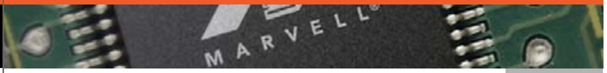


ARMADA® 610 Tablet Reference Platform, Revision 5

for Android™ 3.2, Linux® Kernel 2.6.35

Software Release Notes

Doc. No. MV-S302147-00, Rev. -

November 17, 2011

CONFIDENTIAL

Document Classification: Proprietary Information

Marvell. Moving Forward Faster

Document Conventions

↘	**Note:** Provides related information or information of special importance.
!	**Caution:** Indicates potential damage to hardware or software, or loss of data.
ϟ	**Warning:** Indicates a risk of personal injury.

Document Status

Doc Status: Released	

For more information, visit our website at: www.marvell.com

Disclaimer

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document.

Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.

With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:

1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;

2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,

3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").

At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Patent(s) Pending—Products identified in this document may be covered by one or more Marvell patents and/or patent applications.

Table of Contents

1 Overview

This software release package contains source code for the Marvell® ARMADA® 610 Tablet Reference Platform, Revision 5 for Android™ 3.2, Linux® kernel 2.6.35.

The release package includes:

- Prebuilt binaries – Use to flash the ARMADA 610 Tablet Reference Platform, Revision 5; these binaries are ready for immediate use.
- Source code – Customize and build the code to create new binaries.

THIS SOFTWARE IS PREPRODUCTION SOFTWARE. IT CANNOT BE USED "AS IS" FOR PRODUCTION SYSTEMS. NO WARRANTY, EXPRESSED OR IMPLIED, IS TO BE ASSOCIATED WITH THIS SOFTWARE AND THE USER ASSUMES ALL RISKS.

For the latest Marvell documentation and the latest Marvell software and hardware updates, contact your Marvell representative.

1.1 System Requirements

1.1.1 Hardware Requirements

This release requires the Marvell ARMADA 610 Tablet Reference Platform, Revision 5 with A0/A1/A2 stepping of the ARMADA 610 processor and 1 GB of DDR3 memory.

You can visually identify the reference platform hardware by the printed board (PB) number on the secondary side (bottom side) of the main system board. The Marvell ARMADA 610 Tablet Reference Platform, Revision 5, has a PB number of F00073-500. The PB number indicates the following information:

- The F00073 number identifies the PB as a Marvell ARMADA 610 Tablet Reference Platform.
- The -500 number indicates revision 5 of the unpopulated board and schematics.

For more information about the ARMADA 610 Tablet Reference Platform, Revision 5, refer to the *Marvell® ARMADA® 610 Tablet Reference Platform, Revision 5 User’s Guide*, MV-L100855-10.

1.1.2 Software Requirements

This release version requires:

- Host PC with operating system – Ubuntu® 10.04

Note

Marvell routinely tests the Android build on Ubuntu 10.04. Marvell recommends using Ubuntu 10.04 and manually upgrading the Git to v1.7.1 or later. You can download the Git package from http://git-scm.com/download.

Note

Follow the instructions on http://source.android.com/source/download.html to set up the Android build environment on Ubuntu Linux.

- Android 3.2, Linux kernel 2.6.35

1.2 Platform Features

Table 1 lists the platform features for this version of the Marvell ARMADA 610 Tablet Reference Platform, Revision 5 for Android 3.2 release package.

Table 1: Platform Features (Sheet 1 of 2)

Features		Support
General	Android Version	3.2
	Linux kernel	2.6.35
Power Management	Android power integration	Yes
	Battery information	Yes
	Suspend/Resume	Yes
Video Playback	Video output to HDMI™	Yes
	Video output optimized by overlay	Yes
	Simultaneous video output to HDMI and UI operation on LCD	Yes
	Video rotation	Yes
Audio Playback	Headset switch detection	Yes
	Audio driver integration	Yes
	Audio to headset	Yes
	Audio to speaker	Yes
	Audio to HDMI	Yes
Audio Recording	AMR-NB encoding	Yes
	AAC encoding	Yes
	Sound recorder integration	Yes
Video Recording	Camera stack integration	Yes
	Camera sensor tuning	Yes
Touch	Single touch	Yes
	Multiple touch	Yes
Bluetooth® technology	Bluetooth base stack	Yes
	Advanced Audio Distribution Profile (A2DP)	Yes
	Object Push Profile (OPP)	Yes
	Human Interface Device (HID) Profile	Yes
	Personal Area Networking (PAN) Profile	Yes
	Audio/Video Remote Control Profile (AVRCP)	Yes
Sensors	Gravity sensor	Yes
	Light sensor	Yes
Light	LCD backlight	Yes
Alarm	Trigger alarm from Standby	Yes

Table 1: Platform Features (Sheet 2 of 2)

Features		Support
Wireless	Wi-Fi® access without password	Yes
	Wi-Fi Protected Access (WPA)/WPA2	Yes
	Wi-Fi WPA Enterprise • Protected Extensible Authentication Protocol (PEAP) • Tunneled Transport Layer Security (TTLS) • Transport Layer Security (TLS)	Yes
	Wi-Fi Wired Equivalent Privacy (WEP)	Yes
	Wi-Fi Mobile Hotspot	Yes
	Wi-Fi connection stress test	Yes
	3G USB Dongle	No
Multimedia	GStreamer	See Table 2, Supported Media Format and Codecs, on page 8
	Stagefright	
Video Output	Output through overlay	Yes
Tools	Android Debug Bridge (ADB) integration	Yes
	Marvell Code Performance Analyzer integration	Yes
Graphics	2D/3D graphics controller (GC860[1])	Yes
User Storage	SD card	Yes
	Internal storage partition	Yes
Boot Storage	eMMC	Yes
	SD	Yes
System Update	Fast boot protocol	Yes
	SD upgrade	Yes
	Factory reset	Yes
Security	Wireless Trusted Platform Service Package (WTPSP)	Yes
	Wireless Trusted Module (WTM) Adapter for Linux Kernel Crypto Framework	Yes
	Optimized OpenSSL	Yes

[1] GC860 refers to the Vivante Corporation GCCORE Graphics Processing Unit IP architecture.

1.3 Multimedia Features

This release supports the media formats and the codecs listed in Table 2.

Table 2: Supported Media Format and Codecs (Sheet 1 of 2)

Containers	Extensions	Audio/Video Combinations		Playback Engine	Status
		Audio	Video		
ASF	.asf	WMA	MPEG-4	GStreamer	Ready
	.wmv	WMA	WMV		
	.wma	WMA	--		
AVI	.avi	MP3	H.264	GStreamer	Ready
		MP3	MPEG-4		
		MP3	H.263		
MOV	.mov	AAC	H.264	GStreamer	Ready
		AAC	MPEG-4		
MP4	.mp4	AAC	H.264	Stagefright	Ready
		AAC	H.263		
		AAC	MPEG-4		
		AMR-NB/WB	H.264		
		AMR-NB/WB	H.263		
		AMR-NB/WB	MPEG-4		
	.m4a	AAC	--	Stagefright	Ready
		AMR-NB/WB	--		
	.m4v	--	H.264	Stagefright	Ready
		--	H.263		
		--	MPEG-4		
MPEG-2 PS	.mpg	MP3	MPEG-2	GStreamer	Ready

Table 2: Supported Media Format and Codecs (Sheet 2 of 2)

Containers	Extensions	Audio/Video Combinations		Playback Engine	Status
		Audio	Video		
3GPP	.3gp/ .3gpp .3g2 .3gpp2	AAC	H.264	Stagefright	Ready
		AAC	H.263		
		AAC	MPEG-4		
		AMR-NB/WB	H.264		
		AMR-NB/WB	H.263		
		AMR-NB/WB	MPEG-4		
	.3ga	AAC	--	Stagefright	Ready
		AMR-NB/WB	--		
MKV	.mkv	MP3	H.264	Stagefright	Ready
		AAC	H.264		
AAC	.aac	AAC	--	GStreamer	Ready
	.adts	AAC	--	GStreamer	Ready
MP3	.mp3	MP3	--	Stagefright	Ready
WebM	.webm	Ogg	VP8	Stagefright	Ready
TS	.ts	AAC	H.264	Stagefright	Ready
AMR	.amr	AMR-NB/WB	--	Stagefright	Ready

1.4 Board Support Package Features

Board support package features for the ARMADA 610 Tablet Reference Platform, Revision 5 for Android 3.2 are as follows.

