# Android Software Platform Development at Fujitsu

● Makoto Honda ● Makoto Kobayashi ● Masahiko Nagumo
● Yasuhiro Kawakatsu

**Smartphones using the Android platform first appeared on the market in October 2008. They have since overtaken Apple's iPhone—the first entry in the smartphone market—in number of units shipped and have helped to bring about major changes in the way that mobile phones are used. Android was developed and is distributed as open source software that a device maker integrates into its own hardware after adding original software technologies. The Android platform evolves in short cycles on the basis of software and hardware developments as the network infrastructure continues to expand in the form of WiMAX and LTE and as usage scenarios and services become increasingly diverse. Fujitsu has been developing Android smartphones with compelling functions and enhanced convenience since December 2010, when it released the REGZA Phone T-01C featuring a water-resistant enclosure, one-seg support, and FeliCa contactless IC card and infrared-communication functions. This paper describes Fujitsu's approach to smartphone development, focusing on memory management and current-consumption management as important elements in the system design of the Android software platform, diverse manner modes for enhancing user convenience, high-picture-quality technology achieved by using the Mobile REGZA Engine, and audio-visual device-linking technology based on DLNA standards.**

## 1. Introduction

November 2007 marked the establishment of the Open Handset Alliance (OHA) centered about Google and other firms, accompanied by the announcement by OHA of the Android open software platform. Developed for mobile devices such as smartphones and tablet computers, the Android software platform runs on the Linux operating system. Only one year later, in October 2008, the "HTC Dream" developed by HTC Corporation, a Taiwanese firm, was released by T-Mobile in North America as the first phone product to use the Android platform. This was one year and four months after the appearance of Apple's first iPhone, which took the mobile phone market by storm and launched the worldwide shift to smartphones. Early Android devices did not measure up to Microsoft's Windows Phone or Apple's iPhone in terms of specifications, but thanks to the open-source nature of Android software and ease of programming on the Android platform as well as the evolution of Google Play (Google's marketplace) and Internet services, the smartphone market share of Android devices continued to grow, and in fiscal 2011, it came to exceed 50%.

This onslaught of an open software platform also brought about major changes in the work of developing conventional mobile phones (i.e., "feature phones"). Although a massive amount of source code (about 23 million lines including those for Linux) was released as open source and a board support package (BSP) for software-porting purposes was provided by the chipset vendor, each device maker had to acquire the technical skills needed to make the best use of this platform, which was still a semi-finished product with little in the way of technical documentation. Moreover, not only was Android undergoing either minor or major version upgrades every half-year or so, the device-related requirements were steadily increasing. The HTC Dream that made its market debut in October 2008 was equipped with a single CPU having an operating frequency of 528 MHz, but in three years

time, Android products would come to feature a 1-GHz-class, dual-core CPU, four or five times the amount of memory (1 GB random access memory [RAM]), and six times the display resolution (high-definition [HD] class), as shown in **Figure 1**. Battery capacity, on the other hand, was increased by only 1.5 times for reasons of weight and portability. Power-saving measures are therefore an important issue in Android devices, and it has become necessary to enable switching between operating modes in accordance with the usage scenario without sacrificing performance and to provide power-saving functions that the user can customize. Device makers must therefore develop stable and high-quality products in pace with the evolution of the Android software platform while incorporating original technology and user-friendly features.

In December 2010, Fujitsu released the REGZA Phone T-01C featuring a water-resistant enclosure, one-seg support, and FeliCa contactless IC card and infrared-communication functions. Fujitsu has since been working to improve the usability and performance of its Android smartphones by various means, such as by making the human-centric functions developed and enhanced for feature phones compatible with smartphones, improving the user interface (UI), adopting a dual-core CPU, supporting Long Term Evolution (LTE), Worldwide Interoperability for Microwave Access (WiMAX), and Digital Living Network Alliance (DLNA) standards, and developing functions for linking to audio-visual (AV) devices.

In this paper, we introduce Fujitsu's approach to Android software platform development, focusing on memory management and power-saving measures in system design, an eco function and manner modes implemented in the application framework, and high-picture-quality technology and AV-device-linking technology in multimedia design.

