Software Profiling For Holistic System's Power And Performance Analysis
(Modem Statistics)

Prepared By

Kamlesh Karwande

12MCEC13

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2014

Software Profiling For Holistic System's Power And Performance Analysis
(Modem Statistics)

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

Prepared By

Kamlesh Karwande

12MCEC13

External Guide

Mr.Ashish K Singh.

Intel Technologies India Ltd.

Internal Guide

Prof. Ankit Thakkar.

Nirma University.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2014

Certificate

This is to certify that the Major Project Report entitled **"Software Profiling For Holistic System's Power And Performance Analysis "** submitted by **Kamlesh A Karwande. (Roll No: 12MCEC13)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Mr.Ashish K Singh.
Software Engineer,
Intel India,
Bangalore.

Prof. Ankit Thakkar.
Assistant Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Vijay Ukani.
Associate Professor,
Coordinator M-Tech CSE,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sanjay Garg.
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. K Kotecha.
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Undertaking for Originality of the Work

I, **Kamlesh Karwande**, Roll. No. **12MCEC13**, give undertaking that the Major Project entitled **"Software Profiling For Holistic System's Power And Performance Analysis "** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed by

Mr. Ashish K Singh

(Signature of External Guide)

Prof. Ankit Thakkar

(Signature of Internal Guide)

Acknowledgements

First and foremost, I would sincerely like to thank **Mr.Diganta Roychowdhury**, Manager, Intel Technology India Private Limited, Bangalore.My mentors **Mr.Ashish K Singh and Mrs.Shruti Chandna**. I enjoyed their vast knowledge and owe them lots of gratitude for having a profound impact on this report.

I would like to thank my Mentor, **Mr. Ashish K Singh**, Intel Technology India Private Limited, Bangalore for his valuable guidance. Throughout the training, he has given me much valuable advice on project work. Without him, this project work would never have been completed.

I would also like to thank my Internal guide **Prof. Ankit Thakkar**, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance.

My deepest thank you is extended to **Prof. Vijay Ukani**, PG CSE - Coordinator, Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout the Major Project.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Honorable Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

I would also like to thank **Dr. K.Kotecha**, Director, Institute of Technology, Nirma University, Ahmedabad for providing me an opportunity to get internship at Intel Technology India Private Limited, Bangalore.

I would like to thank my all faculty members for providing encouragement, exchanging knowledge during my post-graduate program.

I also owe my colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.

Kamlesh Karwande

(12MCEC13)

Abstract

Intel Corporation designs and develops Smart-phones based on latest technology. These smart-phones combine the functionality of a mobile communication device with miniature computer like capabilities. It integrates various features like voice communication, image capturing, Internet browsing, audio video playback, message communication and many more. The more the functionality added, the more complex the underlying system, ultimately the Hardware. This in turn takes its toll on power supply, the battery. Thus to increase the battery life the need of efficient energy management rises. This leads us to the ultimate goal, an energy efficient platform.

For this we require assistance from tools which will help in understanding the working of the platform from power as well as performance perspective. Developing such tool will greatly help in understanding and optimizing Intel's mobile platforms.

Modem is a large consumer of power, and hence optimizing the modem utilization can increase the overall battery life of a smart-phone. As a part of this project we further increase the capability of the tool by integrating Modem power statistics. The mechanism is built on custom AT command interfaces.

Getting the modem statistics for various use cases would help in understanding and optimize the modem utilization hence ensuring an overall improvement in the battery-life without compromising on the performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today mobile phones are not only used just as a device for calling and receiving but transformed into a smart-phone[6], with more advanced computing capability and connectivity than a regular cellular phone[6]. The smart-phones includes media players[12], mail functionality, video cameras, high pixel digital cameras GPS6[12] navigation units, etc. to form one multi-use device. Modern smart-phones also include high-resolution touchscreen[11] and web browsers that display standard web pages as well as mobile-optimized sites. High-speed data access is provided by Wi-Fi[6], mobile broadband and Bluetooth[6]. With the increase in the use of each feature in the mobile device power consumption increase and performance becomes important.

The embedded systems and various computing devices such as smart-phones, tablets, portable handheld web devices, ebook readers are often used for prolonged periods without being connected to a power outlet. This usage pattern poses a new set of challenges related to power and performance. These systems comprise of various modules such as: Camera, Accelerometer, GPS, Orientation sensors, temperature sensors, Radio and Modem chips, CPUs, GPUs and bright LCDs. Also there are very large variations in the specifications of such devices with varied hardware configuration including processor, Wi-Fi chips and other modules, although the use cases are similar. The challenge is to balance the performance and power equation so that the device can be used for an acceptable duration within acceptable performance bounds. Performance without power considerations is meaningless, especially in the smart-phone world so energy efficiency is important.

1.1 Problem Definition

Derive methodology for capturing holistic platform[1][5][11] metrics for analyzing use cases for power & performance optimization[9][10][4]. Currently there is no methodology to capture modem power data via software. The only possible way is to connect huge Power data collecting devices such as NI-DAQ's[13] or Monsoon Power Monitors[14]. But again the limitation is, such devices cannot be connected to Form Factors[11], the final customer model. More over the data collected using such power data capturing devices is not precisely allied to the use case running on the device. Human operator has to take care of simultaneous launching of the use case and the recording device.

To address the basis of the issue, a proposal was made to develop a mechanism to capture Modem[6][1]5 related metrics focusing of power and performance use-cases[10].

- The approach has to be robust enough so that the technique can be accommodated into existing PnP analysis tools[9] without much integration overhead.
- The technique must have minimum footprint on the Modem functionality in terms of performance.
- Robust implementation to accommodate changes on both Modem as well as Application processor in future.

1.2 Motivation

Smart-phones, tablets, portable hand-held devices are used for prolonged periods without being connected to a power supply. This usage pattern poses challenges related to power and performance. The challenge is to balance the performance and power so that the device can be used for an acceptable duration within acceptable performance.Here end goal is to achieve highest performance with lowest power consumption, so it's of utmost for smart-phones to be Energy Efficiency. A very high performance without power consideration is meaningless in case of portable devices. It is the energy which needs to be efficient.

Keeping the above points in mind the key goal of getting the Tool to capture Modem Statistics is to get statistical information about the modem that would help us understand the modem power consumption in finer granularity for a given system workload like

browsing/streaming etc. To achieve the goal our primary focus is to get information about two things

- How are resources (for example, core and other IP blocks) getting utilized within modem.
- How are network conditions affecting modem power.

We want the abstraction to be in fairly high level so that the parameters can be understood by a generalist debugging system level problems without the prerequisite of thorough understanding of the protocol stack.

The statistic collection should happen in the following two modes from the user perspective,

1. Cumulative information from the modem for the duration of the entire workload, for example - % of time processor is in deep sleep, sleep or active during browsing over 3G[6]. The Tool sends a message at the beginning of the workload and asks modem side to start the statistics collection. At the end of the workload, the Tool sends another message, receiving which the modem stops the collection, collates the results and sends it out to the Tool for further post-processing.
2. Time-Interval information about the modem states, for example for roughly every 100 milli-sec time buckets, % of time the processor is in deep sleep compared to idle. Essentially, we are interested in drawing a time-plot of the modem feature variations. Here the Tool would initiate a session with the modem and modem would periodically push back the statistics for us.

1.3 Scope of the Work

- Intel SoC Overview.
- Understand Power and Performance features, Measurements and analysis at a platform level on Low Power UMG Platforms.
- Study, understand and use the Tool.
- Study and understand Modem Firmware.
- Study and understand AT Command Structure.

- Study the Remote Procedure Call (RPC) mechanism.
- Integrate Modem statistics capturing capability to the Tool.

The main task is to develop a mechanism to capture modem statistics and integrate them into the tool.

1.4 Organization of Project Report

Following this introduction, chapter 2 presents an overview of the Intel UMG Platform, followed by introduction to Power and Performance Measurements of UMG Platforms its application and requirements.
Chapter 3 presents a literature survey for understanding Modem and its interface with the platform. The two approaches AT command interface and RPC mechanism are also understood.
Chapter 4 explains the Tool's architecture in detail, where the actual implementation should be accommodated for modem data collection.
Actual Modem statistics collection implementation for both AT command based and RPC based and their respective integration is explained in Chapter 5.

The thesis ends with discussing the conclusions derived from the work comparing both the approaches and their pros and cons.

Chapter 2

Intel Mobile Platform Architecture

Intel Corporation[1][11] is unbeaten for silicon manufacturing in the world. For years it has been at the pinnacle for manufacturing state of the art CPU's for Desktop and Server segment. Its latest addition is Atom processor[1][11].