- U-Boot
  - eMMC, non-trusted boot
  - USB Ethernet download
  - zImage format support
  - Support for burning Yet Another Flash File System (YAFFS) image
  - Support for burning an image into eMMC
  - Support for low battery protection
  - Support for power off charger
- Linux Kernel 2.6.35
  - L1 cache
  - L2 cache
  - Interrupt controller
  - Peripheral DMA (PDMA) controller
  - Memory controller
  - Real-Time Clock (RTC)
  - Operating System Timer (OST)
  - Intel® Wireless MMX™[1] technology
  - General purpose Input Output (GPIO) interrupt request (IRQ)
  - Clock management
  - Single level cell (SLC) NAND flash memory
  - OneNAND flash memory
  - Journaling Flash File System, version 2 (JFFS2) support
  - Unsorted Block Image File System (UBIFS) support
  - Yet Another Flash File System (YAFFS)
  - MultiMediaCard (MMC3.2 and MMC4.0)
  - Secure Digital (SD)/SDIO (SD1.1 and SD2.0)
  - UART
  - HDMI Audio
  - SSPA drivers
  - ALSA framework
  - Keypad
  - I2C - Normal I2C (see, I2C Stability, on page 11)
  - DSI LCD panel (base frame, overlay)
  - HDMI LCD TV path
  - Audio DMA (ADMA) controller
  - Marvell Wireless Memory Management technology
  - USB client
  - Maxim MAX8925 Power Management Integrated Circuit (PMIC) and the MAX8952 regulator
  - Battery driver

[1] Intel and MMX and related marks are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

- USB charger
- USB HOST and USB On-the-Go (OTG)
- Video DMA (VDMA) controller
- WM8994 codec
- Dynamic Voltage and Frequency Management (DVFM)
- Multi touch
- On key and Reset
- Wi-Fi, Bluetooth on SD8787
- HDMI EDID
- Marvell Vmeta™ technology, a multiple format high-definition video codec supporting multi instances
- ARMv7 mode
- Audio record
- Power off command
- OmniVision® OV5642 5-megapixel SoC sensor support
- LCD additional mode
- Capacity keypad
- TPK800 touch
- AUO panel
- FM audio
- CyWee motion sensor
- High-bandwidth Digital Content Protection (HDCP)
- Extended File System, version 4 (ext4) support
- ZSP® audio mode
- N-trig® touch support
- ISL9519 charger
- MAX17042 fuel gauge
- CM3213 ambient light sensor (ALS) with interruption

I2C Stability

The power on sequence can impact the I2C stability.

On the Marvell ARMADA 610 Tablet Reference Platform, Revision 5, use the following steps to power on.

1. Unplug the USB cable and the power cable to make sure power is off to the board.
2. Insert the power cable.
3. Press the on-key for about 2 seconds to turn on power to the board.
4. Insert the USB cable.

Following these steps ensures that the board is powered by the DC power, not by the USB cable VBUS.

1.5 Release Package Contents

The following tables list and describe the release package for the ARMADA 610 Tablet Reference Platform, Revision 5 for Android 3.2, Linux kernel 2.6.35.

Table 3: Prebuilt Binary Files

Files	Description
`ARMADA610_ANDROID_HONEYCOMB_PREBUILT_BIN.zip`	Prebuilt bin binaries
• `primary_gpt_8g`	Primary GUID partition table (for 8GB eMMC)
• `secondary_gpt_8g`	Secondary GUID partition table (for 8GB eMMC)
• `system.img`	Android system image
• `userdata.img`	Android userdata image
• `zImage.android`	Kernel image
• `zImage_recovery.android`	Recovery kernel image
• `ramdisk.img`	RAM disk image
• `ramdisk_recovery.img`	Recovery RAM disk image
• `vmlinux`	Kernel image in ELF
• `System.map`	Kernel symbol map
• `Symbols_lib.tgz`	System libraries with symbol
• `nontrusted/u-boot.bin`	Nontrusted U-Boot
• `nontrusted/ntim_platform_512m_ddr3.bin`	NTIM header
• `nontrusted/ntim_platform_512m_ddr3.txt`	NTIM description file
• `nontrusted/MMP2_LINUX_ARM_BL_3_2_21_EB_JO.bin`	NTIM loader
• `nontrusted/dntim_platform.bin`	Dynamic NTIM (DNTIM) header
• `nontrusted/dntim_platform.bin`	DNTIM description file
• `nontrusted/update_droid.zip`	Nontrusted update package
• `nontrusted/update_recovery.zip`	Nontrusted update recovery package
• `trusted/u-boot.bin`	Trusted U-Boot
• `trusted/tim_platform_512m_ddr3.bin`	TIM header
• `trusted/tim_platform_512m_ddr3.txt`	TIM description file
• `trusted/MMP2_LINUX_ARM_BL_3_2_21_TRUSTED_EB_JO.bin`	TIM loader
• `trusted/dtim_platform.bin`	Dynamic TIM (DTIM) header
• `trusted/dtim_platform.txt`	DTIM description file
• `trusted/EncryptKey.txt`	Encryption key
• `trusted/update_droid.zip`	Trusted update package
• `trusted/update_recovery.zip`	Trusted update recovery package

Note The WTM firmware image (currently version 2.1.8.3) must be downloaded separately from the Marvell Extranet at *My Products/Cellular & Handheld Solutions/Applications Processors/ARMADA 610 (MMP2) Software/WTM/Version 2.1.8*. Contact your Marvell representative if you have issues about the download.

Table 4: Source Files

Files	Description
`ARMADA610_ANDROID_HONEYCOMB_SRC`	Source code tarball (patch based source code)
• `setup_android.sh`	Script help to set up the Android code base from the `xxx_src.tgz` and `xxx_patches.tgz`
• `android_patches.tgz`	Marvell patches to the Android Projects
• `android_src.tgz`	Source code for projects added by Marvell
• `marvell_manifest.xml`	Manifest xml file to download the Android source code from Google as a base
• `kernel_patches.tgz`	Marvell patches to `kernel_src.tgz`
• `kernel_src.tgz`	Kernel base source code
• `uboot_src.tgz`	U-Boot base source code
• `uboot_patches.tgz`	Marvell patches to `uboot_src.tgz`
• `obm_src.tgz`	OEM Boot Module (OBM) source code

! Caution The non-trusted image module (NTIM), the trusted image module (TIM), and BootLoader (OEM boot module) files provided are designed and customized for use with the associated Marvell hardware platform. Use these files as a reference. You MUST create the NTIM, TIM, and BootLoader with the correct parameters for your design.

Failure to correctly implement the NTIM, TIM, or BootLoader may result in a boot failure or cause an unreliable operation of your device.

For information and assistance in correctly setting up your NTIM, TIM, and BootLoader, see the Marvell Boot ROM or Marvell Wireless Trusted Tool Package documentation or contact your Marvell Applications Engineer or Field Applications Engineer.

Note For detailed information about the WTPTP release package, see the *Marvell® Wireless Trusted Platform Tool Package for Application Processors Software Release Notes* (MV-S301673-00).

1.6 Marvell Optimization of Adobe Flash Player 10.3

To download the Marvell Optimization of Adobe® Flash® Player 10.3 plug-in for Android 3.2 on the Marvell ARMADA 610 Tablet Reference Platform, go to the Marvell Extranet website at http://www.marvell.com/extranet. Then go to the following folder location:

> My Products/ Cellular & Handheld Solutions/ Applications Processors/ ARMADA 610 (MMP2)/ Software/ Flash_Player/ ARMADA 610 Tablet Reference Platform (Brownstone)/ FP10.3

Note

If you do not have a Marvell Extranet user ID, click on the “register” link at http://www.marvell.com and follow the instructions therein.

