

Writing Driver for the Low Cost Power Management IC

Rahul A. Pipaliya¹ Arjav A.Bavarva² Shankar Patel³

^{1, 2, 3} VLSI & Embedded Systems Design Department

^{1, 2, 3} Gujarat Technological University, Ahmedabad, India

Abstract— In this paper, the topic attempts to clarify that in existing system from Texas Instrument product having AM3359 processor connected with I2C to highly integrated PMIC TPS65950 having cost of \$4.4/1ku. This device provides the sufficient voltages to the various modules like Wi-Fi 802.11 needs 3.3V and 1.8V, NAND flash needs 3.3V, DDR3 needs 1.5V, 2.5inch HDD needs +5V, USB needs 5V. It has also fuel gauge IC called bq27510 which continuously monitors the power of battery and state the charging/Discharging in percentage. The main agenda of this topic is to replace this PMIC with PIC microcontroller having the same function and cost of \$2.8(1-4 piece). So the overall cost can be reduced by \$4.4-\$2.8=\$1.6. For this new device the new driver will have been written on Linux platform. For this new low cost product driver called power, regulator, I2C, IRQ have been written and tested on new hardware.

Keywords: Low Cost PMIC, power management, AM3359, Device driver.

I. INTRODUCTION

The title clarifies that the existing product from Texas Instruments having Power management IC TPS65950 which is having very price \$4.4/1ku also fuel Gauge device bq27510. TPS65950 has three efficient converters which help to distribute the sufficient voltages to the other parts of the board having VDD1, VDD2, VDAC, VPLL1 signal ID. This device also connects to AM3359 processor with I2C bus interface. This bus helps to read and write data from TPS65950 to AM3359 and vice versa. The fuel gauge IC bq27510 estimates how long battery is going to charge and discharge. It will be in form of percentage that shows battery charging and discharging rate in percentage.

This topic shows how to replace the existing PMIC with new cost effective device called PIC16F1789 having price of \$2.8(1-4 piece). So compare this price to existing PMIC TPS65950 then it will save around \$4.4-\$2.8=\$1.6 per new product. If it will be applied for large amount of production then the more benefits will be achieved. PIC16F1789 having 40 Pin IC has to be function of providing sufficient voltages to the different modules of the board. Like Wi-Fi 802.11 needs 3.3V and 1.8V, NAND flash needs 3.3V, DDR3 needs 1.5V, 2.5inch HDD needs +5V, USB needs 5V, all these values of voltage is provided by PIC16F1789 IC.

This project is all about to write the driver in Linux platform for the cost effective product having PIC16F1789 (regulator driver), Fuel gauge bq27510 (Power driver), I2C driver (Core) for interfacing and data acquisition between AM3359 processor and PIC and IRQ driver for interrupt. This all four driver will be based on LINUX platform. So the overall task is to reduce cost of the new product with efficient power management on Linux platform.

For Writing Driver in OS like Linux needs knowledge in some of the important topics like modules, Loadable modules, kernel, userspace, kernel space, scheduler, System call, data transfer mechanism, File system in Linux, Types of Drivers, Virtual file system (VFS). Different layers in Kernel Space.

Basic task is to be performed with writing module, insertion process of module in kernel, study about the different kernel source code, file system, Cross compilation process on customized kit.

This topic is divided into two parts for the testing purpose (1) FRONT END (Software part having write drivers i.e. power, regulator, Core (I2C), IRQ compilation process on Linux based system) (2) Hardware verification for cross compilation process with AM335x board and run APIs to get proper voltages.

II. BLOCK DIAGRAM

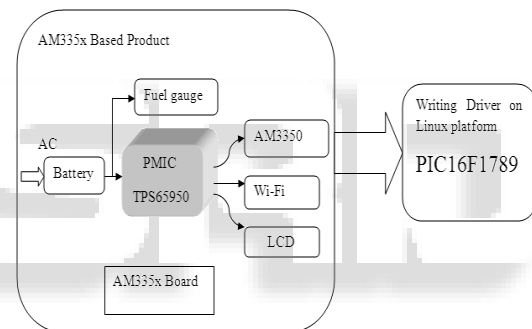


Fig. 1: Block Diagram

Above figure illustrates that diagram at the left side contains PMIC TPS65950 interfacing with AM3350 application processor and fuel gauge IC bq27510 with battery with AC supply which shows the existing mechanism. The existing highly integrated IC TPS65950 with efficient bulk DC to DC converters is distributing the voltage e.g. 3.3V, 1.8V, 5V to the other modules of the board. This also clarifies that the application processor is having interface between this PMIC is with I2C bus on which data transfer is possible. The TPS65950 block diagram has the different sub chips including interface sub chip, power sub chip, Audio sub chip, USB subchip, Auxiliary Subchip. Here is the main focus on the power sub chip having two parts analog and digital power. Digital power has smart reflex, PMC master, PMC slave, Analog part has power divider (LDOs-DC to DC), thermal monitoring system, RC oscillator, RTC, power references. Above Figure also shows new product with PIC16F1789 with the same function having the driver in Linux environment for performing the same voltage range and same power management as the TPS65950 had.

VI. PROJECT DRIVER IMPLEMENTATION

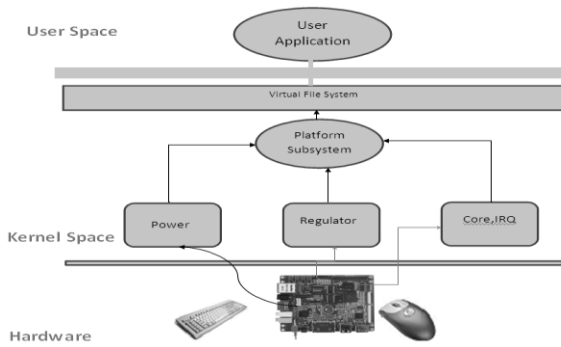


Fig. 5: Driver Implementation

Figure shows the basic implementation of Driver in this project. In this diagram, I2C subsystem has the connection with rest of the all driver and subsystem. Each of the subsystem has the driver, this will help to communicate any of the data transfer to the user space and hardware level. All of them creates interface to communicate between the user space and kernel space[9].

There is very thick boundary between this two level i.e. User space and kernel space shown in figure. No one can enter from the user space to kernel space we have some interface which helps to data transfer from user space to kernel space. System call APIs read, write, open etc. All are used to build up the setup for this two level.

For writing any of the character drivers the following

points are to be considered.

- Define the functions used in writing driver.
- Fill up the structure in file operation.
- Register the driver.

VII. FUEL GAUGE BQ27510

Texas Instruments bq27510 system-side Li-Ion fuel gauge is a microcontroller peripheral that fuel gauging for single-cell Li-Ion battery. The device requires little system firmware development. The bq27510 on the system's main board and manages an battery non-removable or a removable pack. The bq27510 must use a Semitec 103AT NTC thermist or for temperature measurement, or can also be to use its internal temperature sensor.

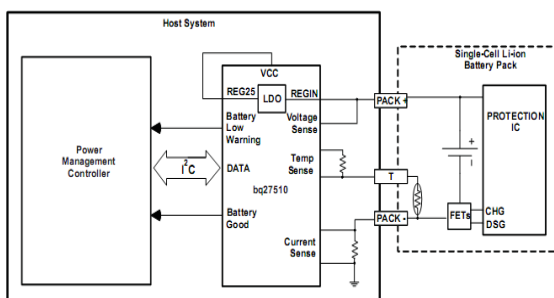


Fig. 6: Fuel Gauge Internal Diagram

To minimize power consumption, the bq27510 has several power modes: NORMAL, SLEEP, HIBERNATE, and BAT INSERT CHECK. The bq27510 passes

automatically between these modes, depending upon the occurrence of specific events, though a system processor can initiate some of these modes directly.

A. I2C Compatible Timing Diagram

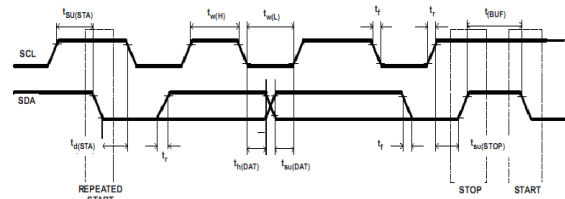


Fig. 7: I2C Compatible Diagram

The key to the bq27510 has high accuracy gas gauging prediction is Texas Instrument's proprietary Impedance Track algorithm. This algorithm uses cell measurements, characteristics, and properties to create predictions that can achieve less than 1% error across a wide variety of operating conditions and over the lifetime of the battery.

The bq27510 measures charge/discharge activity by monitoring the voltage across a small-value series sense (5 mΩ to 20 mΩ typ.) located between the system's Vss and the battery's PACK terminal. When a cell is attached to the bq27510, cell impedance is computed, based on cell current, cell open-circuit voltage (OCV), and cell voltage under loading conditions.

B. I2C Registers

- (1) Status Register #1 (Read only)
- (2) Status Register #2 (Read only)
- (3) Command/Status Register #1 (Read/Modify/Write) Default 00h
- (4) Command/Status Register #2 (Read/Modify/Write) Default 00h.
- (5) Power Supply Status Register (Read/Modify/Write) Default 00h.
- (6) Power Source Status Register (Read/Modify/Write) Default 00h.

VIII. RESULT AND DISCUSSION

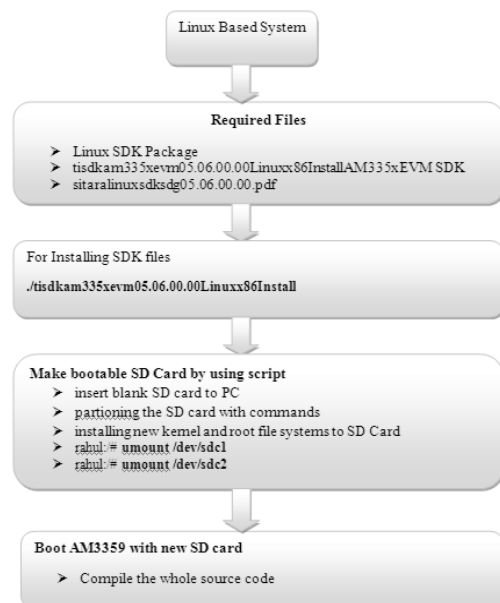


Fig. 8: Testing flow for the project

Project have been divided into two parts

- Front End
- Back end

Front End part taken out of having the basic of Linux environment, driver information, how to write a driver, how to load a module into kernel, data transfer mechanism, read the different kernel source code, study about the power,regulator,I2C driver and its header file, file system mechanism, Read different registers of I2C.

Back End part is totally base on hardware platform which enhanced to power up the performance of the project and get the final output. Driver testing on the hardware kit also include this part. This part also contains the voltage outputs with the special APIs which are being carried out.

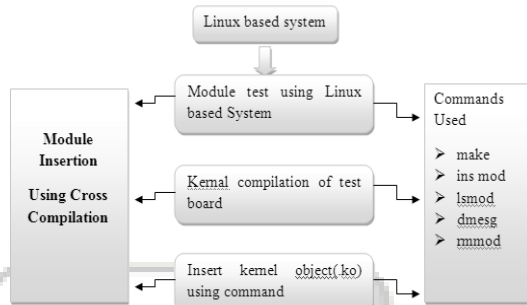


Fig. 9: Module Compilation Process

A. How to load module into kernel

Step1: Write any module in high language in C called module.c

Step:2 Compile it by make file.It creates .ko(kernel object)file.

Step:3 For insert into kernel type command sudo insmod.

Step:4 To show this module in kernel type demsg command.

B. Cross compilation process of any module using kit.

Write Driver in high level language like C having the name module.c

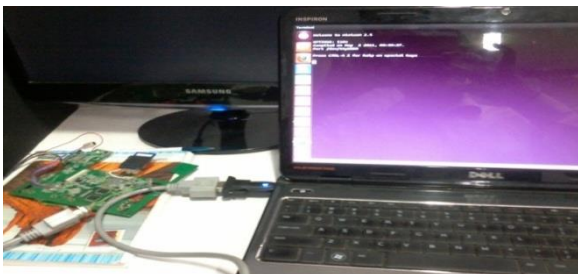


Fig. 10: Cross compilation process of any module using kit.

To enter into kernel type command sudo insmode.



Fig. 11: Cross compilation process of any module using kit.

For viewing this the init function arguments type command dmesg

Here I have used Am335x EV for the cross compilation. For the cross compilation I have folled the following steps:

- Set path for the new hardware
- Run minicom by using command sudo minicom.

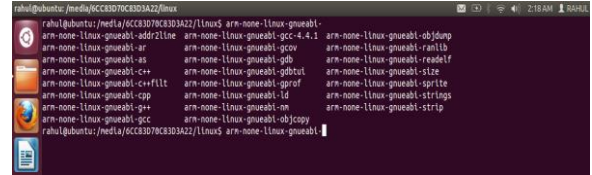


Fig. 12: Cross compilation process of any module using kit.

After setting the path, Switch on Power button and the booting process is on. The name given if figure “Angstrom” indicates that booting process is over.



Fig. 13: Cross compilation process of any module using kit.

After loading the kernel,For remove the module need the command rmmode.



Fig. 14: Cross compilation process of any module using kit.

After writing all the drivers it would be put into the following folders so that we can compile whole kernel and find errors if any.



Fig. 15: Cross compilation process of any module using kit.

Here are the some step for kernel compilation

Step: 1 Installing sdk environmnet of AM335x

- Get the new version in one folder.

Step: 2 Execute the following command to grant yourself full access to the installer

- sudo chmod a+x ti-sdk-am335x-evm-05.06.00.00-Linux-x86-Install

Step: 3 Now start the installer being sure to use "sudo" access

- sudo ./ti-sdk-am335x-evm-05.06.00.00-Linux-x86-Install

Step: 4 Add tool chain in path

- export PATH=\$PATH:/opt/tools/arm-2010q1/bin/

Step: 5 For Kernel compilation

- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean

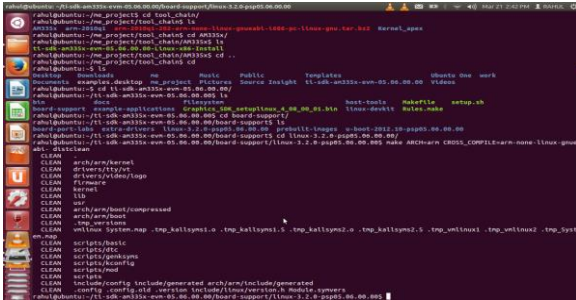


Fig. 16: kernel compilation

Step: 6 Make Menu Config

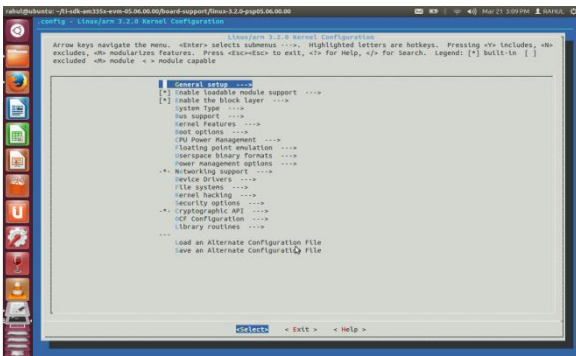


Fig. 17: kernel compilation

Step: 7 After configure the menu and enabling the driver, go for the uImage generated command

- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage

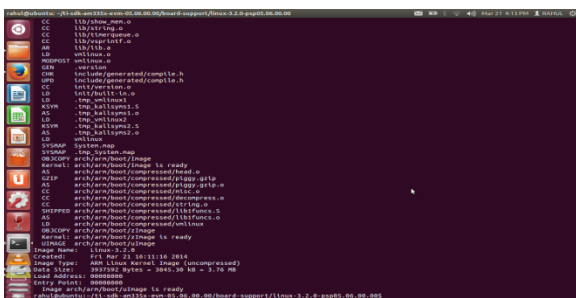


Fig. 18: kernel compilation

Above process will taken it to the uImage generated on the path

Arch/arm/boot/uImage

- uImage
- uboot
- x-loader(MLO)

Step: 8 Take three of the above file into Micro SD card.

IX. CONCLUSIONS

After getting the depth knowledge of Linux kernel ,Linux file system ,u boot ,x-loader and hardware interfacing and device driver. The drivers for this dissertation topic of PIC module i.e. Power, Core/IRQ, regulator is working properly as uImage, MLO, uboot.bin files generated and verification of all these drivers is still going on AM335x Cortex A8 EVM. Practice of Cross compilation of such module on Sitara AM335x ARM Cortex A8 is successfully done.The overall cost may be reduced by \$1.6 per product is one the biggest achievement of this project.

As divided two part of this dissertation topic i.e. FRONT END and BACK END. So the FRONT END part is over and full concentration to the BACK END part for the more accurate result which would on hardware base i.e. Sitara AM335x Cortex A8 EVM from Texas Instruments.

REFERENCES

PAPERS

- [1] J. Monteiro, S. Devadas, P. Ashar, and A. Mauskar, "Scheduling techniques to enable power management," in Proceedings of the 33rd Design Automation Conference, pp. 349–352, Las Vegas, Nev, USA, 1996.
- [2] Mario Menninger "Power Management for Portable Devices" Communications Business Unit Austria Microsystems AG Unterpremstaetten, Austria, 2007.
- [3] Mu-Kai Huang ,J. Morris Chang, Wei-Mei Chen "Grouping-Based Dynamic Power Management for Multi-Threaded Programs in Chip-Multiprocessors", International Conference on Computational Science and Engineering, 2009.
- [4] Personal communication with Prof. Na Kong, T. Shaver Deyerle IV, and Dong Sam Ha, Department of Electrical and Computer Engineering Virginia Tech Blacksburg, VA 24061 USA " Universal Power Management IC for Small-Scale Energy Harvesting with Adaptive Impedance Matching" , 2011.
- [5] S. Vaddagiri, A. K. Santhanam, V. Sukthakar, and M. Iyer, "Power management in linuxbased systems," Linux Journal, 2004.

WEBSITES

- [1] http://www.ti.com/lsds/ti/analog/powermanagement/power_portal.page, Aug 2013.
- [2] <http://www.ti.com/lsds/ti/omap-applicationprocessors/overview.page#performance>, Sept 2013.

BOOKS

- [1] Embedded Linux primer, 2nd Edition A Practical real world approach, Christopher Hallinan, Prentice hall open source software development series. Sept 2006.
- [2] Understanding of Linux kernel, 3rd Edition, Daniel P. Bovet, Marco cesati, O'Reilly, Nov 2005.
- [3] Linux Device Drivers, 3rd Edition ,Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman, O'Reilly media, February 2005.