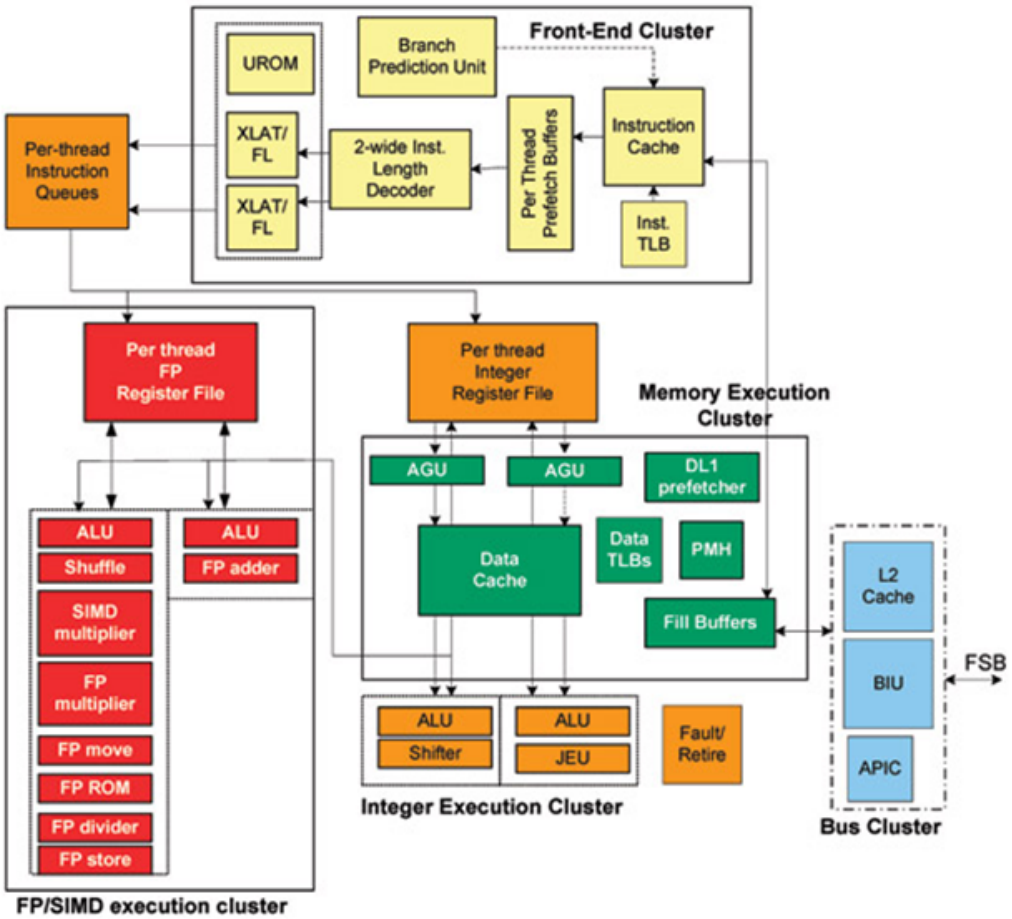


Figure 2.1: Block Diagram of Atom Processor (Source Internet)

Intel Atom is the brand name for a line of ultra-low-voltage Intel x86 and Intel x64 CPUs from Intel. Intel Atom is family of ultra-low power microprocessors specifically designed for Mobile Internet Devices and ultra mobile PCs. It has implemented many techniques to reduce power consumption.

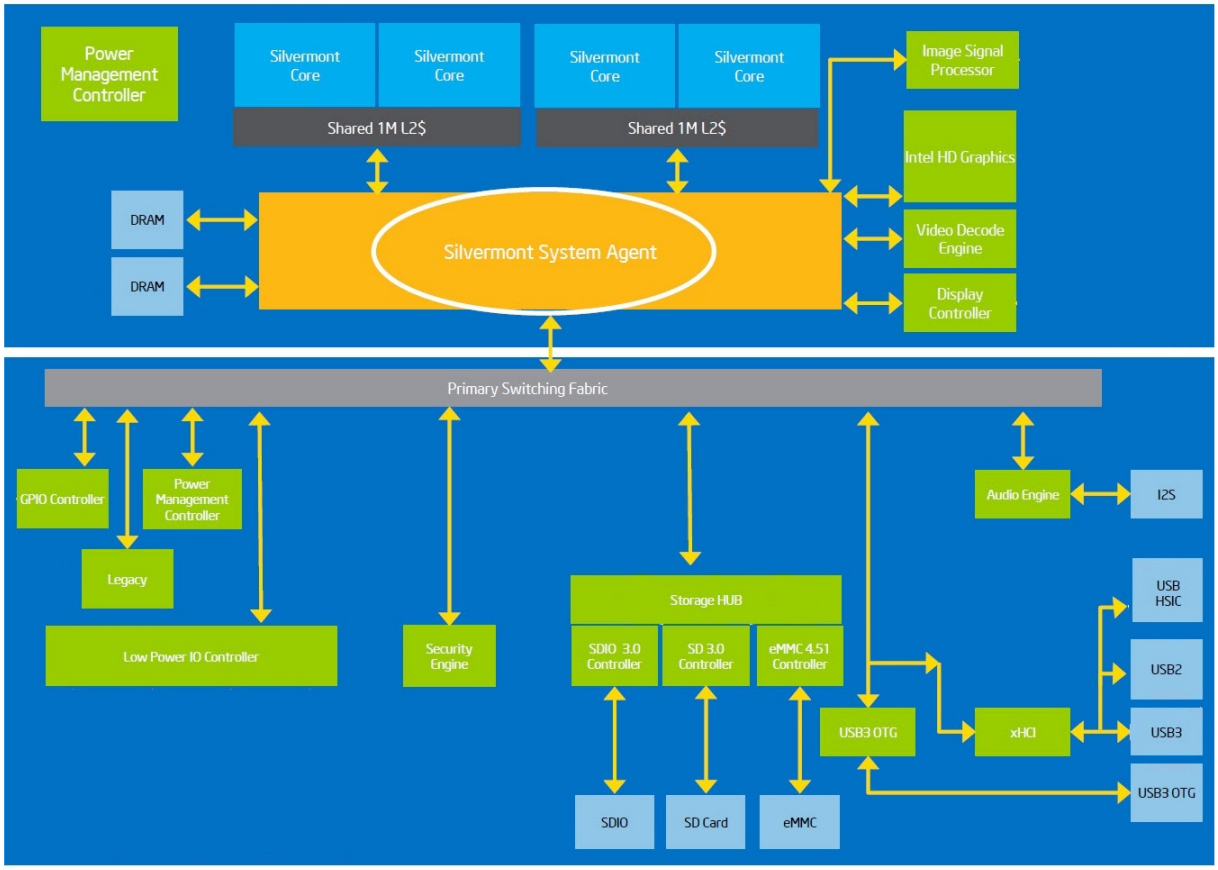


Figure 2.2: Block Diagram of Intel Atom SoC (Source Internet)

Atom processor family has been expanded to new SoC[1] (System on Chip) platform targeting smart phones and tablets. This platform consists of all the IC's required by the system. following are the major components of a platform.

Hardware

- CPU
- GPU
- North complex
- South Complex
- Buses

- Memory
- Motherboard

Software

- Operating System
- Firmware
- Applications such as media player, web browser.

Around early 2012, Intel started expanding the Atom processor family; its existing low power consuming flagship processor series with a new system on chip (SoC) platform designed specifically for low power consuming devices like tablets and smart-phones. The tablets will be powered by Microsoft Windows 8 OS as well as Android OS[12].Atom competes with companies like Texas instruments[13], Qualcomm, Nvidia, Samsung who are currently developing SOCs for smart-phone and tablet market. But all the above companies, use CPU's based on ARM architecture which are designed from the beginning to consume very low power. The x86 (32 bit) based Atom processor line developed by Intel for low power usage in net-books, has been adapted to even lower power usage.As of 2014, Intel Atom-based tablets and phones have been released by several manufacturers.

2.1 UMG Platforms

Ultra Mobility Platforms[1] are designed for a specific segment of ultra-mobile devices namely Ultra mobile PCs (UMPC), Mobile Internet Devices (MID), Smart-phone's. All of these devices are hand held devices in nature, so power requirements of these devices must be very low and should have acceptable performance while running the workloads. Power saving techniques is extensively used in all these devices.

Total power consumed by the platform and its performance varies significantly with the workload as different hardware components and software stacks are utilized differently for various workload. Optimizing the complete hardware and software stacks for the workloads is very important for better performance and power utilization. Hence Power and Performance measurements of UMG platforms become an important activity in analyzing and optimizing the platforms.