If you cannot access the FP10.3 folder, contact your Marvell representative.

2 Installation

This section provides procedures for

- Identifying an ARMv6 or ARMv7 mode boot
- Programming the binaries onto flash memory with the Marvell eXtreme Debugger
- Downloading Android onto flash memory
- Setting up the Android working directory
- Building Android
- Using the optimized OpenSSL for Marvell platforms
- Protecting the HDCP key

Note

See Section 1.5, Release Package Contents, on page 12 for a description of the release package contents.

2.1 Use the Prebuilt Binaries

The following procedures provide information for programming binaries onto flash memory using the Marvell eXtreme Debugger.

2.1.1 Identifying an ARMv6 or ARMv7 Mode Boot

The non-trusted image module (NTIM) or the trusted image module (TIM) is updated to switch the processor core from ARMv6 to ARMv7 mode. If the new NTIM or TIM image is burned into flash correctly, the boot ROM switches the core from ARMv6 to ARMv7 mode during the boot. Without a new NTIM, the processor still boots in the ARMv6 mode.

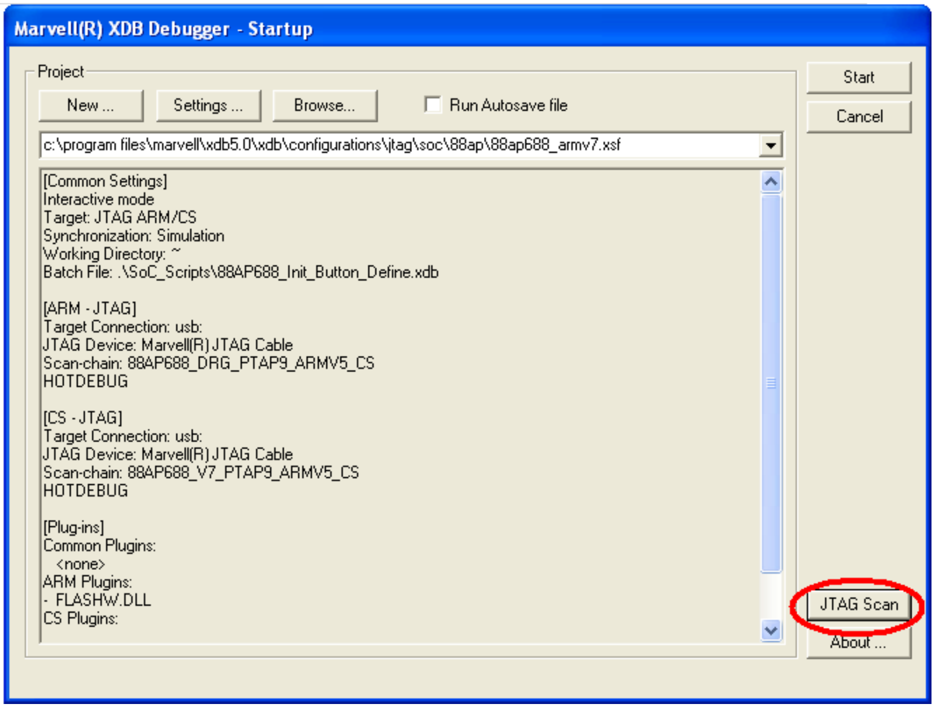
To identify whether the processor is in ARMv6 or ARMv7 mode, use the steps that follow.

Note

You need the Marvell eXtreme Debugger, version 5.2 Beta 1 or later. Download the Marvell eXtreme Debugger from the Marvell Extranet or contact your Marvell representative.

1. With the Marvell eXtreme Debugger (XDB) running, click the **JTAG Scan** button on the Startup screen.

Figure 1: XDB Debugger JTAG Scan Button



c:\program files\marvell\xdb5.0\xdb\configurations\jtag\soc\88ap\88ap688_armv7.xsf

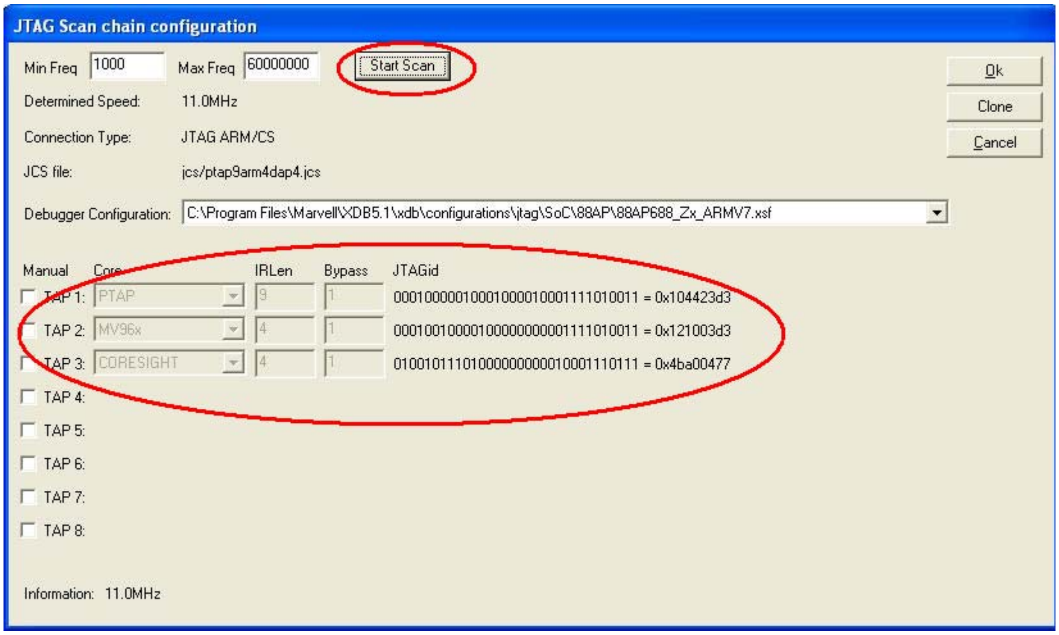
```
[Common Settings]
Interactive mode
Target: JTAG ARM/CS
Synchronization: Simulation
Working Directory: ~
Batch File: .\SoC_Scripts\88AP688_Init_Button_Define.xdb

[ARM - JTAG]
Target Connection: usb:
JTAG Device: Marvell(R) JTAG Cable
Scan-chain: 88AP688_DRG_PTAP9_ARMV5_CS
HOTDEBUG

[CS - JTAG]
Target Connection: usb:
JTAG Device: Marvell(R) JTAG Cable
Scan-chain: 88AP688_V7_PTAP9_ARMV5_CS
HOTDEBUG

[Plug-ins]
Common Plugins:
   <none>
ARM Plugins:
- FLASHW.DLL
CS Plugins:
```

2. Click the **Start Scan** button.

   Figure 2 shows a scan chain that indicates the core mode is in ARMv7 mode.

   - Three Test Access Ports (TAPs) indicate the core is in ARMv7 mode.
   - Four TAPs indicate the core is in ARMv6 mode

Figure 2: Start Scan Button and JTAG Scan Chain



Min Freq 1000 Max Freq 60000000

Determined Speed: 11.0MHz

Connection Type: JTAG ARM/CS

JCS file: jcs/ptap9arm4dap4.jcs

Debugger Configuration: C:\Program Files\Marvell\XDB5.1\xdb\configurations\jtag\SoC\88AP\88AP688_Zx_ARMV7.xsf

Manual		Core	IRLen	Bypass	JTAGid
□	TAP 1:	PTAP	9	1	00010000010001000010001111010011 = 0x104423d3
□	TAP 2:	MV96x	4	1	00010010000100000000001111010011 = 0x121003d3
□	TAP 3:	CORESIGHT	4	1	01001011101000000000010001110111 = 0x4ba00477
□	TAP 4:				
□	TAP 5:				
□	TAP 6:				
□	TAP 7:				
□	TAP 8:				

Information: 11.0MHz

2.1.2 Burning the Trusted Binaries, Kernel, and Ramdisk to eMMC Using JTAG

Perform the steps that follow to burn the trusted WTM firmware, OEM boot module (OBM), U-Boot binaries, kernel, and ramdisk to eMMC flash memory using JTAG.