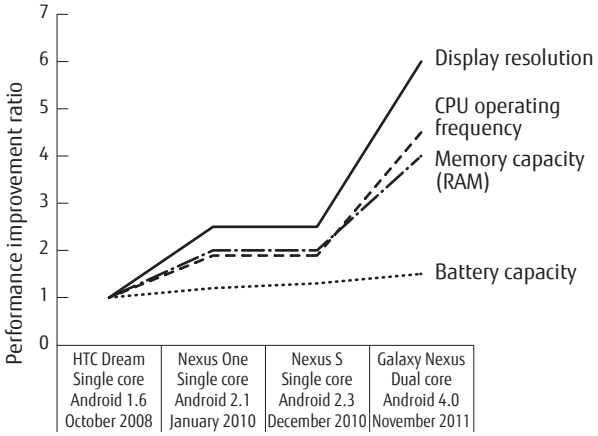


**Figure 1**
**Performance improvement of Android reference devices.**

## 2. System platform technologies

In feature phone development, the device maker handles total system design, from hardware to applications, taking into account stability, consistency, performance, and marketability requirements. In smartphone development, however, the applications developed and installed by the device maker make up only a portion of the applications since most smartphone applications are developed by third parties and are installed by the user after the smartphone purchase. What this means is that a device maker developing a smartphone must design the system so that it will operate stably for the diverse types of applications that might be installed by the user.

Important issues in system design include stability, performance, power saving, operability, compatibility, security, and exception processing. Here we describe memory management, which greatly affects system stability, and current-consumption management, which affects battery life.

### 2.1 Memory management

Total memory size for program execution (i.e., RAM), the amount of memory used by the system ("allocated memory"), and the amount available to the user ("available memory") are shown in **Figure 2** for five Fujitsu-developed devices: REGZA Phone T-01C (released December 2010), F-12C (released August 2011),

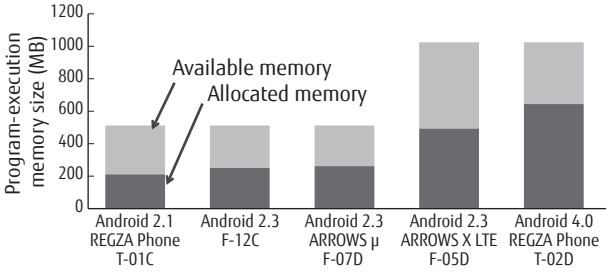


**Figure 2**
**Memory available to user immediately after device startup.**

ARROWS μ F-07D (released January 2012), ARROWS X LTE F-05D (released December 2011) and REGZA Phone T-02D (released July 2012). The total memory size reached 1 GB in the F-05D model, and it remained 1 GB in the T-02D model. However, the amount available to the user actually decreased. There are a variety of reasons for this. In addition to the fact that the amount of memory used by the Android system has been increasing with almost every version upgrade, the amount of memory needed by applications at run-time has been increasing owing to improvements in display resolution and system performance. Moreover, the number of simultaneously running applications has been increasing.

Android provides a mechanism called "low memory killer" (LMK) for efficiently reallocating program-execution memory when the amount available becomes low. It works by releasing and reallocating unneeded allocated memory areas depending on the execution environment. This makes it possible for the user to not only use the available memory shown in Figure 2 but also to use a portion of the already allocated memory. Since total program-execution memory is limited, however, it is desirable that as many unneeded allocated memory areas as possible be released and made available.

Android provides three parameters for managing program-execution memory: LMK threshold, Dalvik heap size, and MAX_HIDDEN_APPS (**Figure 3**). These are summarized below.

1) LMK threshold

The LMK mechanism forcibly terminates low-priority processes running in the background when the amount of available memory falls below a threshold. With Android, application "termination" is not problematic. If multiple applications have been running for some time and memory comes to be needed, LMK simply releases some applications to increase the amount of available memory. The LMK threshold parameter is used for defining when this application-release operation is to be invoked. The amount of memory needed for launching an application is taken into account when setting this parameter.

2) Dalvik heap size

This parameter is used for defining the maximum execution memory size per application. It is defined in Android 4.0 (Ice Cream Sandwich) as a variable called GrowthLimit. The larger the execution memory per application, the smaller the number of applications that can run at the same time. Additionally, an application that uses an amount of memory exceeding GrowthLimit cannot be launched.

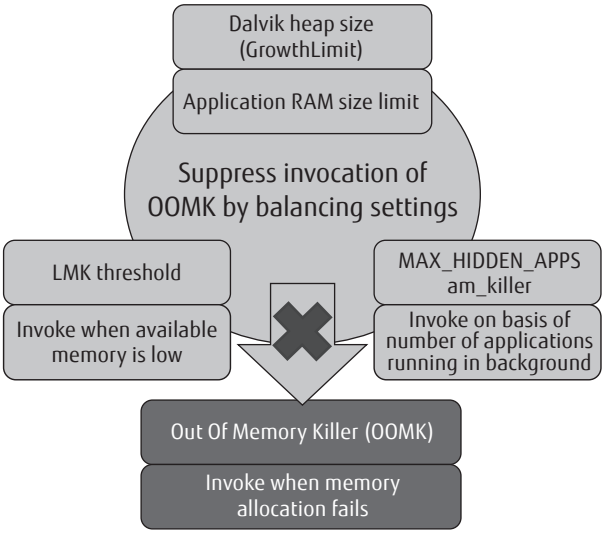


**Figure 3**
**Memory management parameters for process execution in Android.**

3) MAX_HIDDEN_APPS

This parameter is used for defining the maximum number of applications running in the background. It is used to limit the number of launched applications. If the number specified by MAX_HIDDEN_APPS is about to be exceeded, a system process called am_killer releases unnecessary background applications.