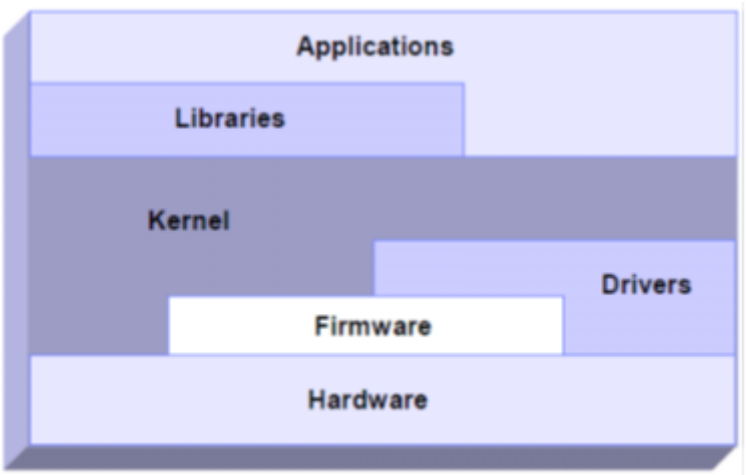


Figure 2.3: Schematic Representation of various Components of Platform[11]

2.1.1 Power measurements of UMG Platforms

A UMG platform consists of many components like CPU, Chipset, Memory, Display,Modem,Audio etc. These components behave differently and consume different amount of power for various workloads and understanding the impact on various platform component because of workloads it's necessary and power measurement is an important tool for that.

Usually DAQs (data acquisition systems) are being used for power measurements. The basic concept is very simple and as follows

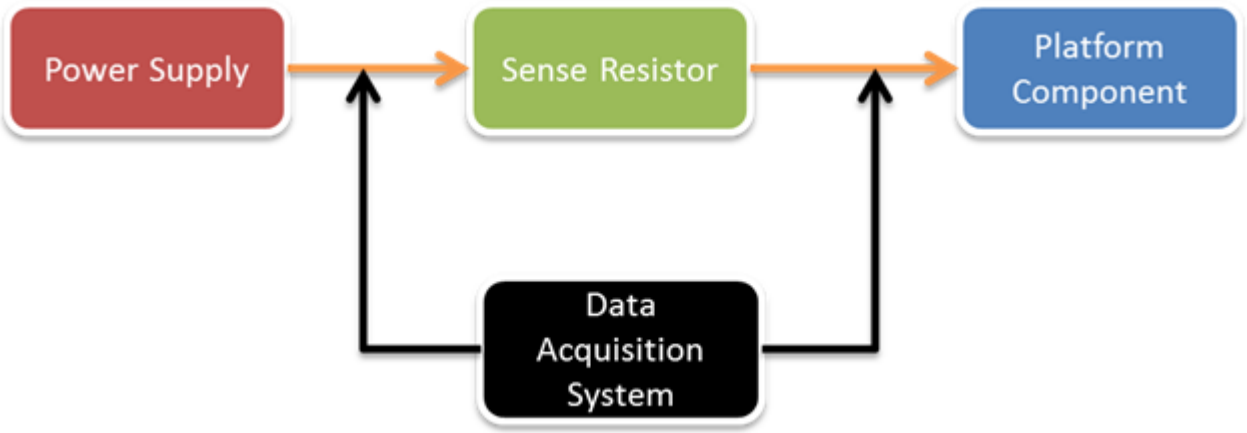


Figure 2.4: System setup for power measurements

- The differential voltage across the sense register is sampled at regular interval and is divided with the value of the sense register to get the current through the rail.

- Computed current is multiplied with absolute rail voltage to get the instantaneous power of the component and average is taken over all the samples for a specific period.

2.1.2 Performance Measurements of UMG Platforms

It is important to be able to assess the performance estimates of a platform. Platform designers use performance estimates to evaluate the effectiveness of a new features. Usually a standard set of application programs are grouped together known as benchmarks. Benchmarks are run on the platforms which give a score which is used for comparisons between different platforms across the industry. Some bench marks are just to compare the performance of CPUs. Other benchmarks related to graphics are 3D Mark, 3D Mobile mark, EEMBC, Dhrystone etc.

There are other tools also to measure the performance for various workloads. The operating system also monitors the performance of the system for various controlling applications.

Tool	**Most Important Tool Function**
top	Process Activity
vmstat	System Activity, Hardware and System Information
uptime, w	Average System Load
ps, pstree	Display the Processes
free	Memory usage
iostat	Average CPU load,Disk Activity
sar	Collect and Report System Activity
mpstat	Multiprocessor usage
numasat	NUMA Related Statistics
pmap	Process memory usage
netstat	Network Statistics
iptraf	Real-Time Network Statistics
tcpdump, etheral	Detailed Network Traffic Analysis
nmon	Collect and Report System Activity
strace	System Calls
proc file system	Various Kernel Statistics
KDE system guard	Real-Time System Reporting and Graphing
Gnome System Monitor	Real-Time System Reporting and Graphing

Table 2.1: Common Tools used in Linux Environment (Source Internet)

2.2 ACPI Standards

The Advanced Configuration and Power Interface (ACPI)[15] specification was particularly developed to create industry common interfaces empowering strong operating system (OS)-directed motherboard device configuration and power management of both entire systems and individual devices. ACPI is the key component in Operating System-guided setup and Power Management (OSPM)[15] [1].

ACPI advanced the current pre-ACPI collection of power management by BIOS code, Advanced Power Management (APM) application programming interfaces (APIs, PNPBIOS APIs, Multiprocessor Specification (MPS) tables and so on into a well-defined power management and configuration interface specification. ACPI gives the intends to an organized move from existing (legacy) equipment to ACPI fittings, and it takes into consideration both ACPI and legacy systems to exist in a machine and to be utilized as required.

The interfaces and OSPM ideas characterized are suitable to all classes of machines including (but not limited to) desktop, mobile, workstation, and server machines. From a power management viewpoint, OSPM/ACPI pushes the idea that the systems should conserve energy by transitioning unused devices into lower power states including placing the entire system in a low-power state (sleeping state[15]) when possible.

2.2.1 Principal Goals

ACPI is the key component in executing OSPM. ACPI-characterized interfaces are proposed for wide adoption to encourage hardware and software vendors to build ACPI-compatible (and, thus, OSPM-compatible) implementations. The principal goals of ACPI and OSPM are to:

1. Enable all computer systems to implement motherboard configuration and power management functions, using appropriate cost/function trade offs.

2. Enhance power management functionality and robustness.

3. Facilitate and accelerate industry-wide implementation of power management.

4. Create a robust interface for configuring motherboard devices.

2.2.2 ACPI States

Platforms compliant with the ACPI specification give OSPM immediate and exclusive control over the power management and motherboard device configuration functions of a computer. Throughout OS initialization, OSPM assumes control over these functions from legacy implementations , for example, the APM BIOS, legacy requisitions. OSPM is answerable for handling of motherboard device configuration events and in addition for controlling the power, thermal status and performance of the system based around client inclination, application requests and OS forced Quality of Service (QOS)[15][11] usability objectives.

ACPI provides low-level interfaces that allow OSPM to perform these functions. The various states by the ACPI specification are:

Global System State

ACPI defines mechanisms for putting the computer as a whole in and out of system sleeping states. It also provides a general mechanism for any device to wake the computer.

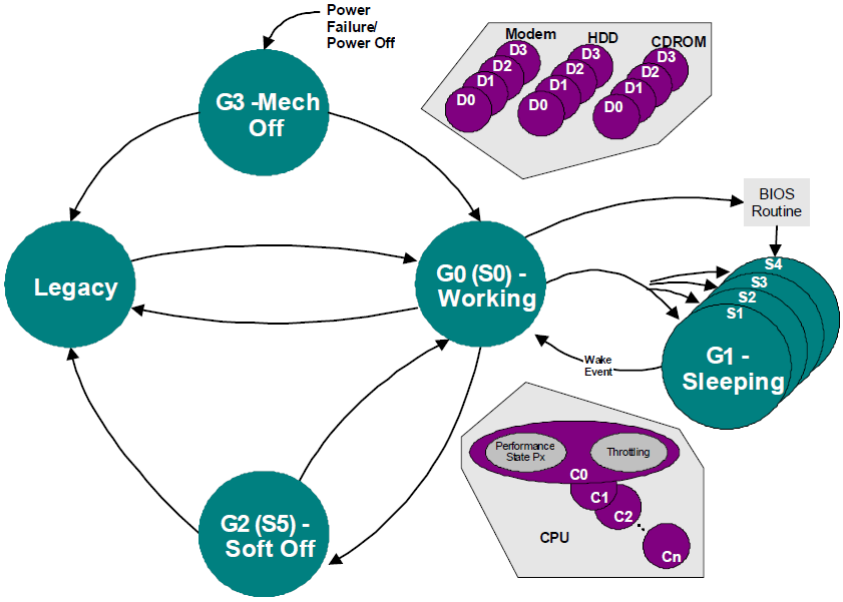


Figure 2.5: Global System Power States and Transitions [15]

The Global System States[15] are:

- **G3 Mechanical Off** :- A computer state that is entered and left by a mechanical

means (external switch).

- **G2 / S5 Soft Off** :- In Computers state G2 S5 the computer consumes extremely low power. No system or user level code is running. The hardware does not maintain or preserves the system content.

- **G1 Sleeping** :- A computer state where the computer consumes a small amount of power, large elements of system context are saved by the hardware and the rest by system software.

