6AO.1.0 Application Notes

Back to 6AO.1.0_Release_Notes

Contents

QSPI NOR/eMMC partitions

In this release, the xloader and bootloader goes into QSPI and rest of the images go in to eMMC partitions. All TI software has been moved to the vendor partition. The table below summarizes the partition info:

Partition Info

Partition Name	eMMC/QSPI	Binary to flash
xloader	QSPI	MLO
bootloader	QSPI	u-boot.img
boot	eMMC	boot.img (zImage + ramdisk.img)
environment	eMMC	• dra7-evm-lcd-lg.dtb (J6 Rev-G) • dra7-evm-lcd-osd.dtb (J6 Rev-H) • dra72-evm-lcd-lg.dtb (J6 Eco Rev-B) • dra72-evm-revc-lcd-osd101t2045.dtb (J6 Eco Rev-C) • dra71-evm-lcd-auo-g101evn01.0.dtb (J6 Entry) • dra76-evm.dtb (J6 Plus EVM)
system	eMMC	system.img
vendor	eMMC	vendor.img
recovery	eMMC	recovery.img
ipu1	eMMC	ipu1 image for late attach
ipu2	eMMC	ipu2 image for late attach
dsp1	eMMC	dsp1 image for late attach
dsp2	eMMC	dsp2 image for late attach
data	eMMC	userdata.img

Syntax for fastboot flash command is

```
fastboot flash <partition name> <image file>
```

To update kernel (zImage) alone, flash the zimage partition

```
fastboot flash zimage <path to zImage>
```

To update ramdisk.img alone, you will have to create boot.img file and flash it. Steps to create boot.img file below

```
cd $YOUR_PATH/emmc_files
mkbootimg  --kernel <path/to/zImage> --ramdisk <path/to/ramdisk.img> --ramdisk_offset 0x01f00000 --base 0x80000000 --output boot.img
```

Note: To update MLO and u-boot.img in SPI, follow commands below. They have to be flashed together.

```
fastboot oem spi
fastboot flash xloader <MLO File>
fastboot flash bootloader <u-boot.img file>
```

Re-Building SGX kernel module

Some changes made to kernel defconfig can causes prebuilt SGX kernel modules to not load

Follow the instructions below for rebuilding SGX modules (pvrsrvkm.ko)

```
cd ${MYDROID}/device/ti/proprietary-open/jacinto6/sgx_src/eurasia_km/eurasiacon/build/linux2/omap_android
export KERNELDIR=${YOUR_PATH}/kernel/android-4.4
export ANDROID_ROOT=${MYDROID}
export CROSS_COMPILE=${MYDROID}/prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-4.9/bin/arm-linux-androideabi-
export KERNEL_CROSS_COMPILE=${MYDROID}/prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-4.9/bin/arm-linux-androideabi-
export ARCH=arm
make TARGET_PRODUCT="jacinto6evm" BUILD=release
```

Push the newly built *pvrsrvkm.ko* file to */system/lib/modules/* location on the target and reboot the board.

Building for custom platform

There are some additional changes needed when building for a custom platform (If your product name is other than jacinto6evm). In the example below we are using an example product name called *newproduct*. Replace this with your product name.

- Change in Makefile to add your product name (device/ti/proprietary-open project)

```
--- a/eurasia_km/eurasiacon/build/linux2/omap_android/Makefile
+++ b/eurasia_km/eurasiacon/build/linux2/omap_android/Makefile
@@ -77,7 +77,7 @@ VS_PRODUCT_VERSION                    := 5
 # FIXME: Re-enable this ASAP
 SUPPORT_ACTIVE_POWER_MANAGEMENT := 0
 endif
-ifneq ($(filter jacinto6evm,$(TARGET_DEVICE)),)
+ifneq ($(filter jacinto6evm newproduct,$(TARGET_DEVICE)),)
 SGXCORE                                       := 544
 SGX_CORE_REV                          := 116
 SGX_FEATURE_MP                        := 1
```

- In the build command above replace TARGET_PRODUCT="newproduct"

J6 Entry Display Width

The J6 Entry uses a lower resolution display than the J6 Eco or J6, resulting in a band of unused display with default settings. To fix this, enter:

```
su
wm density 160
```

Then reboot the board. This fixes the pixel density settings to work better with the J6 Entry panel.

Kernel Modules

As of Android O, loadable kernel modules are required.

Steps to add a kernel module to vendor partition

The framework to include modules to vendor partition and to load them at boot time is now enabled for jacinto6evm. Refer to updated build instructions as well.

