ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

Porting and BSP Customization of Android 4.3 on I.MX6 Based Custom Hardware Platform

Zuberahmed Punekar[1], Harish V Mekali[2], Vivek Kaushik[3]

Student M.Tech [Electronics], Dept of ECE, BMS College of Engineering, Bangalore, India[1]

Assistant professor, Dept of ECE, BMS College of Engineering, Bangalore, India[2]

Project Lead [Embedded Software], TES Electronic Solutions, Bangalore, India[3]

ABSTRACT: Since the introduction of the Android Open Source Project (AOSP) for mobile phones by Google, there has been significant interest in the Original Equipment Manufacturer (OEM) community to also customize Android for other embedded platforms such as Industrial control panels, Human machine interface devices, Security system devices, set-top boxes, car dashboards and others. The advantage of making Android available to multiple device platforms would mean that an application developed for one device could easily be made available for another platform with minimal porting needs. The OS porting and BSP (Board Support Package) customization concept has made OEM community able to meet their strict Time to market requirements. This paper discusses the general steps involved in porting Android 4.3 onto I.MX6 SoC based custom hardware platform.

KEYWORDS: Android Porting, BSP Customization, I.MX6 SoC, Custom Hardware Platform.

I.INTRODUCTION

Today Android operating system (OS) is HOT in market for embedded devices like mobile phones, HMI devices, Industrial control panels and Security system devices. Industry is exploring the ability of Android within other embedded platforms. Some industries replace with exiting operating system with Android because main reason is open source operating system (OS). Today industries select Android OS reason behind this big application market and easy to available to application development tools and also available many on-line group to resolve the development issues.

A Board Support Package (BSP) provides a standardized interface between hardware and the operating system. BSP provide an interface to device drivers which allow the kernel to communicate with the hardware's assets such as device controllers, the microprocessor, memory, internal and external busses. BSP is implementation of specific support code for a given (motherboard) board that conforms to a given OS. BSP is an interface that implements and supports an Embedded OS on Hardware. It supports the major features sets such as storage, networking, display and multimedia. BSP can be quickly customized for the specific need of the customer. With a BSP, one can rapidly bring up an OS on Standard Development Board and evaluate the features of the OS.

The term OS porting means to modify or customize an OS, which is running on particular hardware architecture, in such a way that it can run on another particular kind of architecture when loaded into one. For example, The Linux supporting personal computer mostly use Intel processors, and the architecture of Intel based processors are x86 or x64. But most of the embedded devices use ARM based processors. So if it is required to load the Linux OS into an embedded device, something has to be done that will enable the OS running on x86 based hardware to run on ARM based hardware, and we call it Linux Porting.

Easiest way to create a hardware specific BSP is to use an existing BSP that is designed for similar hardware and develop it to suit custom hardware. The reference BSP so chosen need to be customized so that it supports just those features required for the custom hardware. For example the reference BSP might be supporting both HDMI and LVDS displays but the hardware to which the BSP is being developed need to have only the LVDS display, in such case the

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

reference BSP is customized only to include LVDS display support. Following figure summarizes the process of Porting the BSP Customization