- **G0 Working** :- A computer state where the system dispatches user mode (application) threads and they execute. The system responds to external events in real time.

Device Power State

Device power states[15] are states of individual devices; as such, they are generally not visible to the user. For example, some devices may be in the off state even though the system as a whole is in the Working state. To unify nomenclature and provide consistent behavior across devices, standard definitions are used for the power states of devices.

- **D3 off** :- Power has been fully removed from the device. The device context is lost when this state is entered, so the OS software will reinitialize the device when powering it back on.

- **D3 hot** :- Devices in the D3 hot State are required to be software enumerable. D3hot is expected to save more power and optionally preserve device context. The D3 hot state differs from the D3 off state in two distinct parameters; the main power rail is present and software can access a device in D3hot.

- **D2** :- D2 is expected to save more power and preserve less device context than D1 or D0.Buses in D2 may cause the device to lose some context. Many device classes may not define D2.

- **D1** :- D1 is expected to save less power and preserve more device context than D2. Many device classes may not define D1.

- **D0 fully ON** :- This state is assumed to be the highest level of power consumption. The device is completely active and responsive, and is expected to remember all relevant contexts continuously.

Sleeping State

When the computer is idle or the user has pressed the power button, the OS will put the computer into one of the sleeping (Sx) states[15]. No user-visible computation occurs in a sleeping state. A system in one of these states is not performing any computational tasks and appears to be off. The sleeping sub-states differ in what events can arouse the system to a Working state, and how long this takes.

- **S1 Sleeping State** :- The S1 sleeping state is a low wake latency sleeping state. In this state, no system context is lost (CPU or chip set) and hardware maintains all system contexts.

- **S2 Sleeping State** :- This state is similar to the S1 sleeping state except that the CPU and system cache context is lost the OS is responsible for maintaining the caches and CPU context.

- **S3 Sleeping State** :- In S3 sleeping state complete system contexts are lost except the system memory. Chip set, cacheand CPU context are lost in this state. Its the hardware's responsibility to maintains memory contents restore back some L2 and CPU context.

- **S4 Sleeping State** :- The longest wake latency and lowest power consuming sleeping state supported by ACPI is the S4 sleeping state. Here it is assumed that all the devices are powered off by the hardware platfom manager. But the harware thake care of the device tate by maintaning device context.

Processor Power State

Processor power states (Cx states)[15] are processor power consumption and thermal management states within the global working state, G0. The Cx states possess specific entry and exit semantics and are briefly defined below.

- **C0 Processor Power State** :- While the processor is in this state, it executes instructions. The Processor is in execution mode.

- **C1 Processor Power State** :- This processor power state has the lowest latency.Aside from putting the processor in a non-executing power state, this state has no other software-visible effects. The Processor is in ready state.

- **C2 Processor Power State** :- The C2 state offers more better power savings on the C1 Processor power state. The ACPI system firmware is responsible for maintaining acceptable hardware latency for the state. Based on the latency OS determines the transition of the processor between C1 and C2 states.

- **C3 Processor Power State** :- The C3 state offers much more and better power savings over other higher power states. The ACPI system firmware is responsible for maintaining acceptable hardware latency for the state. While in the C3 state, the processor's caches maintain state but ignore any snoops. The operating software is responsible for ensuring that the caches maintain coherency.

Device and Processor Performance State

Device and Processor performance states (Px states)[15] are power consumption and capability states within the active/executing states, C0 for processors and D0 for devices. The Px states are briefly defined below.

- **P0 Performance State** :-While a device or processor is in this state, it uses its maximum performance capability and may consume maximum power.

- **P1 Performance State** :- In this performance power state, the performance capability of a device or processor is limited below its maximum and consumes less than maximum power.

- **Pn Performance State** :- In this performance state, the performance capability of a device or processor is at its minimum level and consumes minimal power while remaining in an active state. State n is a maximum number and is processor or device dependent. Processors and devices may define support for an arbitrary number of performance states not to exceed 16.

2.3 Power Measument tools

To measure the power consumed by the different IP units[1][11] or components of the platform, external dedicated hardware is required. These power measurement devices are

standalone hardware devices.

2.3.1 Monsoon Power Monitor

Power Monitor[14] is a Power capturing device from Monsoon Solutions Inc. It is a hardware which comes with software residing on a windows machine, which is used to control output voltage and trigger power capture. The Mobile Device Power Monitor hardware and the Power Tool software provide a robust power measurement solution for Android and Window based devices.

The Mobile Device Power Monitor hardware and the Power Tool software can analyze the power on any device that uses a single lithium (Li) battery. Software Developers and Electrical engineers can utilize the Mobile Device Power Monitor hardware and the Power Tool software to optimize the design and analyze the performance of the devices, thus helping in optimization from power prespective.

Figure 2.6: Monsoon Power Monitor (Source Internet)

Principle of the Power Monitor

To measure the current exactly, a double range, self-calibrating, integrating system is utilized. Each one channel has two current ranges with a 16-bit simple analog to digital converter (ADC), unified with a high-resolution range, and the other with a low-resolution range. Via Programming each of these channels are calibrated continuously

and chooses the correct range throughout measurement. The double range plan works in light of the fact that mobile platforms are normally in standby mode and drawing only a few of milliamps of current, or they are running over 100 milliamps. The Mobile Device Power Monitor must be exceptionally precise when the current is low, yet may be less exact as the current increments. Each sample is integrated over its 200-microsecond sample period so that even a concise high-current pulse is captured.

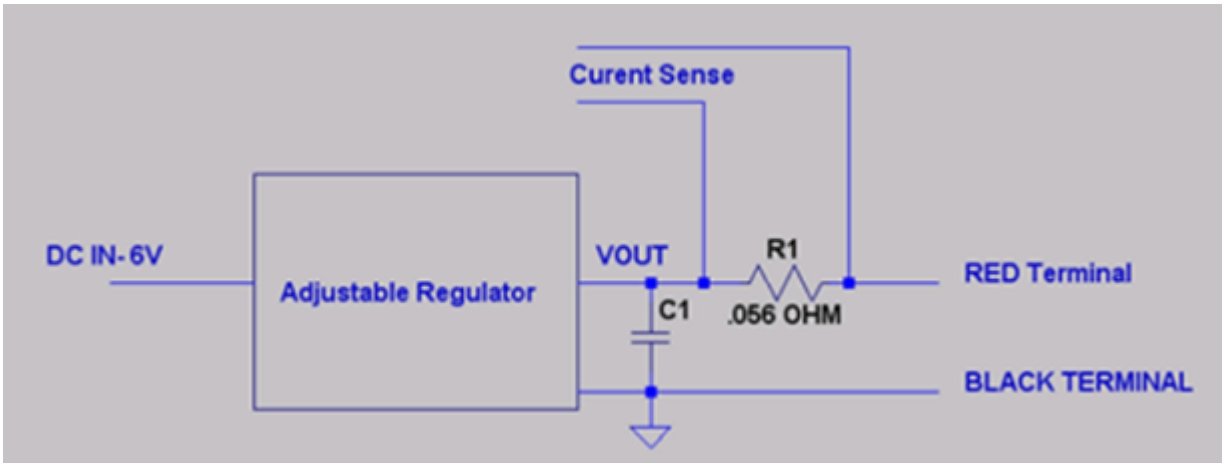


Figure 2.7: Principle of the Power Monitor [14]

Contingent upon system capacitance and other factors, the quickest transient pulse are about the duration of 20 microseconds. The integrator sums up these quick pulses so that a precise evaluation of the normal current is kept up. The unit is self-calibrating. One cycle out of each 1000 cycles is utilized to run either a reference-current adjustment or a zero-current alignment. Via Programming these estimations can be used to null out the offset and gain errors during the system measurement. Since this is carried out automatically, it makes up for slow temperature progressions throughout the estimation run. The only part excluded in the measurement process is the sense resistor. These resistors are aligned at the industrial facility and the adjustment qualities may be balanced and spared by users utilizing the Parameters dialog box. The Power Monitor has a overflow buffer that can hold six packets of 128 bytes each. Throughout the transfer, if the development workstation can't read the information quick enough then this buffers will start to fill up, and samples may be lost. The accepted information is connected so the line seems consistent. To record for the lost examples, the convention holds a 16-bit

counter that tracks the number of lost samples. Since the dropped samples are short of what one percent of a run, the error is minimal. Typically this loss of data is short of what 0.1 percent. In any case, in some cases on long runs or on stacked frameworks the dropped sample check can surpass 65,000.

A number of dropped connections is kept up in the UI. Both of these counts are reset to zero when a run is restarted. These tallies are helpful to figure out how stable the connection is. On the off chance that the dropped packets are more than one percent of all samples, or if there are more than one or two dropped connections for every hour, then it is likely that the workstation has different user programs running that are creating unwanted delays . Shutting these programs or setting up the Power Tool programming on a clean machine ought to explain these issues and decrease the dropped samples and packets.