- Step1: In file device/ti/jacinto6evm/BoardConfig.mk, update variable BOARD_VENDOR_KERNEL_MODULES with module to be included in vendor partition (/vendor/lib/modules)
- Step2: In file device/ti/jacinto6evm/init.jacinto6evmboard.rc, update the modprobe list with module to be loaded by Android init
- Step3: Rebuild Android and flash updated vendor.img and boot.img

For an example, see here (http://review.omapzoom.org/#/c/38531/)

See BOARD_VENDOR_KERNEL_MODULES in device/ti/jacinto6evm/BoardConfig.mk for a list of current modules.

IPC

- IPC version used in this release is **3.47.00.00**
- IPC package is already included when you download Android source following instructions from release notes. It is in path `${MYDROID}/hardware/ti/ipc`. You can also clone the IPC git tree individually from `git://git.ti.com/ipc/ipcdev.git`
- Late Attach and Error Recovery
  - Early_Boot_and_Late_Attach#Enabling_Early_Boot_and_Late_attach_on_Android_.286AM.1.2.29
  - IPC_Slave_Error_Recovery
- UIO and misc driver (for avoiding /dev/mem)
  - IPC_GateMP_Support_for_UIO_and_Misc_Driver
- Benchmarking
  - IPC_BenchMarking
- IPC Users Guide
  - IPC_Install_Guide_Android
  - IPC_Users_Guide

Multimedia

List of Media formats supported by Android https://developer.android.com/guide/topics/media/media-formats.html

Audio

Note: Android media output and input is on the EVM CPU board, while the JAMR3 output is dedicated for radio.

Below is some information about what input/output connectors in VayuEVM can be used for audio and what type of accessories can be connected.

INPUT

- **Microphone:** 3.5mm jack on the CPU board, labeled as “MIC IN” (P10). It’s assigned to Android’s built-in mic
  - Dedicated microphone with 3.5mm stereo plug has to be used
  - 3-pin plug: Tip -> Left channel, Ring -> Right channel, Sleeve -> Ground
  - Microphone from phone headsets cannot be used
- Line-In, Aux-In: 3.5mm jack on JAMR3 board, labeled as “AUX IN (P6)”. It’s assigned to Android’s auxiliary input
  - 3-pin plug: Tip -> Left channel, Ring -> Right channel, Sleeve -> Ground

OUTPUT

- **Speaker:** 3.5mm jack on the CPU board, labeled as “HEADPHONE” (P13). It’s assigned to Android’s speaker output
  - 3-pin plug: Tip -> Left channel, Ring -> Right channel, Sleeve -> Ground
  - The speaker is the default, always connected output in Android, but since there is no on-board installed speakers in the EVM, this output acts like it
  - Typical headphones, headsets can be connected to this jack
- **Line-Out:** 3.5mm jack on the CPU board, labeled as “LINE OUT” (P12). It’s also assigned to Android’s speaker output
  - 3-pin plug: Tip -> Left channel, Ring -> Right channel, Sleeve -> Ground
  - Recommended for outputs with high impedance

USB

ADB

From this release (Kernel 4.4), ADB is enabled via configFS gadget

USB Dual Role / DRD

This release supports USB dual role / Dynamic role switch for USB.

- **Software Configuration:**
  - Make sure following configs are enabled in Kernel defconfig

```
CONFIG_USB_OTG=y
CONFIG_EXTCON_USB_GPIO=y
```

  - The USB port you want to use for USB dual role should be marked as "otg" in dts file, example for usb1 below

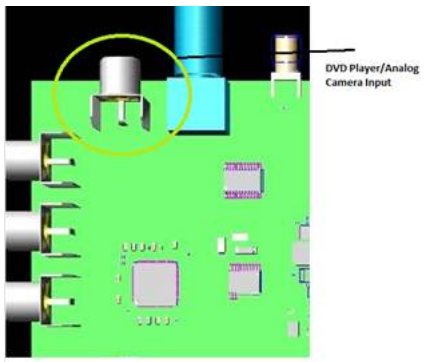
```
&usb1 {
        dr_mode = "otg";
};
```

- **Testing Procedure:**
  1. The dynamic role switching to "host" or "device" mode is based on usb-id pin when the usb cable (type-A for host, type-B for device) is connected to usb1 or usb2 ports
  2. The role switching to "device" or "host" can also be done by writing to dwc3 debugfs nodes

```
#Ex for USB1 port below, use 488d0000.dwc3 directory for USB2 port
echo "device" > /sys/kernel/debug/48890000.dwc3/mode
or
echo "host" > /sys/kernel/debug/48890000.dwc3/mode
```

Camera

Camera Summary

Feature	Camera Module	Board	DIP Switch	Setup
Analog camera ▪ Preview upto D1 NTSC ▪ De-Interlace using VPE	▪ Camera NTSC interlaced capture	CPU Board + JAMR3	SW2 (JAMR Board) OFF OFF	

Note: We now use standard Camera app to test Camera on Android SDK. On first launch of Camera app it will require you to go grant permissions in automatic prompts.