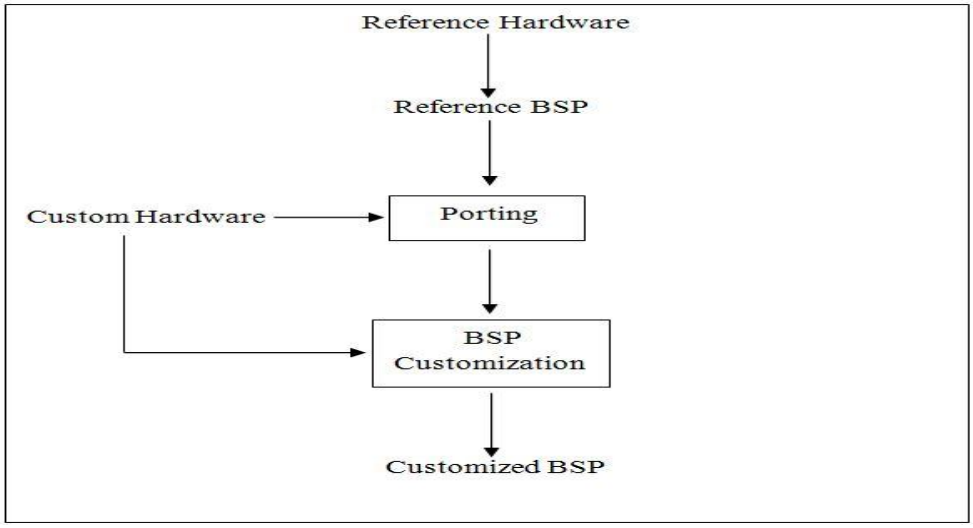

Figure 1 Custom BSP development process.

II.RELATED WORK

In [1] a new kind of vehicle navigation and position terminal system based on 32-bit ARM and GPS has been researched. This paper discusses the development of each part of BSP for WinCE in detail based on the structure program, and analyzes the difficulties in the startup code. The method to write the startup code is described specially.
The [2] tries to address the major challenges in the field of Multiprocessor SOC software design. The software design flows are proposed which overcomes the limitations of typical design flows.
In [3] the hardware components of embedded system for fiscal cash register are introduced. The functions and characteristics of BSP (board support package) in the development of embedded system are discussed.
The [4] discusses the Porting and BSP Customization of Linux on ARM Platform using LTIB development environment.
The [5] discusses the porting and BSP customization of Android on NXP2120 application processor platform.
The [6] discusses the basic concepts involved in porting android to any hardware. It discusses in detail the layered architecture of Android, the layer to which developers gain access and the working of the architecture. It also discusses how Android works on any hardware and the concepts that outline the porting of Android onto any hardware. This paper discusses about the linux kernel used for Android and the Android file system made with Android images.

III.THE I.MX6 BASED CUSTOM HARDWARE PLATFROM

SGET(Standardization Group for Embedded Technologies) is a technical and scientific association[7]. SGET, prepares and compiles technical specifications or standards such as, Implementation guidelines, Software interfaces and System requirements. SGET standards are designed to serve the purpose of Energy efficiency, Environmental protection and Effective technology.

The SMARC ("Smart Mobility ARChitecture") is a versatile small form factor computer Module definition defined by SGET, targeting applications that require low power, low costs, and high performance. The Modules will typically use ARM SOCs similar or the same as those used in many familiar devices such as tablet computers and smart phones. Alternative low power SOCs and CPUs, such as tablet oriented X86 devices and other RISC CPUs may be used as well. The Module power envelope is typically under 6W.

Two Module sizes are defined: 82mm x 50mm and 82mm x 80mm. The Module PCBs have 314 edge fingers that mate with a low profile 314 pin 0.5mm pitch right angle connector (the connector is sometimes identified as a 321 pin connector, but 7 pins are lost to the key). The figure 2(a) shows the block diagram of I.MX6 dual/quad based SMARC module.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

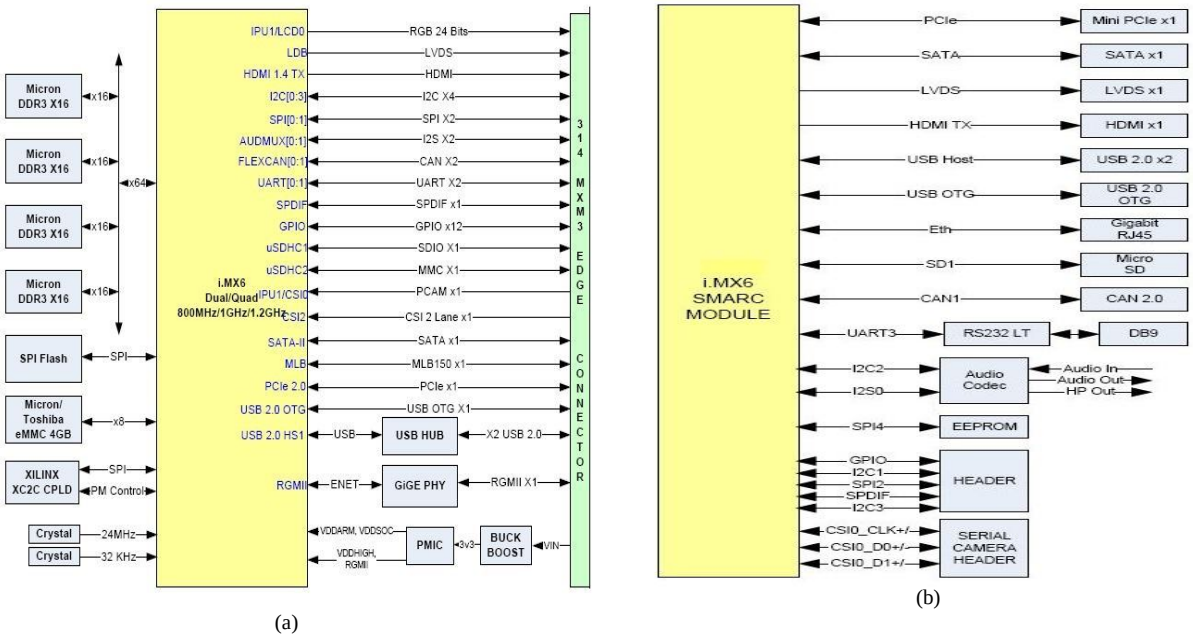

Figure 2 (a) I.MX6 dual/quad based SMARC module (b) Carrier Board with I.MX6 based SMARC Module

The Modules are used as building blocks for portable and stationary embedded systems. The core CPU and support circuits, including DRAM, boot flash, power sequencing, CPU power supplies, GBE and a single channel LVDS display transmitter are concentrated on the Module. The Modules are used with application specific Carrier Boards that implement other features such as audio CODECs, touch controllers, wireless devices, etc. The modular approach allows scalability, fast time to market and upgradability while still maintaining low costs, low power and small physical size. The figure 2(b) shows the carrier board architecture carrying the SMARC module.

IV.INTRODUCITON TO ANDROID

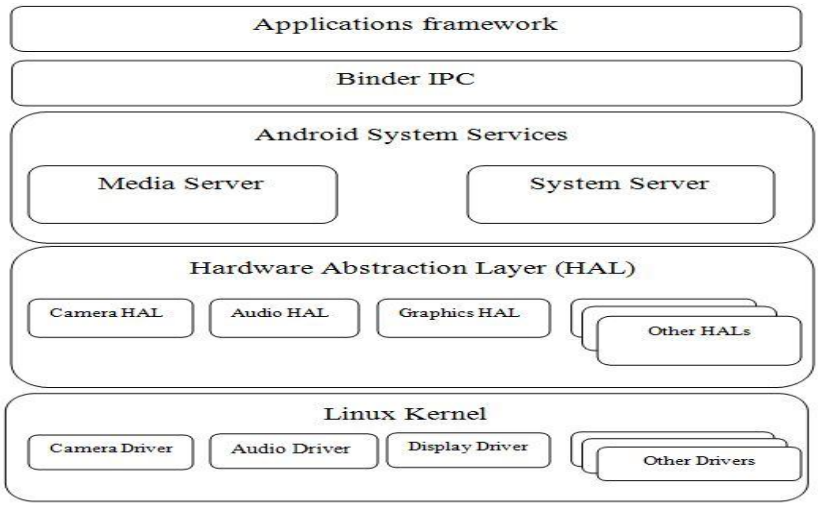

Figure 3 Android low level system architecture

The Figure 3 shows the low level system architecture of the android.

Application framework: Contains the APIs which application developers can use as an interface to the underlying HALs

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

Binder IPC: The Binder Inter-Process Communication mechanism allows the application framework to cross process boundaries and call into the Android system services code.
System services: Most of the functionality exposed through the application framework APIs must communicate with some sort of system service to access the underlying hardware. System services are grouped into two buckets: system and media. The system services include things such as the Window or Notification Manager. The media services include all the services involved in playing and recording media.
Hardware abstraction layer (HAL): The HAL serves as a standard interface that allows the Android system to call into the device driver layer while being agnostic about the lower-level implementations of the drivers and hardware.
Linux Kernel: For the most part, developing device drivers is the same as developing a typical Linux device driver. Android uses a specialized version of the Linux kernel with a few special additions such a memory management system that is more aggressive in preserving memory.

V. PREREQUISITES AND SETUP

To build the Android source files, a Linux host computer is required. Host machine should have Ubuntu 12.04 (32/64 bit) OS version, which is the most tested OS for the Android JB4.3 build. 100 gigabytes of free disk space should be available on Host machine for building images.

The following packages are required to build the android images.

```
$ sudo apt-get install openjdk-7-jdk
$ sudo apt-get install uuid uuid-dev
$ sudo apt-get install zlib1g-dev liblz-dev
$ sudo apt-get install liblzo2-2 liblzo2-dev
$ sudo add-apt-repository ppa:git-core/ppa
$ sudo apt-get update
$ sudo apt-get install git-core curl
```

The Android source code is maintained as more than 100 gits in the Android repository (android.googlesource.com). To get the Android source code from Google repo, follow the steps below:

```
$ cd ~
$ mkdir myandroid
$ mkdir bin
$ cd myandroid
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ ~/bin/repo init -u https://android.googlesource.com/platform/manifest -b android-4.3_r2.1
$ ~/bin/repo sync # this command loads most needed repos. Therefore, it can take several
hours to load.
```

The following commands get the source code for android kernel and boot loader developed for the I.MX6 based Freescale Stadand Development Platforms.
Get jb4.3_1.0.0-ga kernel source code from Freescale open source git:

```
$ cd myandroid
$ git clone git://git.freescale.com/imx/linux-2.6-imx.git kernel_imx # the kernel repo is
heavy. Therefore, this process can take a while.
$ cd kernel_imx
$ git checkout jb4.3_1.0.0-ga
$ cd myandroid/bootable
$ cd bootloader
$ git clone git://git.freescale.com/imx/uboot-imx.git uboot-imx
$ cd uboot-imx
$ git checkout jb4.3_1.0.0-ga
```

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

VI.PORTING AND BSP CUSTOMIZATION

Android consists of a kernel based on the Linux kernel 2.6, with libraries, middleware and application software running on an application frame which contains Java-compatible libraries and APIs written in C [6]. The main hardware platform for Android is the ARM architecture. ARM machine dependent code is placed in the *myandroid/kernel_imx/arch/arm* directory. The contents in the subdirectories of the kernel source tree are defined as follows:

boot: Contains specific booting process code.
kernel: Consists architecture dependent kernel files.
configs: Default configuration for specific platform.
common: Consists common file for the ARM architecture.
mach-*: Consists platform and processor specific files.
mach-mx6: Contains platform and processor specific files for the I.MX6 based application board.

The kernel configuration and build system are based on multiple Makefiles. But we basically interact with the main Makefile, present at the top directory of the kernel source tree. The essential thing to build the Kernel source is *make* utility which is installed onto the host helps to build every file and its dependencies as per the directions in the main Makefile which the relating instructions has written to bind all other Makefiles and dependencies under the hierarchy. The best part of this *make* is to ease building the files that the developer is fascinating in and reducing a lot of time by excluding unemployed files and its dependencies in the course of building.

BSP customization is done by following steps,

1. Go to the Kernel root directory
   $ cd ~/myandroid/kernel_imx
2. Fire the menuconfig graphical utility
   $ make menuconfig
   A GUI as shown in figure 4 below would open

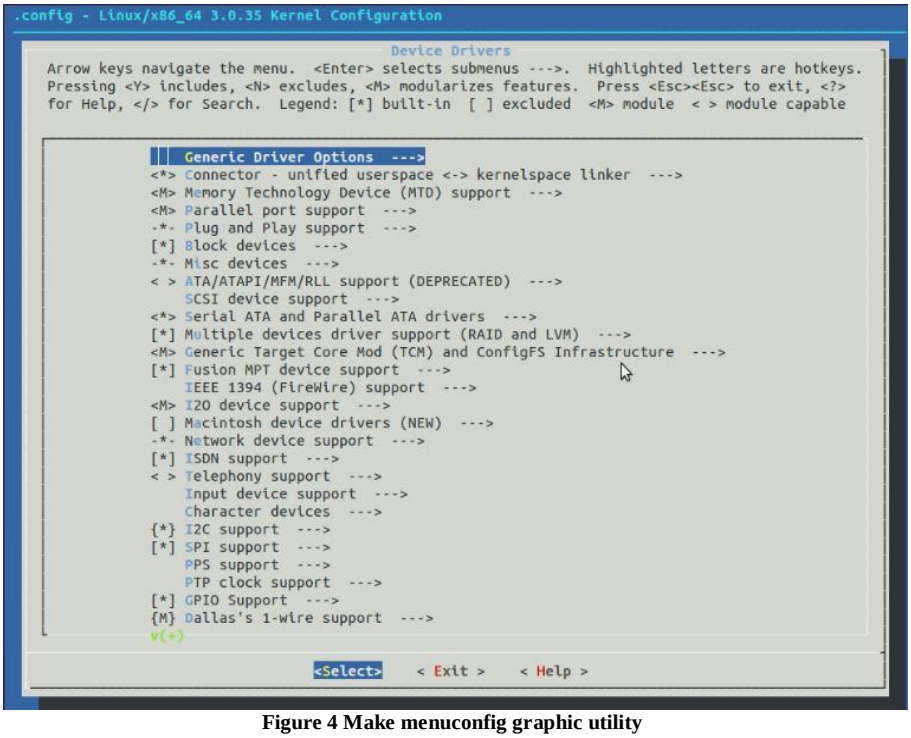

Figure 4 Make menuconfig graphic utility

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

4. Once the customization is done select exit to save the configuration.

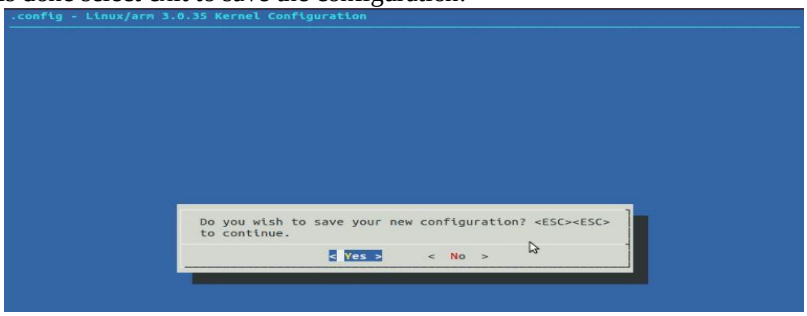

Figure 5 Prompt to save the kernel configuration

5. Configurations will get saved in the .config file in the current directory which is hidden and not accessible for the user.
6. Next time when the kernel is built configurations saved in the .config file are automatically applied.

VII.BUILDING THE ANDROID IMAGES

Android repo provides a universal boot loader U-Boot for different boards at ~/myandroid/bootable/bootloader/uboot-imx/. The boot loader developed fro freescale SABRE-SD development board is based on the I.MX6 quad SoC at ~/myandroid/bootable/bootloader/uboot-imx/board/freescale/mx6q_sabresd suits the needs of all the I.MX6 quad SoC based custom platforms. No to build the boot loader use below commands,

```
$ cd ~/myandroid/bootable/bootloader/uboot-imx
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin/arm-eabi-
$ make distclean
$ make mx6q_sabresd_android_config
$ make
```

The boot loader “u-boot.bin” is generated in the current directory after the successful build. Building the u-boot also generates “mkimage” utility in the path ~/myandroid/bootable/bootloader/uboot-imx/tools which will be used in the generating kernel image.

Assuming U-Boot is already built and mkimage was generated under myandroid/bootable/bootloader/uboot-imx/tools/ run the below commands to build the kernel.

```
$ export PATH=~/myandroid/bootable/bootloader/uboot-imx/tools:$PATH
$ cd ~/myandroid/kernel_imx
$ export ARCH=arm
$ export CROSS_COMPILE=~/myandroid/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin/arm-eabi-
$ make uImage
```

The above command “make uImage” takes the configuration information from “.config” file which is present in the current directory.

Successful build of the Kernel stores uImage in ~/myandroid/kernel_imx/arch/arm/boot/ and message as below would be shown in the figure 6 below.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

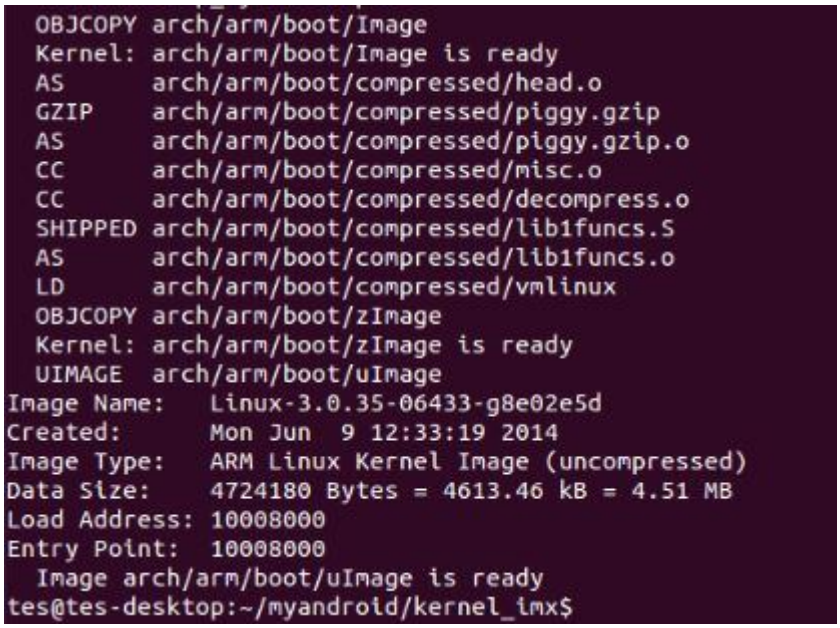

Figure 6 Successful build of the Kernel Image.

VIII FLASHING AND BOOTING

Delete the existing partitions in the SD card and create an ext4 partition for holding the Android file system. Label the partition as "rootfs". Leave 100 MBs of un-allocated space the before partition for holding boot loader and Kernel. All this can be achieved using the "gparted" utility.

Now once the SD card is ready use below commands to flash the Boot loader, Kernel and the Android File system (Obtain the Android file system from the freescale Android 4.3 release for I.MX6)

Flashing the Boot loader

- $ cd ~/myandroid/bootloader/uboot-imx/
- $ sudo dd if= u-boot.bin of=/dev/sdc bs=1k seek=1

Flashing the Kernel

- $ cd ~/myandroid/kernel_imx/arch/arm/boot
- $ sudo dd if=uImage of=/dev/sdc bs=1M seek=1

Note: In the above dd commands device for the flashing is mentioned as /dev/sdc this is because in the current environment the SD card is detected as /dev/sdc.

Flashing the File system

- $ cd *location of the file system tar file*
- $ tar -xvf *name of the file system tar file* -C /media/rootfs/

The SD card is ready to boot on the custom hardware platform.

IX RESULTS

The Figure 7, represents the footprints of the kernel sizes with different BSP configurations, here four generalized configurations like minimum configuration means it is having the basic kernel, it requires low memory space, so less time required to boot the target. Default configuration: in this some of the additional components are added to the minimum configuration like USB, NFS, network, etc. so the time taken to boot the target device is more compared to minimum configuration, because of driver modules need some time to load and initialize the target device. Maximum Configurations: it allows the all functionalities and device drivers are added the OS, so along with the footprint size and boot time is also increased. Finally Customized configuration: Here all the usage of resources are limited i.e the selection for the responsibilities of the kernel is as per the requirement of the target application, sometimes additional

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

drivers re also patched o the kernel depends on the requirement, so this paper proposed Customized OS size is 4.5MB. The final size of the kernel and the file system is 42.29MB, so target device boot time is also very less.

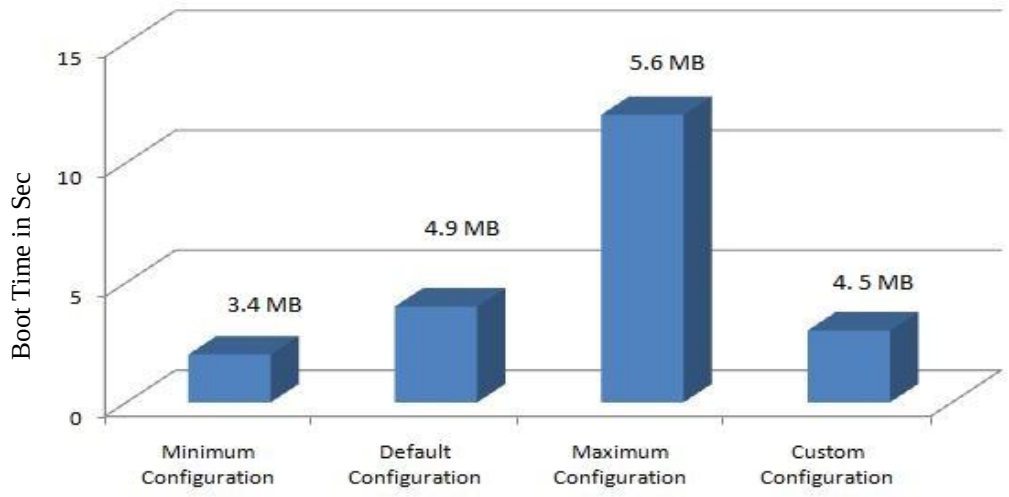

Figure 7 Kernel Footprints Vs Boot time with different BSP configurations.

X CONCLUSIONS

We examined the importance of Android OS, specialties and functionalities in embedded systems. According to the fact the Linux OS and Android OS are practically one and the same. Not totally the same, however the Android kernel is straightforwardly inferred from the Linux. The Android 4.3 is ported on the I.MX6 based custom hardware platform built according to the SGET standards. Porting is carried out in four main steps configuring, building, flashing and finally booting the target platform. Finally the Kernel footprint size and boot time of the device is evaluated for the various BSP configurations.

REFERENCES

[1] Ji Wankang, Xiamen Yang Jia and Hong Yongqiang "BSP Development of WinCE System for Vehicle Navigation Device Based on S3C2440". Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference.
[2] Haid, W, Kai Huang , Bacivarov, I. and Thiele, L "Multiprocessor SoC software design flows". Signal Processing Magazine, IEEE (Volume:26 , Issue: 6) Nov 2009.
[3] An-kun He, Ping-Su, Zhuang-Wu, "BSP Design for Fiscal Cash Register Embedded System". Computer Engineering and Technology, 2009. ICCET '09. International Conference.
[4] K. Esware kumar, M Kamaraju and Ashok kumar yadav, "Porting and BSP Customization of Linux on ARM Platform" International Journal of Computer Applications (0975 – 8887) Volume 80 – No 15, October 2013
[5] V. Hari Prasad, Ashok Kumar Yadav and K Radha, "Porting of Android OS BSP Customization on NXP2120 Application Processor Platform". International Journal of VLSI and Embedded Systems, Vol 04, Article 08146; September 2013
[6] Shankar A, Lal S, "Android porting concepts", Electronics Computer Technology (ICECT), 2011 3rd International Conference on (Volume:5), 8-10 April 2011.
[7] www.sget.org
[6] The i.MX 6 series BSP developers guide at www.freescale.com
[8] Hughes Systique. (2009, March 16). Android Porting Guide for Embedded Platforms. (2nd edition). [Online]. Available: www.hsc.com/.../Android-Porting- on-Embedded-Platform-v2-0633850602027036930.pdf