Power Monitor User Interface

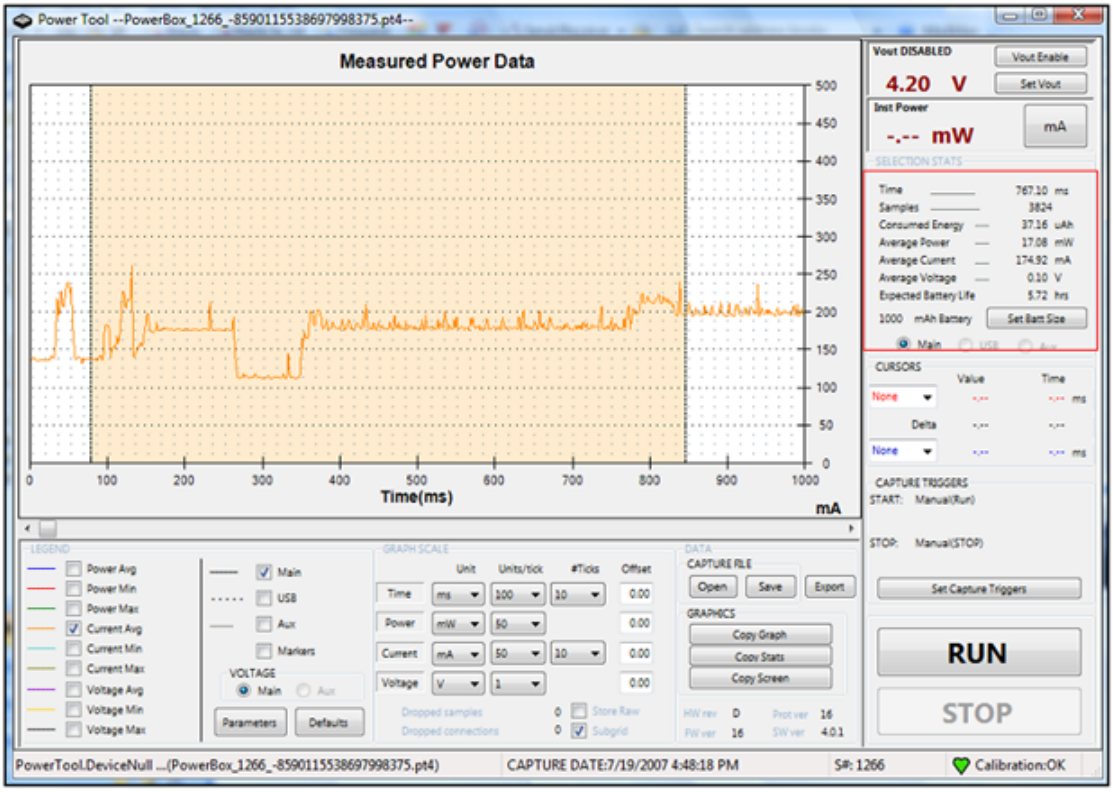


Figure 2.8: Power Monitor User Interface

The User Interface is very simple to understand. We can set the output voltage to any customized value. Graphically it can display Average, Minimum, and Maximum as

well as instantaneous Power, Current and Voltage against time. The parameters of the graph can be varied. Also settings can be made on sampling rate, start capture delay, end time, end samples and so on.

Captured file can be saved in a compressed .PT4 format or .CSV format which can be replayed when needed to understand the platform behavior. In short, the GUI is a very informative and self-sufficient but has one drawback. It requires human input all the time, hence is not very suitable for automation.

The Power Monitor software provides a less informative Command line interface, though not very descriptive it achieves an important purpose, Automation.

Power Monitor Connection Block Diagram

The Power Monitor Connects to the Windows Machine through an USB Interface. The drivers are provided by Monsoon Solutions. The Red and the Black terminals are connected to the Intel Mobile Platform using simple crocodile clip connectors. The Power Monitor hence, not only acts as a power supply/battery to the mobile platform but also capable of monitoring input battery power consumed. Hence it provides a complete solution to monitor platform power controlled through software.

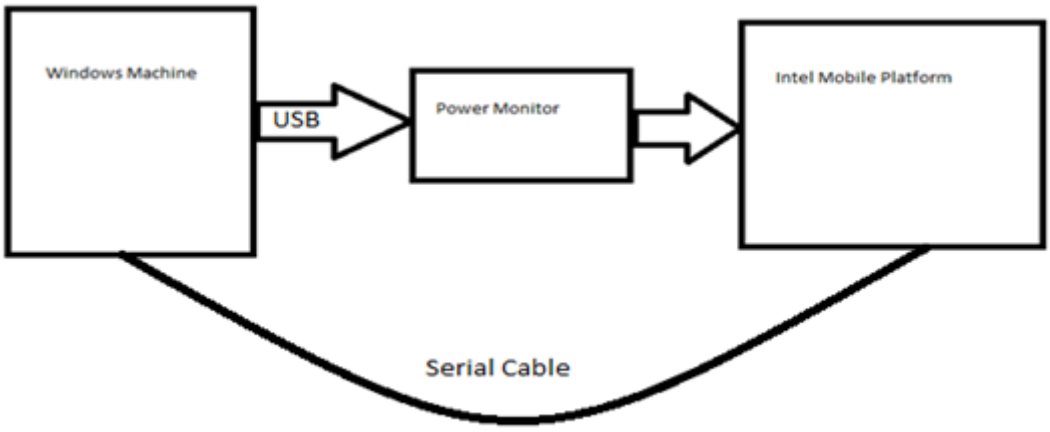


Figure 2.9: Power Monitor & Mobile Device Setup

Intel Mobile platform interfaces with the Windows machine through serial cable. A protocol is defined through which we differentiate power monitor commands. Using this we control the power to the platform.

The operation will be as follows:

Host (Windows Machine) connects to Power Monitor through USB. Power Monitor connects to the target platform by means of connecting wires. The mobile platform connects to Host through a serial cable, thus closing the loop.

1. Whenever Mobile platform requests for a reboot, it will send a request to the Host through the serial cable. Host in turn will acknowledge it by issuing a turn off command to the Power Monitor. After a delay of 1 second, it will issue a turn on command to the Power Monitor thus rebooting the Platform.

2. Whenever a Workload starts, Mobile platform will issue command to the Host for power capture. It will in turn instruct Power Monitor to capture power.

3. Heart beats are simple messages sent to the host to signify that the mobile platform is working fine. In case of any hangs, they will stop. After a delay of 10 seconds, an automatic reboot is issued so that the platform gets up and working. This feature allows no human to be present for monitoring failures.

2.3.2 National Instruments Data Acquisition System (NI DAQ)

National Instruments DAQ[13] is a device with capability to provide a high-output power supply and a structural design optimized in the best possible ways for maximum usability in a wide range of electronic applications. Though it can be used for a wide range of applications which include Voltage, Current, Resistance, Temperature, Sound Pressure, Strain etc. our primary focus is power capturing. In the NI DAQ we use for

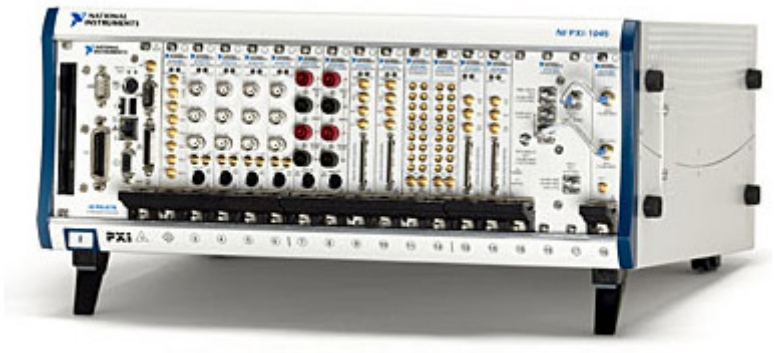


Figure 2.10: National Instruments DAQ

capturing power is PXI 1045. The PXI-1045 combines a high-performance 18-slot PXI backplane. The chassis modular design ensures easy insertion of the PXI slots provide easy replacement and maintanence, which drectly resulting in a very low mean time to repair (MTTR).