DCAN Vehicle HAL

Our DCAN testing is carried out using canutils. This release included testing with a simple Vehicle HAL DCAN implementation.

In order to implement Vehicle HAL using the DCAN interface, the simplest method is to take Google's default camera HAL (hardware/interfaces/automotive/vehicle/2.0/default in this release, further locations are referenced from here) and modify it for your needs. Google's implementation provides a thick proxy layer between the HIDL and vendor implementations, also providing the VehicleHal class for the vendor to implement (common/include/vhal_v2_0). There is also a dummy implementation (impl/vhal_v2_0) where an example implementation of VehicleHal class sits. If you use this dummy implementation as a basis for your development, you simply need to write a new class for CAN that implements their CommBase class, similar to their SocketComm class.

For our test implementation, we used calls to libsocketcan-0.0.8, which you can download here (http://public.pengutronix.de/software/libsocketcan/). For the purpose of this build, you really only need the libsocketcan.c from the source, which you can build and link as a static library in your Android.mk. SocketCAN is an open source API for CAN that works with the c_can_platform driver we are using and works the same as most other socket APIs.

For a working example of CAN code, look at canutils-4.0.6 (https://git.pengutronix.de/cgit/tools/canutils/tree/src?id=canutils-4.0.6).

For more information on using DCAN on our platform, see the Linux_Core_DCAN_User's_Guide.

For more information about building canutils and cross compiling libsocketcan, see AM335X_DCAN_Driver_Guide#CAN_Utilities

Radio

- Radio platform bringup - Jacinto_Radio_Integration
- Radio package is available only through CDDS, please contact Lester Longley (mailto:lester@ti.com) for more info.

HDMI

HDMI only setup

By default dual display is supported in this release, but to bring up a HDMI only setup,

- Non lcd dtb for the EVM should be used (Ex: For J6 EVM, use dra7-evm.dtb)
- you need to set two additional properties
  - ro.hwc.primary.conn
  - ro.hwc.external.conn

Use below commands to set the property and reboot the EVM.

```
# If running via ADB
adb root
adb remount
adb shell 'echo "ro.hwc.primary.conn=HDMI" >> /system/build.prop'
adb shell 'echo "ro.hwc.external.conn=dummy" >> /system/build.prop'
```

or

```
# If running via COnsole directly
su
mount -o rw,remount /system
echo "ro.hwc.primary.conn=HDMI" >> /system/build.prop
echo "ro.hwc.external.conn=dummy" >> /system/build.prop
sync
```

Recovery Mode

Switching to recovery mode is enabled in this release. This feature is enabled using u-boot environment variables. From Android userspace, an environment variable is set which is used by u-boot to decide which image to load (boot.img or recovery.img)

- Build the u-boot env tool (fw_printenv) using linaro tool chain

```
cd ${YOUR_PATH}/u-boot
export CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux/bin/arm-linux-gnueabihf-
make HOSTLDFLAGS=-static env
```

- Create the fw_env.config file with below content (The offset here corresponds to CONFIG_ENV_OFFSET definition in include/configs/dra7xx_evm.h)

```
/dev/block/mmcblk0 260000 0x20000
/dev/block/mmcblk0 280000 0x20000
```

- Copy all files to targer

```
adb root
adb remount
adb push tools/env/fw_printenv /system/bin/
adb push tools/env/fw_printenv /system/bin/fw_setenv
adb push fw_env.config /system/etc/
```

- With steps above, all the necessary tools are in file system. To get in to recovery mode, set the "reboot_image" u-boot environment variable to value "recovery" before doing reboot.

```
fw_setenv -l /data reboot_image "recovery"
```

Boot Options

All eMMC option

If you don't want to use QSPI NOR and want everything to be flashed to emmc and boot from emmc, follow instructions below.

- Remove following lines from fastboot.sh script before flashing

```
echo "Setting target for bootloader to SPI"
${FASTBOOT} oem spi
```

- Required DIP switch settings after flashing: This configuration corresponds to the following device boot order: SD ⇒ eMMC

```
SYSBOOT [0-15]
OFF ON OFF OFF OFF ON OFF OFF    ON OFF OFF OFF OFF OFF OFF ON
```

Single Stage bootloader

To enable the early boot flow, configure the DRA7xx board to boot in QSPI_1 production boot mode by setting the sys_boot dip switches as shown below and reboot the board

```
SYSBOOT [0-15]
OFF ON ON OFF ON ON OFF OFF    ON OFF OFF OFF OFF OFF OFF ON
```

You can stop the single stage boot flow and get to u-boot prompt, if you keep pressing character 'c' on the UART console when powering up the board.

