e x11
```

2) You need to read and accept the EULA.

```
Do you accept the EULA you just read? (y/n)
```

1.4.2 To continue an exist build environment

Perform the following commands in terminal console

```
$ cd ${BSP_HOME}
$ source setup-environment ${BDIR}
```

1.4.3 To build sdcard image

1) To create/continue a build environment
2) Perform the following command in terminal console

```
$ bitbake fsl-image-qt5
```

3) The file, fsl-image-qt5-${MC}.sdcard, will be located in directory, ./tmp/deploy/images/${MC}, while building process finished successfully.

1.4.4 To build toolchain installer

1) To create/continue a build environment
2) Perform the following command in terminal console

```
$ bitbake fsl-image-qt5 -c populate_sdk
```

3) The installer, poky-eglibc-x86_64-fsl-image-qt5-cortexa9hf-vfp-neon-toolchain-${POKY}.sh, will be located in the directory "./tmp/deploy/sdk".

1.4.5 To build u-boot

1) To create/continue a build environment
2) Perform the following command in terminal console

```
$ bitbake u-boot-imx
```

3) The two files, u-boot_crc.bin & u-boot_crc.bin.crc, will be located in directory, ./tmp/deploy/images/${MC}.

1.4.6 To build linux kernel

1) To create/continue a build environment
2) Perform the following command in terminal console
   A. to show up menuconfig
      ```
      $ bitbake linux-imx -c menuconfig
      ```
   B. to do build
      ```
      $ bitbake linux-imx
      ```
3) The two files, zImage & **${IMX6PROC}-${BOARD}-${BOARD_REV}**.dtb, will be located in directory, ./tmp/deploy/images/**${MC}**.

1.5 Creating boot-up on-board flash from prebuilt image

1.5.1 To create one boot-up SD card

Perform the following command in terminal console

```
$ gunzip -c ${PREBUILT_IMAGE} | dd of=${SD_DEVICE} bs=1M
```

1.5.2 To transfer whole system to on-board flash

1) Boot up from SD card
2) Perform the following commands in debug console
   ```
   # cd /mk_inand
   # ./mksd-linux.sh /dev/mmcblk0
   ```
3) press y followed by Enter, if following message shows up:
   ```
   All data on /dev/mmcblk0 now will be destroyed! Continue? [y/n]
   ```
4) While "[Done]" shows up means the transferring is finished.

1.6 Creating boot-up on-board flash from built sdcard image

1.6.1 To create one boot-up SD card

Perform the following commands in terminal console

```
$ pushd ${BSP_HOME}/${BDIR}/tmp/deploy/images/${MC}/
$ dd if=${SDCARD_IMAGE} of=${SD_DEVICE} bs=1M
$ popd
```

1.6.2 To transfer whole system to on-board flash

1) Boot up from SD card

2) Insert USB stick that contains **${SDCARD_IMAGE}**, USB stick will be auto mounted to /media/sda1.

3) Perform the following commands in debug console

```
# umount /media/mmcblk0p?
# cd /media/sda1
# dd if=${SDCARD_IMAGE} of=/dev/mmcblk0 bs=4M conv=fsync
# P2START=$(fdisk -lu | grep mmcblk0p2 | awk '{print $2}')
# echo -e "d\n2\nn\np\n2\n${P2START}\n\nw\n" | fdisk -u /dev/mmcblk0
# umount /media/mmcblk0p2
# e2fsck -f -y /dev/mmcblk0p2
# resize2fs /dev/mmcblk0p2
# poweroff
```

Chapter 2

Customization

2. Customization

2.1 Package addition

2.1.1 To add tcf-agent & openssh-sftp-server

1) Navigate to the directory where fsl-image-adv.inc located

```
$ cd ${BSP_HOME}/sources/meta-advantech/recipes-fsl/images
```

2) Add following line to fsl-image-adv.inc

```
IMAGE_INSTALL += " tcf-agent openssh-sftp-server "
```

3) Continue an exist build environment and build sdcard image (ref. 1.4.2, 1.4.3)

2.1.2 To add chromium browser

1) Navigate to the directory where local.conf located

```
$ cd ${BSP_HOME}/${BDIR}/conf
```

2) Add following two lines to local.conf

```
CORE_IMAGE_EXTRA_INSTALL += "chromium"
LICENSE_FLAGS_WHITELIST="commercial"
```

3) Continue an exist build environment and build sdcard image (ref. 1.4.3)

2.2 Setting up SDK

1) Please follow 1.4.4 to build one toolchain installer
2) Perform the following command in terminal console

```
$ cd ${BSP_HOME}/${BDIR}/tmp/deploy/sdk
$ sudo \
./poky-eglibc-x86_64-fsl-image-qt5-cortexa9hf-vfp-neon-toolchain-${POKY}.sh
```

3) Enter directory or press Enter while following question shows up:

```
Enter target directory for SDK (default: /opt/poky/1.7):
```

4) Just press Enter while following question shows up:

```
You are about to install the SDK to "/opt/poky/1.7". Proceed[Y/n]?
```

5) While following message shows up means the SDK is ready.