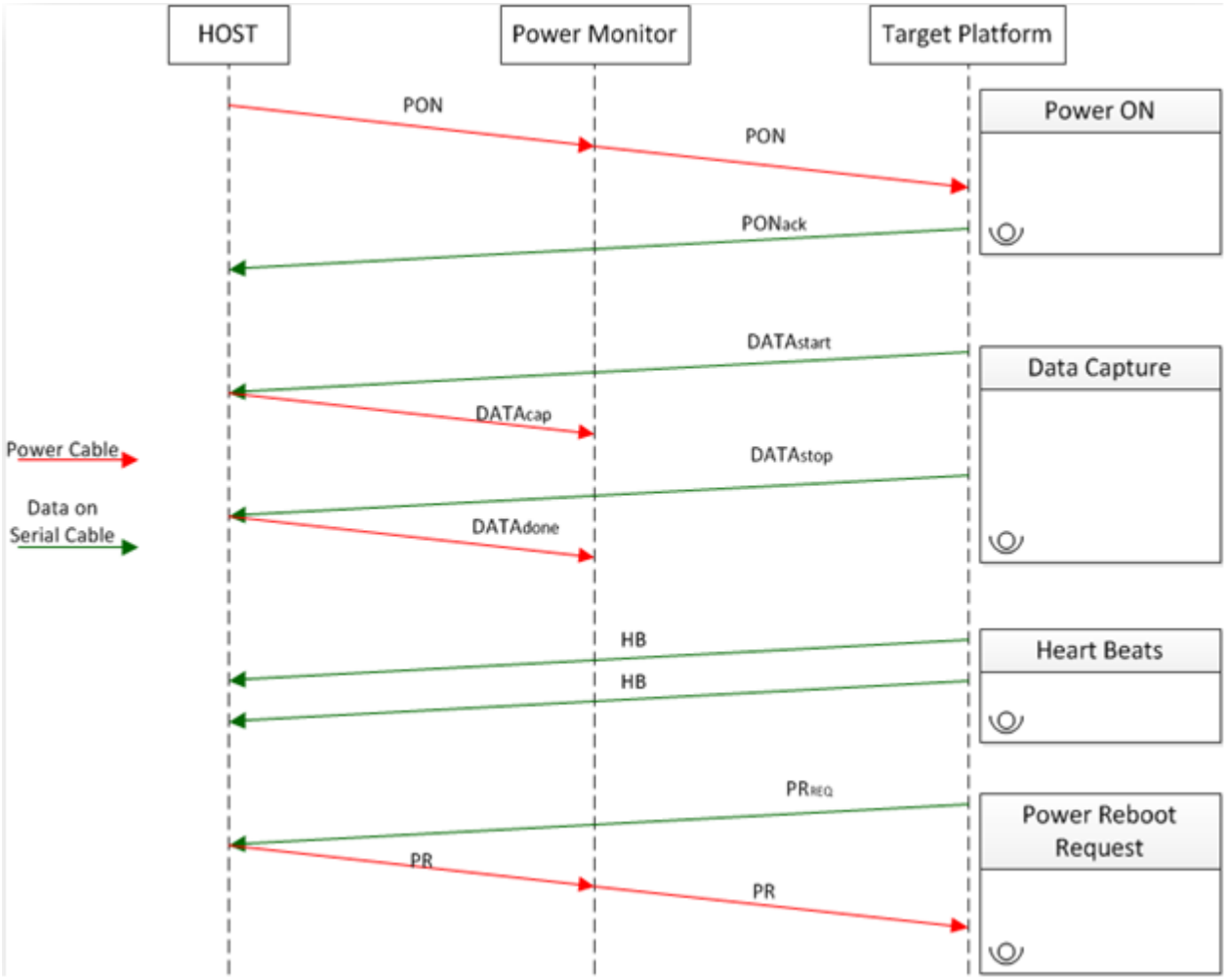


Figure 2.11: Power Monitor Communication Protocol

PXI 6289 Data Acquisition Cards

National Instruments M Series high-accuracy multifunction data acquisition (DAQ) devices are optimized for 18-bit analog input accuracy. This resolution is equivalent to 5 12 digits for DC measurements. To ensure accuracy, the NI-PGIA 2 amplifier technology is optimized for low noise and fast settling to 18 bits and the onboard low pass filter rejects high-frequency noise and prevents aliasing. M Series devices are ideal for applications including test, control, and design. All high-accuracy devices have a minimum

of 16 analog inputs, 24 digital I/O lines, seven programmable input ranges, analog and digital triggering, and two counter/timers.

Figure 2.12: Data Acquisition Cards

The main feature important in selecting 6289 is its 18 bit resolution and 500 KSamples/sec sampling capability. In a Mobile Platform where even spurious wakes can contribute to a milli watt of power, it is very essential that we sample the channels at a very high rate and accurately. The channels we monitor in Mobile platforms can go very low in range of micro watts. In such case, 16 bit resolution is not sufficient. Hence, only a 18 bit resolution can give us more accuracy.

National Instruments LabVIEW

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a platform and development environment for a visual programming language from National Instruments. The purpose of such programming is automating the usage of processing and measuring equipment in any laboratory setup. LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various versions of UNIX, Linux, and Mac OS X.

LabVIEW ties the creation of user interfaces (called front panels) into the development cycle. LabVIEW programs/subroutines are called virtual instruments (VIs). Each

VI has three components: a block diagram, a front panel and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. Controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. However, the front panel can also serve as a programmatic interface. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows non-programmers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and the documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for high-quality G programming. For complex algorithms or large-scale code, it is important that the programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the possibility of building stand-alone applications.

LabVIEW is software designed to cater to all areas. Our area, limited to power capture does not need all the features. In fact, we end up using only 2% of all the features available! Hence paying the license fees for this seems very impractical. Also NI provides free DAQmx drivers and APIs in C, which can be used to create our own software capable of performing like LabVIEW. Hence the need for the project. By doing this, we will achieve,

1. Cost effective software capable to performing as LabVIEW.

2. Tweak the software as per needs to be integrated with other Languages.

3. Distribute it freely within Intel Corporation having to face no licensing issues.

Chapter 3

Literature Survey

3.1 Modem Fundamentals

Modem[?] is one of the most important part of mobile phones. It plays a very crucial role in voice communication, short message communication and all the network related activities on a mobile phone. More over smart phones allow high end functionality such as web browsing, email communication media download and many more. All these functionalities at the basic require usage of Modem.

Modem Model variant that comes with Merrifield[?] platform is XMM7160[?] which is continued with Baytrail[?] also. XMM 7160 is smallest multi-mode / multi-band 2G/3G/LTE slim modem.

Modem is independent of the platform its integrated on. It has its own processor called Baseband processor, memory,decode unit, DSP power management unit. It is connected to the platform via UART channel. Modem Firmware controls the modem activity and modems communication with the application processor.

3.1.1 XMM7160 Modem

The XMM 7160 HSPA+/LTE Modem Platform includes several platform variants targeted for a wide product range like M2M modules, MIDs, eBook readers, data card modems, as well as a broad range of medium to high-end smart phones.

The X-GOLD 716 GSM/UMTS/GPRS/EDGE/HSPA+/LTE Baseband Controller is the basis for the platform family. It is an integrated, low-power, 40 nm baseband device that includes all required digital, analog and power management functions.The platform

supports up to nine LTE bands (plus sub-bands), five UMTS/HSPA+ bands and four frequency bands for EDGE standards, respectively. These ingredients enable the new solution to offer industrys leading edge PCB (printed circuit board) footprint. Intel Mobile Communications complements the XMM 7160 HSPA+/LTE Modem Platform with its own worldwide proven triple-mode 3GPP Protocol Stack in its Release 9 version, which enables to provide complete in-house system solutions.

Smart phones are the main application where the XMM 7160 platform is connected to an application processor. A system block diagram depicting the XMM 7160 platform in a smart phone application is shown below.

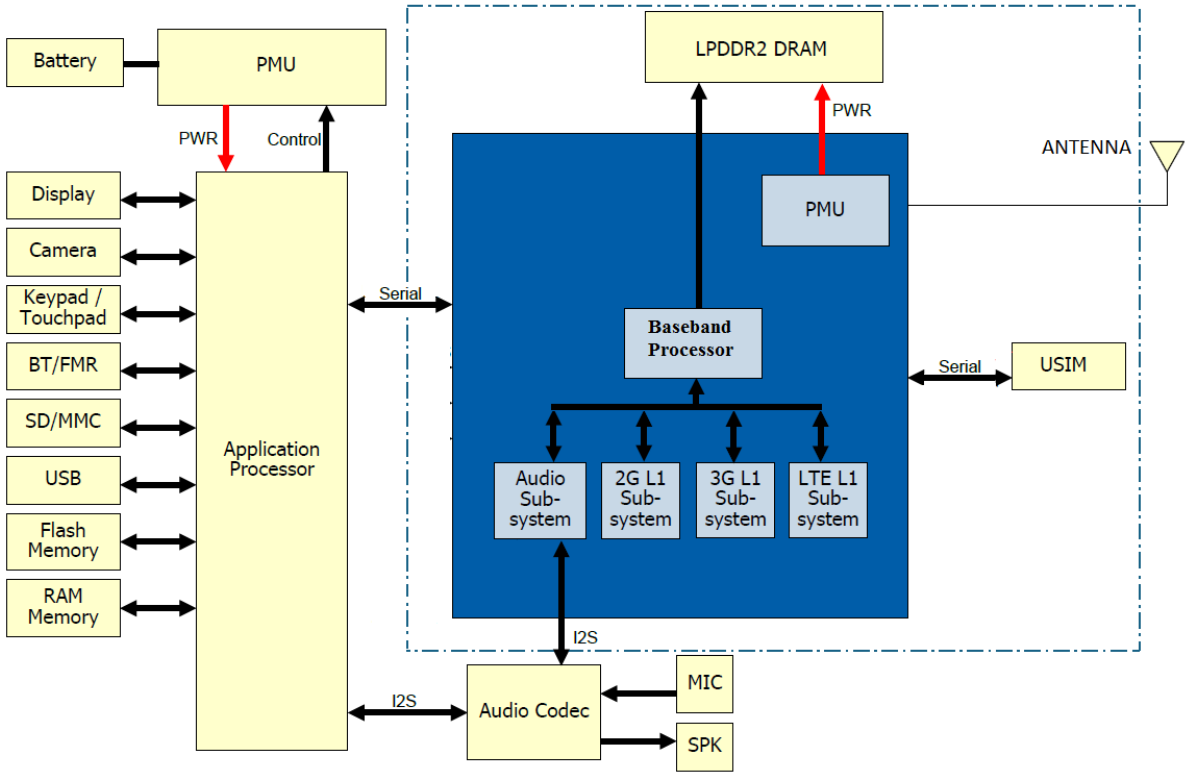


Figure 3.1: Logical Diagram of The Modem (Image courtesy Intel IMC Team)