Note You need the Marvell eXtreme Debugger, version 5.2 Beta 1 or later. Download the Marvell eXtreme Debugger from the Marvell Extranet or contact your Marvell representative.

1. Use the Marvell eXtreme Debugger (XDB), version 5.2 Beta 1 and the appropriate configuration file for the steps that follow:
   - If the processor boots in the ARMv6 mode, use the `pxa688 _a0_armv6.xsf` file.
   - If the processor boots in the ARMv7 mode, use the `pxa688 _a0_armv7.xsf` file.

   Leave all settings at their default settings.

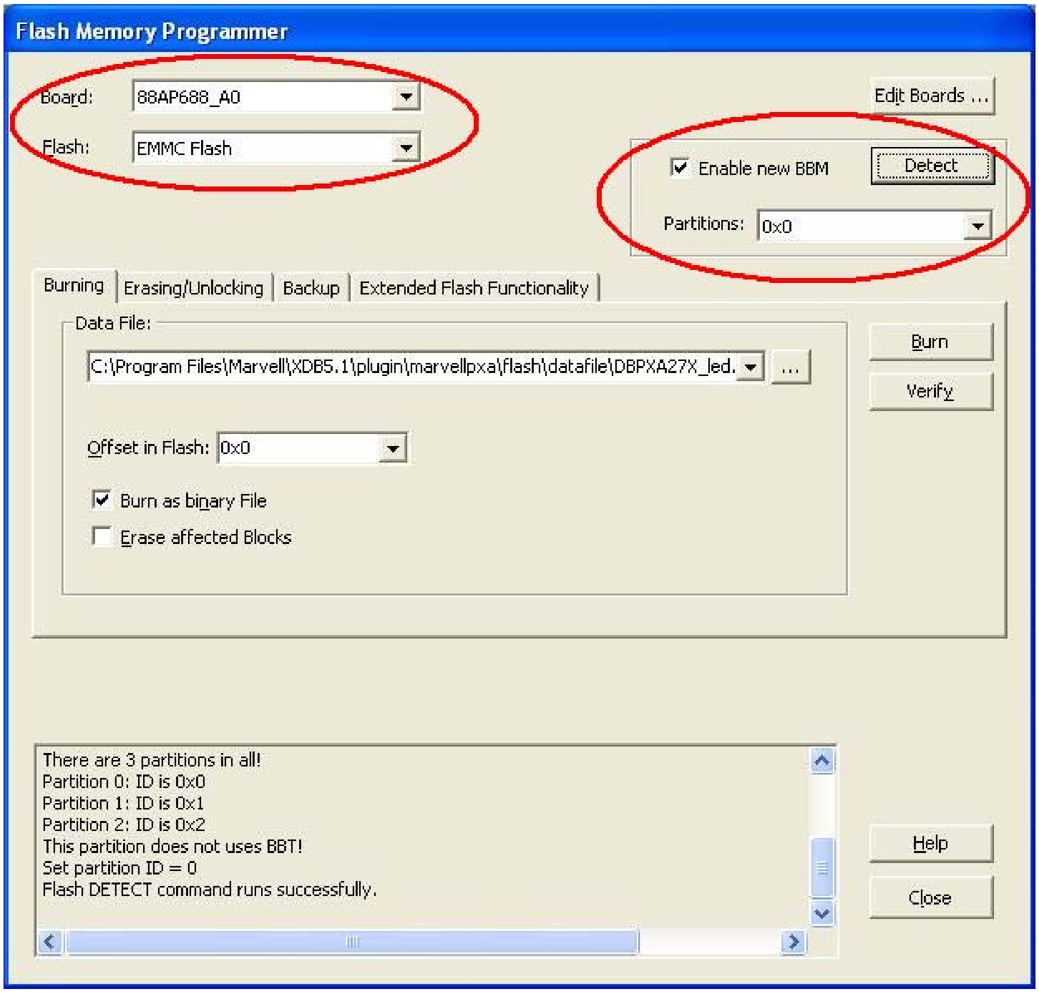
Note To identify in which mode your processor boots, see Section 2.1.1, Identifying an ARMv6 or ARMv7 Mode Boot, on page 15.

2. With XDB running, select **Flash** on the toolbar menu.
3. Click **Burn Flash** on the drop-down menu.
4. In the **Board** field, select **88AP688_A0**, and in the **Flash** field, select **EMMC Flash** (see Figure 3 on page 19).

Note If you are using the 88AP688 Z0/Z1 stepping processor, select **88AP688_ZX** in the **Board** field instead.

5. Check the **Enable new BBM** check box and click the **Detect** button to detect the eMMC partition (see Figure 3).

Figure 3: eMMC Flash Detect eMMC Partition



Board: 88AP688_A0

Flash: EMMC Flash

☑ Enable new BBM

Partitions: 0x0

Data File: C:\Program Files\Marvell\XDB5.1\plugin\marvellpxa\flash\datafile\DBPXA27X_led.

Offset in Flash: 0x0

☑ Burn as binary File

☐ Erase affected Blocks

```
There are 3 partitions in all!
Partition 0: ID is 0x0
Partition 1: ID is 0x1
Partition 2: ID is 0x2
This partition does not uses BBT!
Set partition ID = 0
Flash DETECT command runs successfully.
```

6. Select partition **0x1**. Burn the following binary files to the addresses as follows:

```
tim_platform_512m_ddr3.bin                    --> 0x0
Wtm_rel_mmp2.bin                              --> 0x4000
MMP2_LINUX_ARM_BL_3_2_21_TRUSTED_EB_JO.bin    --> 0x30000
u-boot.bin                                    --> 0x60000
u-boot.bin (as recovery u-boot)               --> 0xB0000
```

7. Select partition **0x0**. Burn the following binary file to the address as follows:

```
dtim_platform.bin            --> 0x780000
zImage.android               --> 0x980000
ramdisk.img                  --> 0x1180000
zImage_recovery.android      --> 0x1980000
ramdisk_recovery.img         --> 0x2180000
```

In step 6, the `Wtm_rel_mmp2.bin` image is a loadable WTM kernel firmware binary image that is executed by the ARMADA 610 on-chip secure processor. This image provides the cryptographic services for both Federal Information Processing Standard (FIPS) and non-FIPS mode operations.

To acquire the image, download the `WTM_Firmware_ARMADA610_2.1.8.3.zip` file from the Marvell Extranet at *My Products/Cellular & Handheld Solutions/Applications Processors/ ARMADA 610 (MMP2)/Software/WTM/Version 2.1.8*.

Once downloaded, extract both `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin` and `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin` images from the zip file. Both binary images support the same set of WTM primitive functions with the same API definition. However, the `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin` binary image performs the device RKEK/EC521-DK provision with primitives using the buffers within the secure SRAM to emulate the provisioning over the FUSE/OTP macro. On the other hand, the `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin` image performs the device key provision with primitives directly operating over the FUSE/OTP macro. With real OTP operation, the performed platform provision becomes permanent.

It is recommended to use the virtual OTP version of the WTM kernel binary image for platform software development. To do this, change the binary image file name `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin to Wtm_rel_mmp2.bin`.

It is recommended to use the real OTP version of the WTM kernel binary image for the device that is ready to be deployed as a product. To do this, change the binary image file name `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin to Wtm_rel_mmp2.bin`..

2.1.3 Burning the Non-trusted Binaries to eMMC Using JTAG

Perform the steps that follow to burn the non-trusted WTM firmware, OBM and U-Boot binaries to eMMC flash memory using JTAG:

Note

You need the Marvell eXtreme Debugger, version 5.2 Beta 1 or later. Download the Marvell eXtreme Debugger from the Marvell Extranet or contact your Marvell representative.