Linux too incorporates in its kernel a mechanism called "out of memory killer" (OOMK) for terminating processes when an insufficient amount of memory is available. This mechanism forcibly terminates low-priority processes when memory requests fail owing to insufficient memory. When OOMK is executed, system processes including Android ones may also be terminated, resulting in an unstable system. It is therefore desirable to prevent OOMK from being invoked if at all possible, and one way of doing this is to set the LMK threshold, Dalvik heap size, and MAX_HIDDEN_APPS parameters to appropriate values.

However, as memory size tends to differ for each application and usage scenario, it is not easy to determine the most appropriate values for these parameters at the initial design stage. The following procedure is

therefore used to tune them.

- The parameters are initially set on the basis of the values given by the chipset vendor and by taking into account Android/Linux design information and application trends in the market.
- Preliminary process load tests are performed using a basic software configuration (excluding applications in development), starting with these initial values and adjusting them as needed to achieve uninterrupted operation.
- Final process load tests are performed near the end of the development process using all application integrated and a fully loaded configuration. Further adjustments are made to the parameter settings to achieve uninterrupted operation.

Since performing load tests and analyzing the causes of system instability can be time consuming, reducing the ranges of these settings in the initial development stage is an important design task.

### 2.2 Current-consumption management

All of the 2012 summer smartphone models from the various makers feature a CPU operating frequency exceeding 1 GHz with most models having a multi-core configuration. This presents a problem since faster operating CPUs consume more power while an increasing variety of usage scenarios demand even longer battery lives.

This situation calls for a design that can reduce current consumption by lowering CPU operating frequency as much as possible whenever high-speed processing is deemed unnecessary. Two methods have been adopted as mechanisms for controlling operating frequency:

1) Automatic control on the system side
2) Active control on the application side

The first method reduces power consumption by changing the CPU operating frequency in accordance with the operation and usage conditions. For example, the CPU operating frequency can be lowered while the user is listening to music with the screen off after a certain amount of time has elapsed since the last screen operation. However, usage scenarios differ from user to user, which means that automatically controlling the CPU operating frequency on the system side for all scenarios can detract from the convenience expected of a smartphone. This is where the second method comes into play in the form of a power-saving function that can be set by the user as described in the following section.

## 3. Application framework technologies

Despite the fact that the Android framework is clearly superior, users accustomed to using feature phones claim that some aspects of Android are poor in terms of convenience. Battery life, in particular, is a key issue. As explained in the previous section, current consumption should not be controlled solely on the system side for all usage scenarios—control from an upper layer should also be provided. The following describes a power-saving function as well as new manner modes that Fujitsu implemented within the application framework in its 2012 summer models.

### 3.1 Power-saving technology (eco function)

Fujitsu's 2012 summer smartphone models come equipped with an eco function called "NX!eco" consisting of multiple eco modes for improving control of current consumption and extending battery life. Typical eco modes adjust the brightness of the screen or turn off peripheral devices like those for Wi-Fi, GPS, and Bluetooth functions when remaining battery power reaches a certain level. Fujitsu's NX!eco function provides three eco modes and four eco levels, enabling the user to customize eco operations (**Figure 4**).

One feature is the timer eco mode, which enables the eco level to be changed in accordance with the time periods set by the user. Specifically, the user can set the eco level for each of two time periods. This makes it possible, for example, to automatically switch the power-saving function in accordance with changes in the user's lifestyle between daytime and evening hours.