X-GOLD 716 Baseband Controller and PMU

The X-GOLD 716 is a Baseband Controller with integrated modem power management unit. This system on chip is designed with the latest low power CMOS process technologies which provide significant performance to meet the ever increasing demands of the cellular subscriber market for feature rich terminals at lowest power consumption and a very competitive cost position.

The processing of the upper 2G/3G/LTE cellular protocol stack layer are handled by an ARM 1176 enhanced with level-1 and level-2 caches. For operations related to the data plane of the LTE and 3G protocol stack, a set of dedicated HW accelerators have been added. The physical layer of the cellular protocol stack is handled by dedicated hardware accelerator subsystem for GSM, UMTS and LTE. Advanced voice processing is provided by the inclusion of a rich audio DSP subsystem.

SMARTi 4G RF Engine

The RF Engine utilized on the XMM 7160 platform is based on the SMARTi 4G RF Transceiver. SMARTi 4G is a highly integrated LTE/UMTS/GSM/EDGE-transceiver with all necessary features to enable multi-mode, multi band mobile cellular devices. SMARTi 4G directly supports LTE/3G/2.5G/2G bands concurrently without additional discrete RF path switches.

The SMARTi 4G RF Engine on the XMM 7160 platform is an example of a RF solution for a quad-band GSM / hexa-band UMTS and up to nona-band LTE (plus sub-bands) mobile communication device product including RX diversity. Chosen to explain a typical application of the SMARTi 4G chip reference design it includes a Multi-mode PA, a Quad-band GSM / Penta-band UMTS/LTE Duplexer Filter Module, two single LTE Duplexer Filters and an Antenna Switch Module. In addition, it features an Antenna Switch Module and a GSM/UMTS/LTE Filter Module for the diversity receiver paths. Beside the above mentioned core devices, the SMARTi 4G RF Engine also supports the usage of an Antenna Tuner device. The Antenna Tuner can be connected via a plug-on board to the RF Engine antenna connector.

3.2 Power Control Trace Driver

The Power Control Trace driver is part of the Power Control driver subcomponent. AP Power Trace functionality which provides insite into Modem power states is a part of Power Control Trace driver. Primary motivation of the Power Control Trace driver is to trace function call sequences, significant Power Control events, internal states and data of the Power Control driver and its sub-components, e.g. CGU, PMU, SPCU and CPU.

Power Control Module is a very central component in the overall system architecture. This module used by every other driver in the system to fulfill their individual voltage

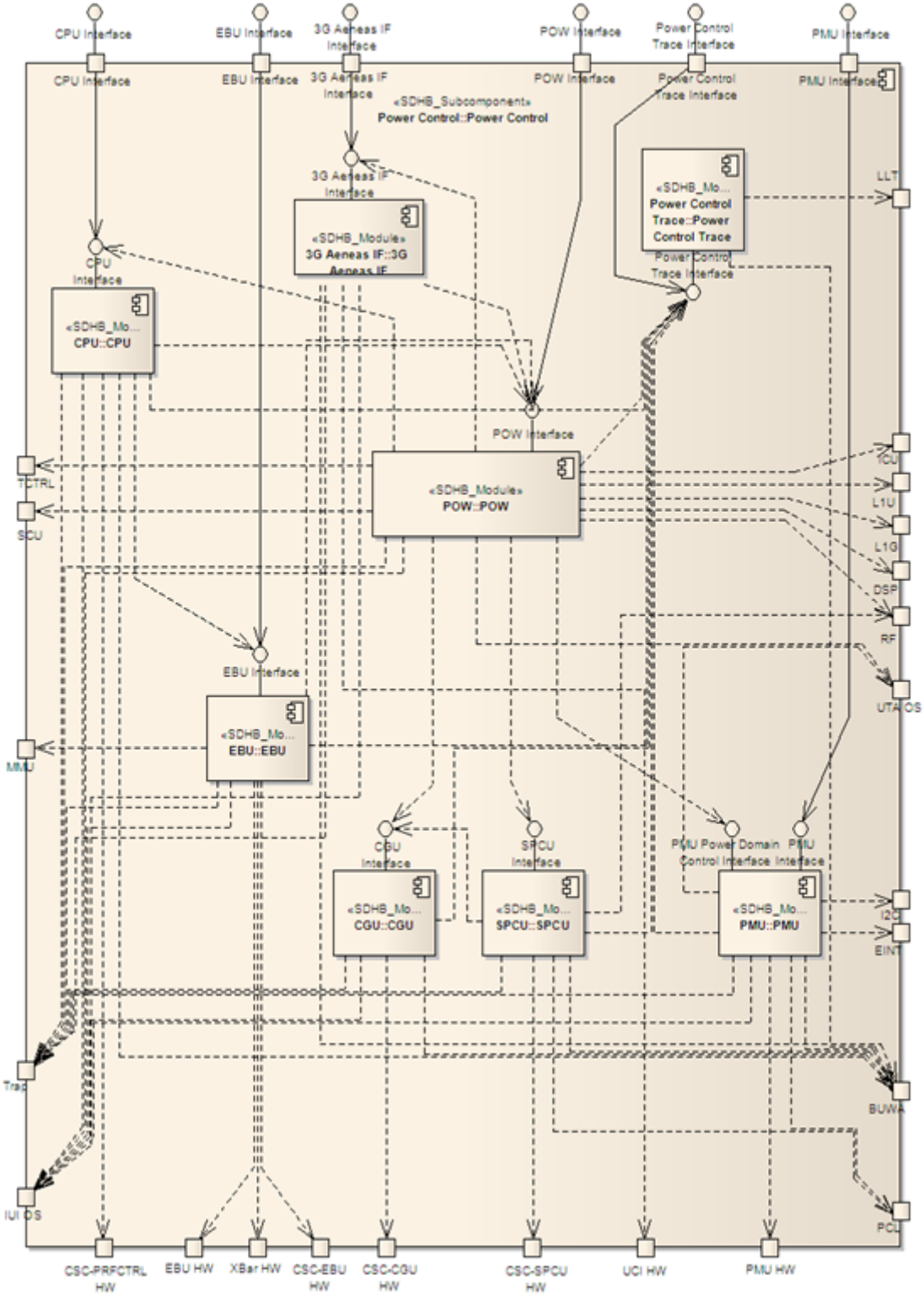


Figure 3.2: Power Control Trace Driver (Image courtesy Intel IMC Team)

and clock requirements. Power Control Module ensures that the most optimal clocks and voltage levels are maintained depending on the system activities and load, so that overall

power consumption can be minimized. This is achieved by monitoring idle states of the CPU, buses and some central peripherals, helping it to take decisions on entering deep sleep and how long the system can remain at low power.

From AP viewpoint it is of main interest to precisely know the time periods of various states. Therefore, this special AP trace mechanism omits a large part of the complex and detailed information as Power Control Trace could provide, but gives this data of rather statistical character. Thus, keeping the above goal in perspective, a combination of functionality both on BB and AP side are proposed to achieve these objectives.

3.3 AT Command Interface

AT Commands[**?**] [**?**] are the Attention Commands used for communication between Baseband (BB) processor, the one on the Modem and the Application Processor. All the commands passed by AP to BB are via AT commands. The command set consists of a series of short text strings which combine together to produce complete commands for operations such as dialing, hanging up, and changing the parameters of the connection.

AT commands work on a request response mechanism. Where the application processor issues an AT command to the modem over a predefined serial channel.The modem's baseband processor on receipt sends back a response to the AP. the response can be the data requested by the AT command of an acknowledgment. Some of the at commands are

- AT+CGMM :- The get the Modem Model
- AT :- Check if modem is live or not
- AT+CGMI :- Get Manufacturer information

Capturing the power data of the modem is made possible by implementing a custom AT command, which will give us requested data, from the modem firmware. Since, communicating with the modem is slightly different from communication with rest of the modules as it is external to SOC; the implementation of this module is slightly different from conventional techniques used.