```
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

2.3 Setting up cross compiling environment

1) SDK has been set up. (ref. 2.2)
2) Perform the following command in terminal console

```
$ source /opt/poky/${POKY}/environment-setup-cortexa9hf-vfp-neon-poky-linux-gnueabi
```

2.4 Building & updating u-boot manually

2.4.1 To build u-boot

1) The cross compiling environment has been set up. (ref. 2.3)
2) Make one copy from Yocto working directory

```
$ mkdir -p ~/code
$ pushd ${BSP_HOME}/${BDIR}/tmp/work/${MC}-poky-linux-gnueabi/
$ rm -rf ~/code/u-boot-imx
$ cp -a ./u-boot-imx/${UBOOT}/git ~/code/u-boot-imx
$ popd
```

3) Configure u-boot

```
$ cd ~/code/u-boot-imx
$ make distclean
$ make ${MX6PROC}${BOARD}${BOARD_REV}_${MEM_SIZE}_config
```

4) Start building u-boot

```
$ make -j4 LDFLAGS=
```

5) The two files, u-boot-crc.bin & u-boot-crc.bin.crc, are located in directory "`~/code/u-boot-imx`".

2.4.2 To update u-boot to target device

1) Perform the following command to transfer to exist boot-up SD card

```
$ dd if=u-boot_crc.bin.crc of=${SD_DEVICE} bs=512 seek=2 conv=fsync
$ dd if=u-boot-crc.bin of=${SD_DEVICE} bs=512 seek=3 conv=fsync
```

2) Make sure new u-boot does work then perform the following commands to transfer to on-board flash

```
$ dd if=u-boot_crc.bin.crc of=/dev/mmcblk0 bs=512 seek=2 conv=fsync
$ dd if=u-boot-crc.bin of=/dev/mmcblk0 bs=512 seek=3 conv=fsync
```

2.5 Building & updating kernel/modules/dtb manually

2.5.1 To build kernel/modules/dtb

1) The cross compiling environment has been set up. (ref. 2.3)

2) Make one copy from Yocto working directory

```
$ mkdir -p ~/code
$ pushd ${BSP_HOME}/${BDIR}/tmp/work/${MC}-poky-linux-gnueabi/
$ rm -rf ~/code/linux-imx
$ cp -a ./linux-imx/${KERNEL}/git ~/code/inux-imx
$ popd
```

3) Configure linux kernel

```
$ cd ~/code/linux-imx
$ make distclean
$ make imx_v7_adv_defconfig
$ make menuconfig PKG_CONFIG_SYSROOT_DIR= PKG_CONFIG_PATH=
```

4) Start building linux kernel

```
$ make -j4 zImage LOADADDR=0x10008000 LDFLAGS=
```

5) The kernel image file, zImage, is located in the directory "./arch/arm/boot/".

6) Start building kernel modules

```
$ make -j4 modules LDFLAGS=
```

7) Copy all modules to a temporary rootfs directory, "~/temp/rootfs"

```
$ make modules_install INSTALL_MOD_PATH=~/temp/rootfs
```

8) Start building device tree blob

```
$ make -j4 ${IMX6PROC}-${BOARD}-${BOARD_REV}.dtb
```

9) The device tree blob, **${IMX6PROC}-${BOARD}-${BOARD_REV}**.dtb, is located in the directory "./arch/arm/boot/dts/".

2.5.2 To update kernel/modules/dtb to target device

1) Copy zImage & **${IMX6PROC}-${BOARD}-${BOARD_REV}**.dtb to the 1st partition of SD card

2) Copy modues to the 2nd partition of SD card.

3) Make sure all new linux kernel, device tree and kernel modules work well, then copy all of them to the on-board flash

Chapter 3

System Recovery

3. System Recovery

Please refer to 1.5 & 1.6 to create a boot-up SD card and transfer whole system to on-board flash.