Another feature is an original eco mode, which enables the eco level to be customized so that suppression of communications from applications running in the background can be finely set. Although Android 4.0 added a function for suppressing background application communications, the function applies to all applications; that is, settings cannot target individual applications. Fujitsu has made it possible to link this communications-suppression function with eco modes so that communications can be suppressed for individual applications. Plus, when making such settings,

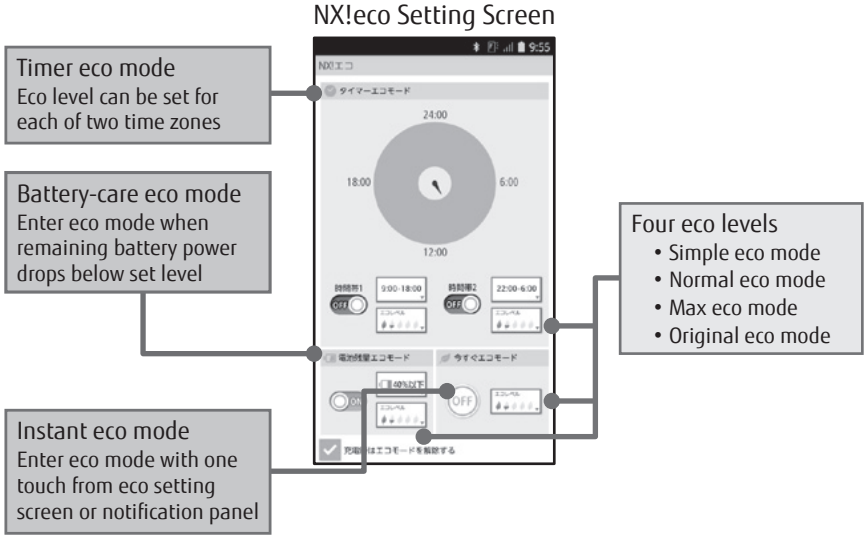


**Figure 4**
**NX!eco power-saving function in 2012 summer models.**

**Table 1**
**Effect of suppressing background communications on current consumption.**
(Average consumed current when two SNS applications and one newsgathering application are running)

| Clock time | 20:00–24:00 | 0:00–3:00 | 3:00–6:00 |
|---|---|---|---|
| Without suppression of background communications | 34 mA | 25 mA | 20 mA |
| With suppression of background communications | 16 mA | 16 mA | 19 mA |
| Reduction in current consumption | 53% | 36% | 5% |

the user is presented with the current amount of data communications for each application as reference for deciding specific settings. Although the results depend on how the feature is used, suppressing background communications in this way can reduce current consumption by as much as half. **Table 1** summarizes the average current consumption for three time periods for an actual usage scenario: two social-networking-service (SNS) applications and one newsgathering application are running. These results show that the effect of suppressing background communications on current consumption is smaller late at night (3:00–6:00), when smartphone usage is typically low, and higher in the evening (20:00–24:00), when smartphone usage is typically high. This means that the user can generate a current-reduction effect by setting this eco function in accordance with the usage scenario.

Some of the 2012 summer models also come equipped with an "application battery diagnosis" function that enables the user to check on applications that are running and consuming battery power even when the screen is off. This monitoring function is installed in the Android framework, which means that monitoring can be performed without applying a load to the system.

## 3.2 Convenience-enhancing technology (manner modes)

Android includes a manner mode as standard enabling an ON/OFF setting to be made for device sounds and for device vibration too. It does not, however, allow individual settings to be made for alarm sounds or for sounds that occur when listening to music or video using earphones or a Bluetooth headset. These were commonplace functions in feature phones, and the lack of them in Android's standard manner mode detracts from smartphone convenience.

Fujitsu's 2012 summer smartphone models come equipped with the following four manner modes to enhance convenience.

- Normal manner mode
- Silent manner mode
- Alarm-On manner mode
- Original manner mode

For example, the alarm-On manner mode means that only the alarm will sound, which makes for a highly convenient mode for users who use the alarm for waking up. The original manner mode, meanwhile, allows the user to make separate settings for vibration, call-ringtone volume, media-playback volume, alarm volume, and notification volume, thereby providing the user with a high level of customization. To achieve these functions, Fujitsu has added new functions to the Android framework for sound output path control and for volume control in the original manner mode.

## 4. Multimedia technologies

The appearance of large-screen, high-performance smartphones and tablet computers and the evolution of a high-speed network infrastructure as in WiMAX and LTE have fueled the growth of a video streaming market for mobile devices. At the same time, Wi-Fi environments have penetrated the home, and AV devices that can connect to mobile devices via such standards as DLNA and High-Definition Multimedia Interface (HDMI) are on the increase. As a result, a need has been felt for technology that can enable users to watch video content stored on their smartphones on external devices with high picture quality. Also of importance in this regard is the fact that multimedia broadcasting for smartphones, tablets, and other mobile devices began operations in April 2012.