1. Use the Marvell eXtreme Debugger (XDB), version 5.2 Beta 1 or later and the appropriate configuration file for the steps that follow:
   - If the processor boots in the ARMv6 mode, use the `pxa688 _a0_armv6.xsf` file.
   - If the processor boots in the ARMv7 mode, use the `pxa688 _a0_armv7.xsf` file.

   Leave all settings at their default settings.

Note

To identify in which mode that your processor boots, see Section 2.1.1, Identifying an ARMv6 or ARMv7 Mode Boot, on page 15.

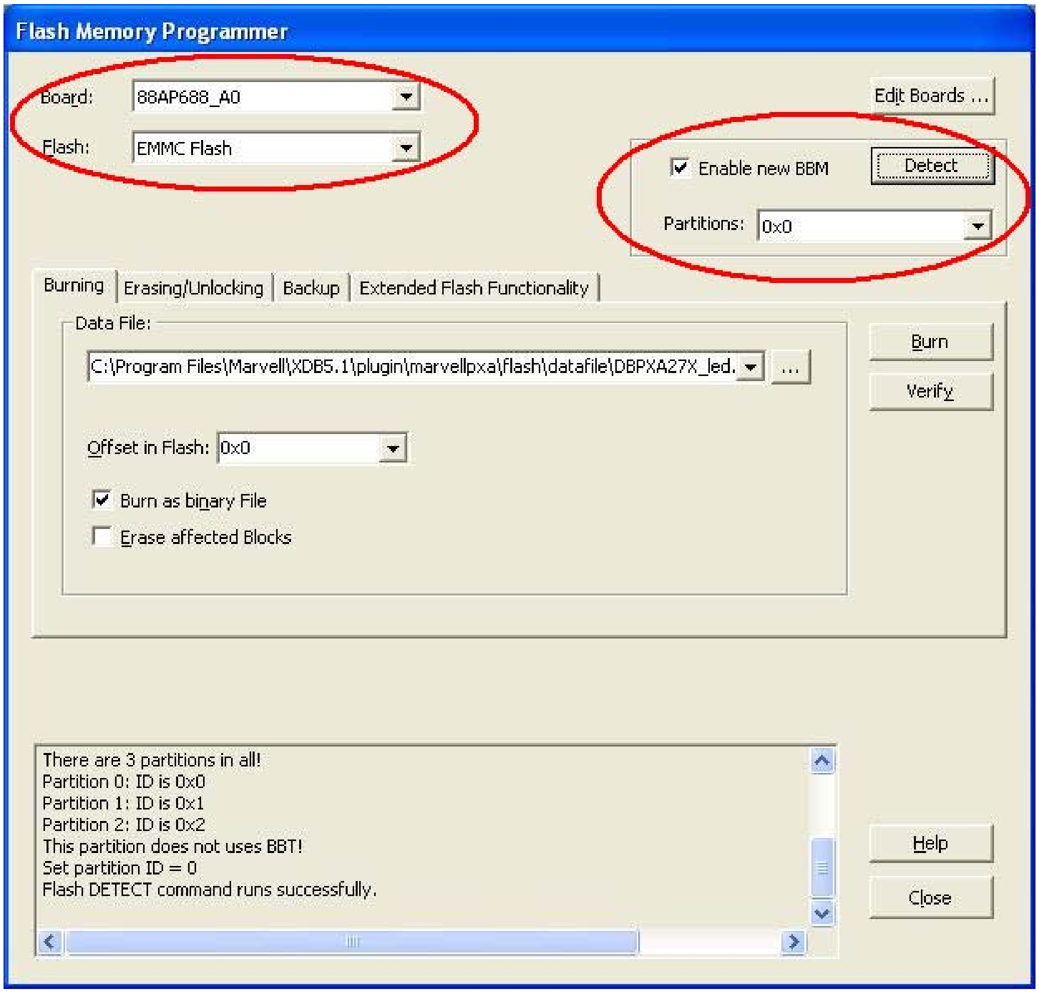
2. Burn OBM and U-Boot into eMMC flash memory using XDB:
   a) In XDB, select **Flash** on the toolbar menu. Click **Burn Flash** on the drop-down menu.
   b) In the **Board** field, select **88AP688_A0**, and in the **Flash** field, select **EMMC Flash** (see Figure 4 on page 22).

Note

If you are using the 88AP688 Z0/Z1 stepping processor, select **88AP688_ZX** in the **Board** field instead.

   c) Check the **Enable new BBM** check box and click the **Detect** button to detect the eMMC partition (see Figure 4).

Figure 4: eMMC Flash Detect eMMC Partition



Board: 88AP688_A0

Flash: EMMC Flash

☑ Enable new BBM

Partitions: 0x0

Data File: C:\Program Files\Marvell\XDB5.1\plugin\marvellpxa\flash\datafile\DBPXA27X_led.

Offset in Flash: 0x0

☑ Burn as binary File

☐ Erase affected Blocks

```
There are 3 partitions in all!
Partition 0: ID is 0x0
Partition 1: ID is 0x1
Partition 2: ID is 0x2
This partition does not uses BBT!
Set partition ID = 0
Flash DETECT command runs successfully.
```

d) Select partition **0x1**. Burn the following binary files to the addresses as follows:

```
ntim_platform_512m_ddr3.bin          --> 0x0
Wtm_rel_mmp2.bin                     --> 0x4000
MMP2_LINUX_ARM_BL_3_2_21_EB_JO.bin   --> 0x30000
u-boot.bin                           --> 0x60000
u-boot.bin (as recovery u-boot)      --> 0xB0000
```

e) Select partition **0x0**. Burn the following binary file to the address as follows:

```
dntim_platform.bin     --> 0x780000
```

In step 2d, the `Wtm_rel_mmp2.bin` image is a loadable WTM kernel firmware binary image that is executed by the ARMADA 610 on-chip secure processor. This image provides the cryptographic services for both Federal Information Processing Standard (FIPS) and non-FIPS mode operations.

To acquire the image, download the `WTM_Firmware_ARMADA610_2.1.8.3.zip` file from the Marvell Extranet at *My Products/Cellular & Handheld Solutions/Applications Processors/ ARMADA 610 (MMP2)/Software/WTM/Version 2.1.8*.

Once downloaded, extract both `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin` and `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin` images from the zip file. Both binary images support the same set of WTM primitive functions with the same API definition. However, the `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin` binary image performs the device RKEK/EC521-DK provision with primitives using the buffers within the secure SRAM to emulate the provisioning over the FUSE/OTP macro. On the other hand, the `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin` image performs the device key provision with primitives directly operating over the FUSE/OTP macro. With real OTP operation, the performed platform provision becomes permanent.

It is recommended to use the virtual OTP version of the WTM kernel binary image for platform software development. To do this, change the binary image file name `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin to Wtm_rel_mmp2.bin`.

It is recommended to use the real OTP version of the WTM kernel binary image for the device that is ready to be deployed as a product. To do this, change the binary image file name `wtm_rel_mmp2_realOTP_2.1.8.3.MP.bin to Wtm_rel_mmp2.bin`.

2.1.4 Burning Android on eMMC Using U-Boot

Use the U-Boot commands that follow to tftp `zImage.android`, `primary_gpt_8g`, `secondary_gpt_8g`, `ramdisk.img`, and `system.img` to eMMC flash memory.

Note

The kernel, recovery kernel, ramdisk and recovery ramdisk *must be burned using JTAG in trusted mode* while in non-trusted mode, they can be burned using JTAG or U-Boot.

The following procedures require the connection of the USB cable for USB Ethernet between the host Linux PC and the ARMADA 610 Tablet Reference Platform.

After the first TFTP command is issued, use the ifconfig utility to set the host side USB0 connection to `192.168.1.100`. The Network File System (NFS) server is fixed as `192.168.1.100:/nfs/android`. If you want to change it, modify `vendor/marvell/brownstone/rootdir/rdinit` and build the kernel again.

You do not see the USB0 Ethernet interface on the Linux host until AFTER the first TFTP “t” command is issued from U-boot.

1. By default, the initial partition is set to 0. Alternately, you can run the following instructions to make sure.

```
MMP2       --> mmc sw_part 0
```

2. Burn GPT.

```
MMP2       --> t primary_gpt_8g
MMP2       --> mmc write 0 0x22 0x1100000
MMP2       --> t secondary_gpt_8g
MMP2       --> mmc write 0xecafdf 0x21 0x1100000
```

3. Burn the kernel

```
MMP2       --> t zImage.android
MMP2       --> mmc write 0x4c00 0x2000 0x1100000
```

4. Burn the ramdisk image.

```
MMP2       --> t ramdisk.img
MMP2       --> mmc write 0x8C00 0x800 0x1100000
```

5. Burn the system image.

```
MMP2       --> t system.img
MMP2       --> unsparse 0x9ec00 0x80000 0x1100000
```

6. Burn the userdata image.

```
MMP2       --> t userdata.img
MMP2       --> unsparse 0x11ec00 0x500000 0x1100000
```

7. Recover the images.

```
MMP2      --> t zImage_recovery.android
MMP2      --> mmc write 0xcc00 0x2000 0x1100000
MMP2      --> t ramdisk_recovery.img
MMP2      --> mmc write 0x10c00 0x800 0x1100000
```

Note

When you are done, power off the ARMADA 610 Tablet Reference Platform and power it on again to boot from eMMC.

Rebooting by pressing the reset button does not work.

2.2 Use the Patch-based Source Code

The source code package includes the code for the kernel, U-Boot, Android, and everything needed to boot Android on an ARMADA 610 Tablet Reference Platform, Revision 5 with the ARMADA 610 processor with A0/A1/A2 stepping and 1GB of DDR3 memory.

The Android code is provided as a group of patches based on a certain version of Android source code. A manifest file is provided to download that version of Android code from the Android Open Source Project (AOSP) at http://source.android.com.

The kernel and U-Boot are provided as a tar ball of base code and a tar ball of patches Marvell made on it.

2.2.1 Setting Up the Android Working Directory

Use the steps that follow to set up the code base.

Note

Check the version of your Git. You can do this by typing `git version`. If the Git version is 1.6.x.x. You can go ahead with it.

If the Git version is 1.7.1.x or later, open the `~/.gitconfig` file and add the following section:

```
[am]
    keepcr=true
```

If the Git version is 1.7.0.x, upgrade your Git to a version later than 1.7.1.x. You can download the package from http://git-scm.com/download.

1. Go to http://source.android.com to download the "repo" tool and set up the build environment for Android.
2. Create the Android working directory and download the initial code base.

```
$ mkdir <android_working_dir>
$ cd <android_working_dir>
$ repo init -u ssh://partner.source.android.com:29418/platform/manifest -b
honeycomb-mr2-release
$ repo sync
```

3. Switch the code base specified by `marvell_manifest.xml`.

```
$ cp <installed_source_dir>/marvell_manifest.xml .repo/manifests/
$ repo init -m marvell_manifest.xml
$ repo sync
```

4. Apply the Marvell patches:

```
$ cd <installed_source_dir>
$ ./setup_android.sh <android_working_dir>
```

2.2.2 Building the Source Code

This section provides two ways to build the source code:

- Section 2.2.2.1, Build the Source Code Using a Script
- Section 2.2.2.2, Build the Source Code Manually Using Commands

2.2.2.1 Build the Source Code Using a Script

Use the following script to build all the source code images in one step:

```
$ cd <android_working_dir>
$ ./build.sh user // Select "user" as the parameter if you are a normal user.
Select "userdebug" or "eng" to specify what you want to do. If you do not
specify a parameter, the default value is "userdebug".
```

2.2.2.2 Build the Source Code Manually Using Commands

Use the steps that follow to build Android.

1. Set up the build environment:

```
$ cd <android_working_dir>
$ . build/envsetup.sh
$ chooseproduct brownstone
$ choosevariant <build variant> //Select "user" as the build variant if you are
a normal user. Select "userdebug" or "eng" to specify what you want to do.
$ export ANDROID_PREBUILT_MODULES=kernel/out/modules/
```

2. Build the kernel and modules:

```
$ cd <android_working_dir>
$ cd kernel
$ make all
```

Note

The location of the zImage is at `kernel/out/`and the location of the modules is at `kernel/out/modules`.

3. Build Android:

```
$ cd <android_working_dir>
$ make -j4
```

Note The location of the Android GPTs, `ramdisk.img`, `system.img`, and `update_droid.zip` files are at `out/target/product/brownstone`.

4. Build U-Boot and OBM:

Before building U-Boot and OBM, extract `wtm_rel_mmp2_virtualOTP_2.1.8.3.MP.bin` from `WTM_Firmware_ARMADA610_2.1.8.3.zip` which you can get from the Marvell Extranet. Rename it as `Wtm_rel_mmp2.bin` and copy it to `boot/obm/binaries`.

Note The WTM firmware image (currently version 2.1.8.3) must be downloaded separately from the Marvell Extranet at *My Products/Cellular & Handheld Solutions/Applications Processors/ARMADA 610 (MMP2) Software/WTM/Version 2.1.8*. Contact your Marvell representative if you have issues about the download.

Issue the following commands:

```
$ cd <android_working_dir>
$ cd boot
$ make all
```

Note The `u-boot.bin` and `OBM` files are at `boot/out/nontrusted`, while the unified WTM files are at `boot/out/`.

5. Build the update packages:

```
$ cd <android_working_dir>
$ make droidupdate
```

Note The `update_droid.zip` and `update_recovery.zip` files are at `out/target/product/brownstone`.

2.2.3 Using the Optimized OpenSSL for Marvell Platforms

An OpenSSL patch is included in this platform release. This patch fixes a SHA384/512 bug in Android OpenSSL, and optimizes the Android OpenSSL cryptographic library for Marvell platforms. By default, the SHA384/512 bug fix is enabled, the optimization is disabled.

To enable the optimization, this patch requires the Marvell Wireless Trusted Platform Service Package (WTPSP) in this platform release. See the WTPSP release notes for information about how to enable it. Contact your Marvell representative if you have any issues with this package.

When the optimization is enabled, the following OpenSSL cryptographic schemes are optimized: SHA1/224/256, MD5 message digest, AES (CBC mode), RC4 and DES (CBC, CBC3 mode).

Note

Whether the optimization is enabled or not enabled, the Android OpenSSL cryptographic API stays unchanged. Thus, applications using the OpenSSL cryptographic library do not need to be modified. However, an application rebuild is required when the optimization is enabled.

To enable Marvell optimization for OpenSSL:

1. Put the Marvell WTPSP middleware library and header file into `<android_working_dir>/external/openssl/crypto/` and rename it as `libwtpsp.a`.
2. Add the following into `external/openssl/include/openssl/opensslconf.h`.

```
#ifdef _ARM_
#ifndef OPENSSL_MRVL
#define OPENSSL_MRVL /* enable marvell crypto support */
#endif
#endif
```

3. Add the following into `vendor/marvell/brownstone/BoardConfig.mk`.

```
USE_MARVELL_CRYPTO := true
```

4. Build the Android system image.

2.3 Protect the HDCP Key

The ARMADA 610 platform supports the high-bandwidth digital content protection (HDCP) protocol that is communicated across the display port, digital visual interface (DVI), high-definition multimedia interface (HDMI), gigabit video interface (GVIF), or unified display interface (UDI) connection.

To protect the device unique HDCP key set (40 keys with each 56 bits) used for streaming multimedia data ciphering, as well as the key select vector (40-bit KSV) for the HDCP authorization, HDMI/HDCP service in the Android package is capable of wrapping the HDCP key set during device HDCP provisioning through the WTPSP and WTM, and loading the HDCP key set into the platform HDCP key register during the platform booting sequence through the WTPSP and WTM.

Note

Marvell does not provide any HDCP key. To enable HDCP, get the HDCP key from Digital Content Protection, LLC: http://www.digital-cp.com/

To protect the HDCP key on the ARMADA 610 platform, use the steps that follow.

1. When building your Android code base, set up a proper configuration file in `/system/etc/HDCP/config`, indicating the location of the plaintext HDCP key file and where to save the wrapped cipher HDCP key file.
2. On the product line when Android boots, push the plaintext HDCP key file into the device at `/plaintext_key_file_path/plaintext_key_file_name` indicated by the configuration file through ADB.

   In the next power cycle when Android boots and the HDMI service is launched, the plaintext HDCP key is automatically wrapped to a cipher HDCP key. The cipher key file is saved at the `/cipher_key_file_path/cipher_key_file_name` indicated by the configuration file. The plaintext key file is deleted.

   Once the HDCP key is wrapped, HDMI service loads the cipher key into the platform HDCP key registers.

A sample of the configuration file:

```
/HDCP/p_key.img;

/HDCP/c_key.img;
```

Note

The format of the plaintext HDCP key, `p_key.img`, should be 5 bytes of the KSV + 3 bytes of 0x00 + 280 bytes of private keys.

Note

HDCP key wrapping can be done multiple times, if desired, on the product line in the Device Manufacturing (DM) life cycle.

THIS PAGE INTENTIONALLY LEFT BLANK

3 Recovery and Updates

This section provides procedures for

- Using the recovery and update features
- Checking device requirements
- Checking the kernel and U-Boot requirements
- Customizing the update

3.1 Recovery and Update Features

The following sections provide information about the following features:

- Burning recovery images
- Entering recovery mode
- Performing a factory reset
- Using the update package
- Using fastboot

3.1.1 Burning Recovery Images

To enable the recovery and update features, two additional images are required:

- `zImage_recovery.android`
- `ramdisk_recovery.img`

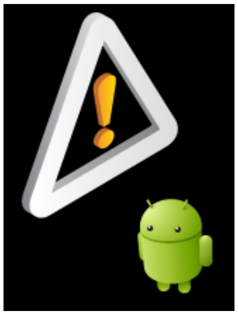
Burn these images to eMMC flash memory as follows.

```
MMP2 --> t zImage_recovery.android
MMP2 --> mmc write 0xcc00 0x2000 0x1100000
MMP2 --> t ramdisk_recovery.img
MMP2 --> mmc write 0x10c00 0x800 0x1100000
```

3.1.2 Entering Recovery Mode

On the Marvell ARMADA 610 Tablet Reference Platform, the system enters Recovery mode if the user powers on the device and presses the Volume Down button. An icon is displayed when the system enters Recovery mode.

Figure 5: Recovery Mode Icon



Pressing the Power button switches the screen between the Recovery menu and the background icon.

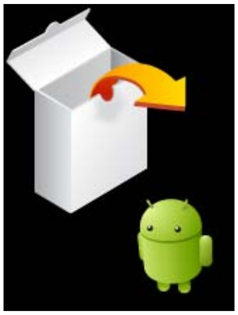
The Recovery menu contains four options:

- reboot system now
- apply update from /sdcard
- wipe data/factory reset
- wipe cache partition

Use the Volume Up button to move the highlight to the desired option. Use the Volume Down button to select the option.

When the “apply update from /sdcard” option is selected and the system is installing the update, the background is set to the installing icon (see Figure 6).

Figure 6: Installing Icon



3.1.3 Performing a Factory Reset

The factory reset is a user interface interaction which erases data and the cache partition. If "Settings => Privacy => Factory data reset" is selected, the system reboots in Recovery mode to erase data and cache.

3.1.4 Using the Update Package

To update using the SD card, put the update package on the SD card and reboot the device by pressing the appropriate button to enter Recovery mode (see Section 3.1.2, Entering Recovery Mode, on page 31). Then, select the menu option "apply update from /sdcard," select the update package, and start the update.

To update in Over-the-Air (OTA) mode, implement an Android service to check if a new update package is ready in the OTA server, download the update package to the device, and reboot the device in Recovery mode to process the update.

3.1.5 Using Fastboot

Fastboot is a protocol used to update the flash filesystem in Android devices from a host over USB. The system enters the Fastboot mode if the user powers on the device and presses the Volume Up button. A partition table is displayed in the host console when the system enters the Fastboot mode.

To make use of fastboot, you need the fastboot program compiled for your host computer (the location should be `out/host/linux-x86/bin/fastboot)`.

Here are the commands you can run on your host after fastboot has been started on a device connected via USB. You can get command hints from the fastboot binary's "--help" command:

Usage: `fastboot [ <option> ] <command>`

Commands:

`flash <partition> [ <filename> ]`	Writes a file to a flash partition
`erase <partition>`	Erases a flash partition
`reboot`	Reboots the device normally

Options:

`-w`	Erases userdata and cache
`-s <serial number>`	Specifies the device serial number
`-p <product>`	Specifies the product name

You can use fastboot to burn all images to eMMC flash memory as follows. If the target image is in current path, you can omit the filename option.

```
# fastboot flash kernel [<image path>/zImage.android]
# fastboot flash ramdisk [<image path>/ramdisk.img]
# fastboot flash system [<image path>/system.img]
# fastboot flash reckernel [<image path>/zImage_recovery.android]
# fastboot flash recovery [<image path>/ramdisk_recovery.img]
```

You can put all images in the current path and use `flashall` to burn all images with one command:

```
# fastboot flashall
```

3.2 Device Requirements

3.2.1 Key Layout

The ARMADA 610 Tablet Reference Platform boots normally when the On/Off power button is pressed. The device enters a Recovery mode if the user presses the appropriate button.

Several other keys (buttons) are required to operate the menu in the Recovery mode, such as the Volume Up button for moving the highlight to an option, the Volume Down button for selecting an option, and so on. See Section 3.1.2, Entering Recovery Mode, on page 31 for a description of these buttons.

For the location and description of the ARMADA 610 Tablet Reference Platform, Revision 5 buttons, see the *ARMADA 610 Tablet Reference Platform, Revision 5 User's Guide*, MV-L100855-10.

3.2.2 eMMC Partition

To enable the recovery feature, four more partitions must be added:

- kernel recovery partition
- RAM disk recovery partition
- misc partition
- cache partition

The misc partition is used for interaction between recovery and the BootLoader. The cache partition is used for interaction between recovery and the main system.

Use the GUID Partition Table (GPT) partition method instead of Master Boot Record (MBR) to support more partitions. Use the "parted" command to generate the GPT table; include the primary and secondary GPT header.

1. Make a partition image with the same size as the on-board flash memory. For example, generate an 8 GB image file, `gpt.img`:

```
# dd if=/dev/zero of=gpt.img bs=512 count=15511552// 15511552blocks in 8GB
flash, 512 bytes per block
```

2. Format and partition the image file:

a) Enter the partition program.

```
# parted gpt.img
```

b) Format the image file as GPT.

```
# (parted) mklabel gpt
```

c) Change the unit to byte.

```
# (parted) unit b
```

d) Add a new partition "dtim".

```
# (parted) mkpart
# Partition name? [ ]? dtim
# File system type? [ext2]?
# Start? 7864320// 0x3c00 × 512 = 18350080
# End? 8912895        // 0x4400 × 512 - 1 = 8912895
```

e) You can also rename the partition.

```
# (parted) name 1     // the number is the index number of the partition
# Partition name? [dtim] dtim
```

f) Add more partitions; follow the previous steps.

Note

1. At any time, you can use the "print" command to check the partition information.

```
# (parted) print
```

2. The order you create new partitions is the order listed in the GPT table. It is the order recognized in uboot/kernel.

   If you create partitions in the following sequence: dtim -> kernel -> ramdisk ->… Then, partition "dtim" is mmcblk0p1, "kernel" is mmcblk0p2, "ramdisk" is mmcblk0p3, … .