USB Peripheral Boot

This mode is for flashing a fresh new board, which has no MLO or u-boot present on SPI or eMMC (To avoid SD card)

Build required components

- Build usbboot Linux Host binary

```
cd ${YOUR_PATH}
git clone git://git.omapzoom.org/repo/omapboot.git
cd omapboot
git checkout 609ac271d9f89b51c133fd829dc77e8af4e7b67e
cd host/tools
make
```

- Build special SPL for peripheral boot

```
cd ${YOUR_PATH}/u-boot
export CROSS_COMPILE=${MYDROID}/prebuilts/gcc/linux-x86/arm/arm-eabi-4.8/bin/arm-eabi-
export ARCH=arm
make dra7xx_evm_config
make menuconfig # Enable "Boot Images" -> "Enable SPL with DFU"
make
```

Steps to use peripheral boot

- Disconnect the power supply from the board
- Change DIP switch as below, this configuration corresponds to the following device boot order: USB

```
SYSBOOT [0-15]
OFF OFF OFF OFF ON OFF OFF OFF    ON OFF OFF OFF OFF OFF OFF ON
```

- Connect micro USB cable from Linux PC to USB3.0 port on EVM
- Execute the following command on Linux PC

```
cd $YOUR_PATH/emmc_files
sudo ./usbboot-stand-alone -s ./spl/u-boot-spl.bin
```

- Power on or reset the board
- At this point DRA7xx communicates with host machine, u-boot-spl will be transferred into DRA7xx internal memory and ready for DFU usage

```
reading ASIC ID
CHIP: 5641
rom minor version: 01
IDEN: 00000000000000000000000000000000000000000000
MPKH: 0000000000000000000000000000000000000000000000000000000000000000
CRC0: af96e165
CRC1: 00000000
device is GP
sending 2ndstage to target...
```

- Execute below commands from Linux PC to transfer u-boot via dfu-util. After executing second command, u-boot will start running immediately, interrupt and stop at u-boot console.

```
sudo dfu-util -l
sudo dfu-util -c 1 -i 0 -a 0 -D u-boot.img -R
```

- Now the u-boot is transferred via dfu, now u-boot starts executing. Stop at u-boot prompt and execute below commands and put the board in fastboot mode

```
=> env default -f -a
=> setenv partitions $partitions_android
=> env save
=> fastboot 0
```

- Now you can flash the regular boot loaders to QSPI via fastboot and have valid bootloaders on the board. Use the non DFU enabled bootlaoder images

```
# For flashing to QSPI, issue commands below from Linux PC
cd $YOUR_PATH/emmc_files
sudo ./fastboot oem spi
sudo ./fastboot flash xloader MLO
sudo ./fastboot flash bootloader u-boot.img
```

- Now change the DIP switch back to SD=>QSPI configuration

Android Verified Boot

Android_Verified_Boot

AOSP Delta

To find out the delta/patches that TI added on top of Android AOSP, follow the instructions below

- This release is based on Oreo version of AOSP (8.0.0)
  - AOSP Branch: oreo-r6-release
  - Corresponding TI Branch: d-oreo-r6-release
- Once you have the 6AO.1.0 repo downloaded on your PC, use the command below to list out all patches added by TI on top of AOSP projects

```
cd $MYDROID
repo forall -p -c 'git log --oneline omap-mirror/oreo-r6-release..omap-mirror/d-oreo-r6-release'
```

Resource Allocation

6AO_Resource_Allocation

Errata Disposition

DRA7xx_Errata_Status#DRA7xx_Status_.282016LTS.29

Post release patches

Domain	Issue/Subject	Fix	Component	Release
Display	Fix for fence timeout issue	http://review.omapzoom.org/#/c/38769/	AFS	-

NOTE: The list above does not cover Kernel/u-boot fixes/enhancements post 6AO.1.0 release. Please follow TI Kernel and u-boot branches for latest changes.

{{

1. switchcategory:MultiCore=
   - For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
   - For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **6AO.1.0 Application Notes** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **6AO.1.0 Application Notes** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article **6AO.1.0 Application Notes** here.*

*For technical s please post yo questions at http://e2e.ti.co Please post o comments abo article **6AO.1.(Application N here.*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity