Android™ on i.MX Applications Processors

Sridharan Subramanian
Senior Product Manager – Software and Platforms
Abstract

Android™ is a software platform and operating system for mobile devices, based on the Linux® kernel, developed by Google and later the Open Handset Alliance. It is penetrating in different target segments from mobile devices to netbooks to even Automotive. Freescale is providing Android solutions on its i.MX processors and this session will go into details of the port.
Contents

► Freescale i.MX Processor Linux Approach

► i.MX Processor Android

► i.MX Processor Android Integration Details

► i.MX Processor Android Validation
Freescale i.MX Processor Linux Approach
Freescale i.MX Processor Linux Approach

► A layered approach with the right selection of components to provide a common Linux stack that enables customers to easily migrate across i.MX SoCs and penetrate the markets that they want to engage in.

► Common board support packages (BSP) consisting of kernel optimizations, hardware drivers and unit tests; Customizations for specific target segments.

► Extensive portfolio of optimized multimedia codecs (ARM and Video processing unit accelerated).

► Common middleware with reuse from open source frameworks that enable integration of proprietary components also - multimedia framework plugins, power management, security/DRM, graphics (OpenVG/OpenGL-ES), connectivity.

► Leverage Ubuntu, Android and ARM® Linux (Gnome Mobile) solutions.
i.MX Processor Linux BSP

► Features
  • Out-of-the-box integrated Linux environment – tools + kernel + drivers
  • Extensively tested, hardened and validated
  • Optimized for target platforms
  • Accelerated codecs support
  • Common code base across different i.MX SoCs

► Packages
  • Boot loader binaries and source files
  • Source and patches for kernel and file system
  • Source code for unit tests of the drivers
  • Linux Target Image Builder (LTIB)
  • Proprietary 3rd party components in binary code format
  • Prebuilt binaries
  • Tool chain for ARM9/ARM11/ARM12 (Linux – Open source/Codesourcery)
  • BSP Documentation (Reference manual, User’s Guide, release notes)
i.MX Optimized Multimedia Codecs

► Key Features
  - Comprehensive suite of optimized codecs (~40+ Audio/Video/Image Codecs)
  - Highly optimized software that is coded by Freescale processor experts
  - Consistent application programming interface (API) and frameworks across all software packages including OpenMAX support
  - Codec APIs have been optimized from system design perspective and achieve optimal system performance along with related middleware wrappers
  - Supplemented with Freescale development tools, sample test streams and documentation

► Codec Software Packages include:
  - Codec libraries with a standard C-callable API
  - OpenMAX plugins that provide an API layer between the multimedia framework and the codec library
  - Audio/video file containers (parsers) that support popular multimedia content, such as .aac, .avi, .asf, .mp3 and .mp4 files
  - Bundle of Freescale audio/video sample test streams
  - Complete documentation, including API documentation, release notes and data sheets
i.MX Processor – Android
What Android Is and Is Not

► It’s a software stack for mobile devices including **OS (Linux)**, **middleware and key applications**.
► It’s a different Linux OS (or “distribution”) based on Linux kernel. The system libraries, system initialization and program interface in it are distinct from a “standard” Linux OS.
► It’s not ONLY an application framework for Linux, although it does include it’s own app framework (window management, inter-app communication, event dispatch, …). **It’s much more**.
► It’s not ONLY a Java API for phone, although it does include a Java virtual machine (called “Dalvik”) and all system interfaces are exposed only through Java libraries. **It’s much more**.
► It’s not a full phone stack. It’s ONLY SW running on application CPU. It will interact with wireless protocol (GSM/GPRS/WCDMA/…) running on separate baseband chip to implement telephony features.
Android – Software Stack on Mobile Device

Apps (Java) – Everyone can create his/her own application based on “Open” Android API

Android “Program” API

Middleware (Java) – App framework including window/focus management, inter-app communication, event notification, etc.

Middleware (C/C++) – system libraries for media, graphic, database, font, web engine, etc.

Android “Porting” I/F

2.6 based Linux kernel with Android patch
i.MX Processor Android Approach

► Readiness
  • An “integrated” solution (kernel + Android framework + dev/debug environment). Customer is able to directly develop applications on this “integrated” solution or easily modify/replace their own drivers based on reference code.

► Performance
  • Higher performance by utilizing hardware acceleration for graphics and multimedia and optimized codecs.

► Contribution
  • Participate and contribute in the OHA community.
Multimedia – Audio/Video Codec

APPLICATIONS
- Audio/Video Apps

APP FRAMEWORK
- MediaPlayer/Recorder Class

RUNTIME -> NATIVE
- JNI

NATIVE LIBRARIES
- MediaPlayer Service (Native) → Media Library
- PV OpenCore Player/Author Engine
- PV MediaInput Node
- PV SW Codec/Parser
- PV MediaOutput Node
- FSL OMX Core
- OpenMAX IL I/F
- FSL Audio Codecs with OMX IL
- FSL Video Codecs with OMX IL

LINUX KERNEL
- VPU Driver

- Verify audio/video player/authoring is available in Application level.
- Verify audio/video player/authoring work fine in PV engine level via PV test utilities.
- Integrate FSL OpenMAX core.
- Integrate FSL Audio Codecs OMX component.
- Integrate FSL Video Codecs OMX component. If it’s HW codec (e.g. i.MX51), integrate VPU driver also.
Integration details (Bottom-up)

► Multimedia Audio/Video Codecs
  • Integrate low-level video drivers (if h/w accelerated)
  • Integrate Freescale Audio/Video Codecs OMX Components
  • Integrate Freescale OpenMAX core
  • Verify A/V player/authoring works fine in PV engine via PV test utilities
  • Verify audio/video player/authoring is available in Application level (works as expected in Android)

► Camera
  • Leverage low level V4I2 API
  • Use user space lib (Glue IPU/Camera driver with camera service)
  • Interact via JNI to Camera service
  • Utilize android.hardware.camera class
  • Verify Camera Service (parameter setting, preview, still image capture,..etc.) is available in Application level (works as expected)
Integration details (Bottom-up) – Contd.

► GPS
  - Integrate low level Linux GPS driver
  - Interact with the GPS lib through HAL
  - Interact with the user HAL through JNI
  - Verify Location Service is available in Application level (ensure it works as expected in Android)

► WiFi
  - Integrate low level WiFi driver kernel module
  - Setup WPA supplicant daemon
  - User HAL (Update to load WiFi kernel module)
  - Verify WiFi Service (enable, search list, connect,…etc.) is available in Application level as expected (i.e. WiFi HW/driver is properly integrated into Android)
Integration details (Bottom-up) – Contd.

► Bluetooth
  • Integrate low level BlueZ kernel module driver
  • Communicate from A2DP or device service to BlueZ library through dbus
  • Verify Bluetooth Service (service discovering, paring, headset, hands free, A2DP,…etc.) is available in Application level as expected (i.e. Bluetooth HW/driver is properly integrated into Android)

► Graphics
  • For 2D, utilize the SGL graphic engine and surface manager to render to framebuffer
  • For 3D, investigating on integration of OpenGL/ES wrappers
i.MX Processor Android Validation
Freescale provides platform software components under the Android framework, to enable customers develop final Android-based solutions.

Our validation approach consists of validating the key platform pieces that constitute an Android port – which is focused on the HAL, BSP and Codecs.

From a system stand point, we focus on integration and validation of certain components like Codecs, WiFi, Bluetooth, GPS, camera, graphics, and extensions that would feed into the Android stack.
# Android Test Scope

<table>
<thead>
<tr>
<th>Test Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| **System Test** | ► System test: Validate the fundamental functionalities and performance of the system. E.g. power consumption, system power on/off, etc.  
► Application test: Validate the functionality of system supported applications. These include default Android applications like media player, camera, etc. and basic functionalities for BSP relevant features under Android stack. E.g. USB, SD, Keypad, Touchpanel. |
| **Codec Test** | Test the Codecs after integrating them through OpenMax IL into OpenCore. Test scope includes:  
► Quality and Profile Test (Using Gallery for Video and Image test, music player for audio test to do playback test of the test vectors, covering different codec profiles/formats that are supported)  
► Trick-mode test  
► Stress and security test etc. |
| **BSP Test**   | ► BSP tests validate functionality, performance and stability of BSP modules (e.g. drivers, core lib, etc.).  
► Also validate the reliability, robustness, and stability of Linux kernel OS through LTP suite |
Codecs Validation

- Exhaustive bitmatch on target platform for the suite of conformance test vectors
- Platform testing with the platform test suite for bitmatch, re-entrancy, pre-emption, relocation, endurance, robustness and error recovery
- Test streams used for validation (raw and composite streams)
  - Conformance streams
  - Performance/stress streams
  - Error streams
- 3rd party codec/platform certification (Microsoft, DivX, etc.)
- OpenMax interface validation
- System tests for validation in OpenCore and through Android applications like Gallery and Media player
Thank you for attending this presentation. We’ll now take a few moments to review the audience questions, and then we’ll begin the question and answer session.