Fujitsu equipped its REGZA Phone T-01C released in December 2010 with a function for receiving one-seg broadcasts and installed Mobile REGZA Engine 3.0 developed by Toshiba in the Android framework to enable users to watch video images with quality comparable to that of large screens on their smartphones. Here we introduce Mobile REGZA Engine 6.0 implemented in the REGZA Phone T-02D and DLNA-supporting technology for linking smartphones with AV devices.

### 4.1 High-picture-quality technology (Mobile REGZA Engine)

The evolution of the Mobile REGZA Engine is summarized in **Table 2**. Mobile REGZA Engine 6.0 features active noise reduction, which achieves optimal noise removal by extracting noise features included in video through video analysis and appropriately switching the magnitude of the noise reduction. It also features a sharpness equalizer that identifies contour and texture sections in the video through video analysis and performs contour-compensation processing using optimal parameters for each area of the video image. These techniques have not been limited to video applications such as existing one-seg broadcasts—they have also been applied to still-image applications, thereby expanding the usage range of high-picture-quality technology.

### 4.2 AV-device-linking technology (DLNA function)

Fujitsu's ARROWS Tab LTE F-01D tablet released in October 2011 was equipped with DLNA and Digital Transmission Content Protection over Internet Protocol

**Table 2**
**Evolution of Mobile REGZA Engine.**

| | 2010 Winter/Spring | 2011 Summer | 2011 Winter/Spring | 2012 Summer |
|---|---|---|---|---|
| Version | 3.0 | 4.0 | 5.0 | 6.0 |
| Super resolution | ○ | ○ | ○ | ○ |
| Frame interpolation | ○ | ○ | ○ | ○ |
| Noise reduction | 2D | 2D | 3D | 3D |
| Active noise reduction | – | – | – | ○ |
| Color management | 2D | 2D | 3D | 3D |
| Mobile texture realizer | – | ○ | ○ | ○ |
| Sharpness equalizer | – | – | – | ○ |

(DTCP-IP) support to enable users to watch TV broadcast programs stored on video recorders on a tablet computer. To this end, we made original additions to the Android 3.0 framework used in the F-01D tablet in the form of an MPEG-2 codec with DTCP-IP support for transferring broadcast data from an AV device. DTCP-IP copyright-protection technology is used for safely transferring broadcast data. Furthermore, for the REGZA Phone T-01D released in 2011 winter-spring, we installed a function for transferring content currently being broadcast to the smartphone for real-time viewing and a function for storing 720p-recorded content on the smartphone for viewing on the go. We also developed original real-time-viewing and channel-switching capabilities as functions for linking with a Toshiba AV device ("REGZA Blu-ray" recorders equipped with a Blu-ray disc drive). Moreover, for the REGZA Phone T-02D released in 2012 summer, we enhanced video quality at the time of viewing and improved device operability such as by enabling continuous viewing of individual drama episodes and providing relay playback between an AV device and the smartphone.

## 5. Conclusion

This paper described Fujitsu's approach to smartphone development using the Android platform, focusing on memory management, power-saving measures, convenience-enhancing manner modes, high-picture-quality technology, and AV-device-linking technology. The appearance of Apple's iPhone and the Android platform has reshaped the saturated feature-phone market as users turn to smartphones in droves. Looking forward, Internet services centered on smartphones and tablets are expected to expand while usage scenarios and lifestyles become increasingly diverse and sophisticated. Nevertheless, there will still be users who have difficulty adjusting to smartphones. There is therefore a need to pursue a level of convenience at least on par with that of feature phones in addition to achieving a highly safe-and-secure design and long battery life. The smartphone application execution environment is already reaching a level no different than that of personal computers of about a decade ago. Given this and the fact that smartphones are now being equipped with all sorts of communication functions and sensors, evaluation techniques and quality indices different from those of feature phones will be needed to enable users to carry around their smartphones all the time.

At Fujitsu, we have been aggressively pursuing improvements to user operability and convenience as reflected by our development of the Raku-Raku (easy-to-use) series of smartphones for NTT DOCOMO and a Human Centric Engine. In future R&D, we will enhance these technologies to further promote the use of smartphones while continuing to develop smartphone platform and evaluation technologies to enhance operation stability and availability.

**Makoto Honda**
*Fujitsu Ltd.*
Mr. Honda is engaged in the development of smartphone software and promotion of open-source software.

**Masahiko Nagumo**
*Fujitsu Ltd.*
Mr. Nagumo is engaged in the design and development of smartphone software frameworks.

**Makoto Kobayashi**
*Fujitsu Ltd.*
Mr. Kobayashi is engaged in the design and development of smartphone software systems.

**Yasuhiro Kawakatsu**
*Fujitsu Ltd.*
Mr. Kawakatsu is engaged in the design and development of smartphone multimedia platforms.