3. At any time, you can type "?" for more details for help.

```
# (parted) ?
```

g) Quit the partition program.

```
# (parted) quit
```

4. Get the GPT table from the image file:

```
# dd if=gpt.img of=primary_gpt bs=512 count=34
```

```
# dd if=gpt.img of=secondary_gpt bs=512 count=33 skip=15511519
```

On the Marvell ARMADA 610 Tablet Reference Platform, the partition table is as follows.

Image Name	Start Block Address	End Block Address	Block Size	Byte Size	Logical Partition	Partition Name	Type[1]
DTIM	0x3c00	0x4400	0x800	1 MB	mmcblk0p1	dtim	ext2 (not formatted)
Kernel	0x4c00	0x8c00	0x4000	8 MB	mmcblk0p2	kernel	ext2 (not formatted)
Ramdisk	0x8c00	0xcc00	0x4000	8 MB	mmcblk0p3	ramdisk	ext2 (not formatted)
Recovery Kernel	0xcc00	0x10c00	0x4000	8 MB	mmcblk0p4	kernel_r	ext2 (not formatted)
Recovery Ramdisk	0x10c00	0x14c00	0x4000	8 MB	mmcblk0p5	ramdisk_r	ext2 (not formatted)
Misc	0x14c00	0x1ec00	0xa000	20 MB	mmcblk0p6	misc	ext2 (not formatted)
Cache	0x1ec00	0x9ec00	0x80000	256 MB	mmcblk0p7	cache	ext4
Android System	0x9ec00	0x11ec00	0x80000	256 MB	mmcblk0p8	system	ext4
Android Data	0x11ec00	0x61ec00	0x500000	2.5 GB	mmcblk0p9	userdata	ext4
Mass Storage	0x61ec00	0xecac00	0x8ac000	4440 MB	mmcblk0p10	m_storage	fat32

[1] "Type" is the "File system type" in the "mkpart" command that generates the GPT table. The partition is raw if it is not formatted.

3.3 Kernel and U-Boot Requirements

The following sections list kernel and U-Boot requirements.

3.3.1 Kernel Requirements

The `reboot()` system call with the "recovery" parameter must set a register bit to inform the BootLoader to enter Recovery mode and to reset the bit after entering Recovery mode.

3.3.2 U-Boot Requirements

U-Boot must perform the following procedures:

- Detect the appropriate key and enter the Recovery mode if the Volume Down button key is pressed.
- Check the recovery indication register bit and enter the Recovery mode if the bit is set, then reset the bit.
- Read the misc partition, if the buffer is equal to "boot-recovery", enter the Recovery mode; if the buffer is equal to "update-firmware," get the BootLoader image from misc and reflash itself.
- Support fastboot protocol. Detailed information is at the following location:
  `bootable/bootloader/legacy/fastboot_protocol.txt`

3.4 Update Customization

The following sections describe how to customize generating the update package, signature, and certification.

3.4.1 Generating the Update Package

The update package is a ZIP file containing an update binary, an update script, some certification information, and all the files needed for the update. The directory structure is as follows.

```
|-- META-INF
|   |-- CERT.RSA
|   |-- CERT.SF
|   |-- MANIFEST.MF
|   |-- com
|       |-- google
|           |-- android
|               |-- update-binary
|               |-- updater-script
|-- system/…
|-- <other folders>/…
|-- zImage
|-- <other images>
```

The update packages can be generated automatically in the Android system build. In the system, the update binary and all the files needed for the update are copied to a specific folder. In addition, the update script is generated by several Python tools.

If you want to customize an update package, for example, just update one or several `.so` files or `apk` files, generate the package manually as follows.

1. Make an update folder and copy all folders and files into it:

   ```
   # mkdir <update folder>
   ```

   ```
   # cp -p -r <folds and files need to update> <update folder>
   ```

2. Make a meta info folder, copy the updater binary to it, and create an update script:

   ```
   # mkdir -p META-INF/com/google/android
   ```

3. Compress the update folder:

   ```
   # zip -qry ../update.zip .
   ```

3.4.2 Generating the Signature and Certification

By default, the update process uses the Android test key pair to sign the update package, `testke.x509.pem` and `testkey.pk8` in `build/target/product/security/`. The signature using the private key is done in the build process and the public key is dumped to the recovery RAM disk at the same time. The recovery process verifies the update package before extracting and upgrading the package.

To use a customized key, follow these steps:

1. Generate the private key:

   ```
   # openssl genrsa -3 -out customer_key.pem 2048
   ```

2. Generate the certification:

   ```
   # openssl req -new -x509 -key customer_key.pem  -out customer_key.x509.pem
   -days 10000
   ```

3. Turn the private key to PKCS #8 standard:

   ```
   # openssl pkcs8 -in customer_key.pem -topk8 -outform DER -out customer_key.pk8
   -nocrypt
   ```

4. Sign the `update.zip` by `<android>/out/host/linux-x86/framework/signapk.jar`:

   ```
   # java -Xmx512m -jar signapk.jar -w customer_key.x509.pem customer_key.pk8
   update.zip update_signed.zip
   ```

5. Using `<android>/out/host/linux-x86/framework/dumpkey.jar`, dump the public key and build in the recovery ramdisk:

   ```
   # java -jar dumpkey.jar customer_key.x509.pem > keys
   ```

THIS PAGE INTENTIONALLY LEFT BLANK

4 Marvell Code Performance Analyzer

The Marvell Code Performance Analyzer v2.3 is supported in this release.

4.1 What's New

- Call Stack sampling data collection and corresponding data analysis
- Remote data collection via the Android Debug Bridge (ADB) for the Android device

4.2 Features

The following sections provide information about the supported and unsupported features for this release.

4.2.1 Supported Features

- Remote data collection via TCP/IP
- Remote data collection via ADB for the Android device
- Target local data collection in a connectionless environment
- Call Stack sampling data collection and corresponding data analysis
- Hotspot sampling data collection and corresponding data analysis
- Counter monitor data collection and corresponding data analysis.
- Real-time counter monitor and post analysis

4.2.2 Unsupported Features

The software development kit (SDK) for dynamic code is not supported in this release.

4.3 System Requirements

This release supports the Marvell Code Performance Analyzer, version 2.3. Download this version from the Marvell Extranet website at http://www.marvell.com/extranet. If you do not have a Marvell Extranet user ID, click on the “register” link at http://www.marvell.com and follow the instructions therein.

4.4 Installation

Before running the data collector, go to the `/system/bin` folder and run the following command to load the kernel driver:

```
$ ./load_mpdc.sh
```

4.5 Known Issues or Limitations

- Before loading the kernel driver with the `load_mpdc.sh` command, run the `su` command on the board. This prevents an “operation not permitted” message.
- It is recommended to first turn off the Marvell Scalable Power Management (MSPM). Otherwise, the Marvell Performance Data Collector (mpdc) may not work normally. Use the following commands to turn off MSPM on your target:

  ```
  echo 0 >/sys/power/mspm/mspm
  ```
- When using the command line, if you get the error message "Fail to communicate with daemon:Success", reload your activity and run it to collect the result again.

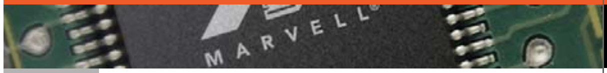
- When doing remote data collection on the Android target, if the analyzer exits abnormally, the status of the mpdc_svr may still be connected. If this happens, restart the mpdc_svr on the target.
- If the samples/second value is set too large, the events/sample value is adjusted after a calibration to a smaller value, which makes the interrupt happen more frequently. If this happens, the system becomes busy and the mpdc stops after the expected duration is expired. In addition, most of the samples will be located on the process "mpdc_d". In this case, create your activity with a decreased samples/second value and start it.
- When using the command line, multiple activities cannot be started at the same time.
- If you want to make mpdc_svr listen to another port through using the command "mpdc_svr -p <PortNumber>" on the Android target, a segmentation fault occurs.
- Do not set OS_TIMER as the event in the Event-Based Sampling (EBS).

A Revision History

Date	Revision	Description
November 17, 2011	-	Initial release.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK



Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.988.8279

www.marvell.com

Marvell. Moving Forward Faster