Capturing the power details from the modem was made possible by the introduction of custom AT commands. The Application Processor (AP) sends a Start AT command and

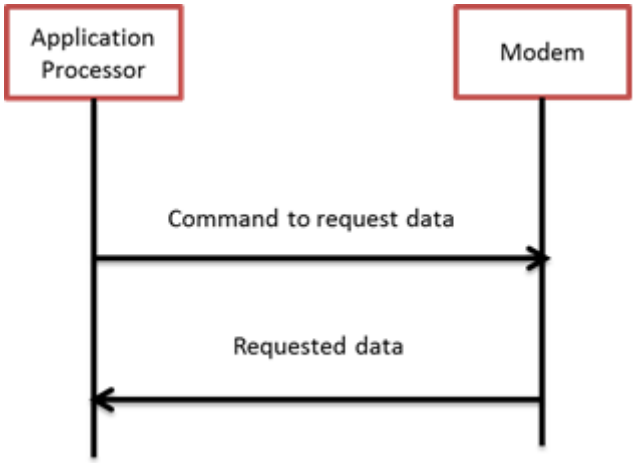


Figure 3.3: AT Command Interface with AP

by this determines start of AP tracing inside the Base band (BB). Once this command is received by the BB, the appropriate counter variables are populated and accumulate its values depending on the times the BB resides in which of those power states and as long as AP tracing is performed, respectively. If the AP wants to retrieve the collected trace data, it sends the stop AT command to the BB. If this AT command is received on the BB side, it immediately stops tracing, reads out the accumulated power state and trace time values, populates the AT response with it and provides this info to the AP.

3.4 RPC Interface

An Inter-process communication technique; Remote procedure call (RPC) allows a executable in one address space to trigger execution of computer program in another address space (in here Modem). This is achieved by abstracting the coding details from the programmer. This implies that the programmer essentially write the code in the similar way he uses for local subroutine execution. Thus from the programmers point of view, wheather the subroutine is remote or local to the program in execution does not matter.

In the RPC scenario the remote server is always known, while the client initiates the RPC communication by sending the request message consisting of the procedure to execute and the parameters accepted by the procedure.

Here the Application Processor is the host client while the Modem is the Remote

Server. The channel used is gsmtty21 (DLC21) for the RPC communication. The

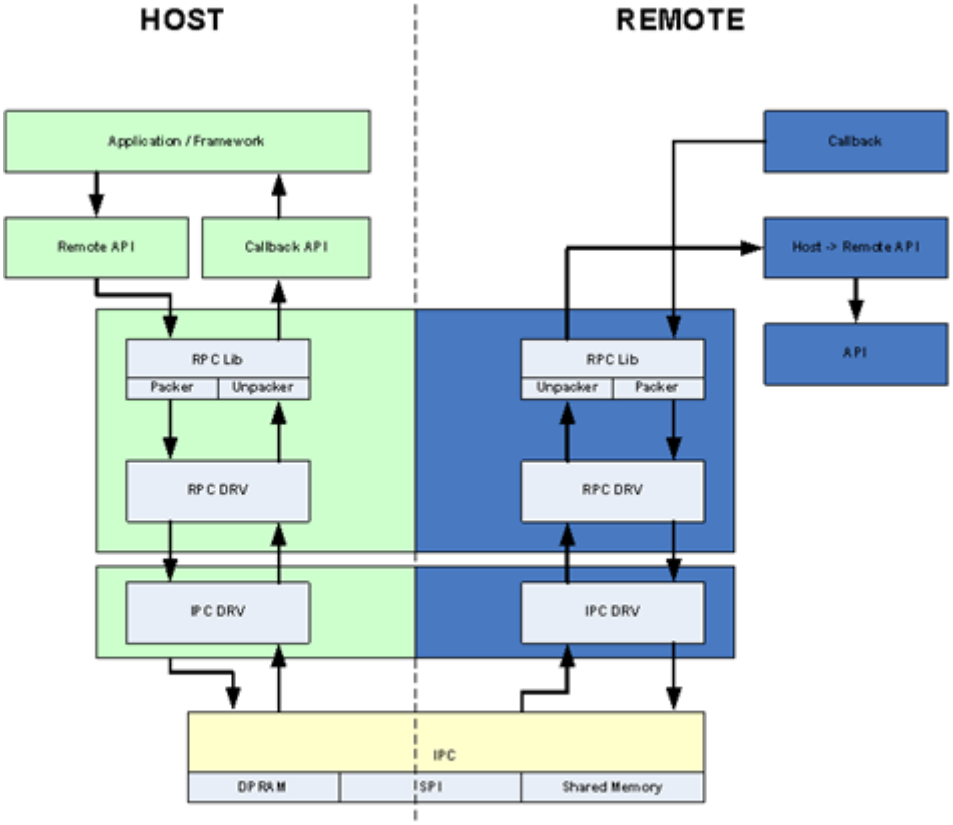


Figure 3.4: RPC Channel For Communication

host sends a request to the remote including the function it wants to be executed and its actual parameters. The remote sends a response, including the result after execution is completed.

The RPC API is divided into 4 layers:

1. **Remote API** :- Remote APIs are APIs which are executed in separate address space (commonly on another processor).

2. **RPC library** :- RPC library contains RPC packers and unpackers. Packer is a stub for actual APICallback(Remote API)which does marshalling(serialization) and sending the marshaled data to RPC message queue .Unpacker is a function which demarshall (de-serialize) the data received from ClientServer over IPC and invokes the appropriate API callback.

3. **RPC Queue** :- RPC maintains a single queue to receive data from packers and IPC. This message queue is processed by RPC task. Messages in queue are differentiated into write message and read message. This is differentiated by the readwrite param in the message queue .Write messages are posted by packers and read messages are posted by IPC driver.

4. **RPC DRV** :- RPC driver is responsible for processing the messages in RPC queue .It maintains a RPC task for processing the messages in RPC queue. Marshalled data in Write message is sent to IPC for transmission to the processor. Data from read queue is forward to Unpacker which will then invoke appropriate APIcallback after demarshalling the data.

5. **IPC driver** :- This is responsible for actual data transfer between processors. This can be realized using DPRAM, SPI, SHARED MEMORY etc.

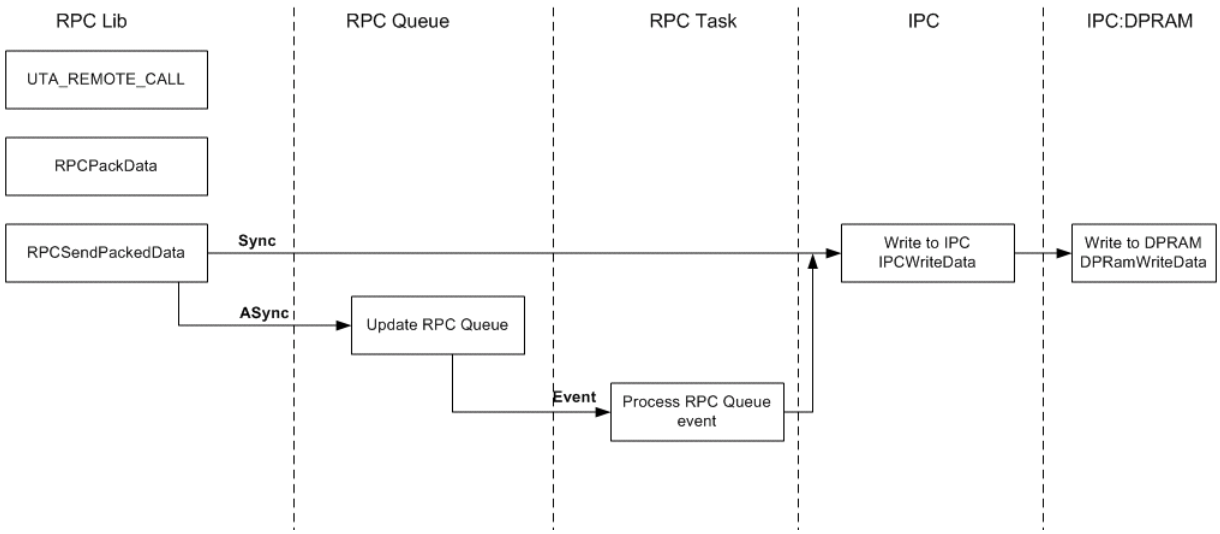


Figure 3.5: Data Transfer from Host to Remote

When application invoked the Remote API on host side the Remote API on Host side (Stub) serializes all the data (function params adds function id and the size of data actually serialized) and sends it to RPC driver through rpc queue. RPC task will then fwd the data in queue to IPC driver. IPC driver will then send the serialized data to IPC running on Remote side.

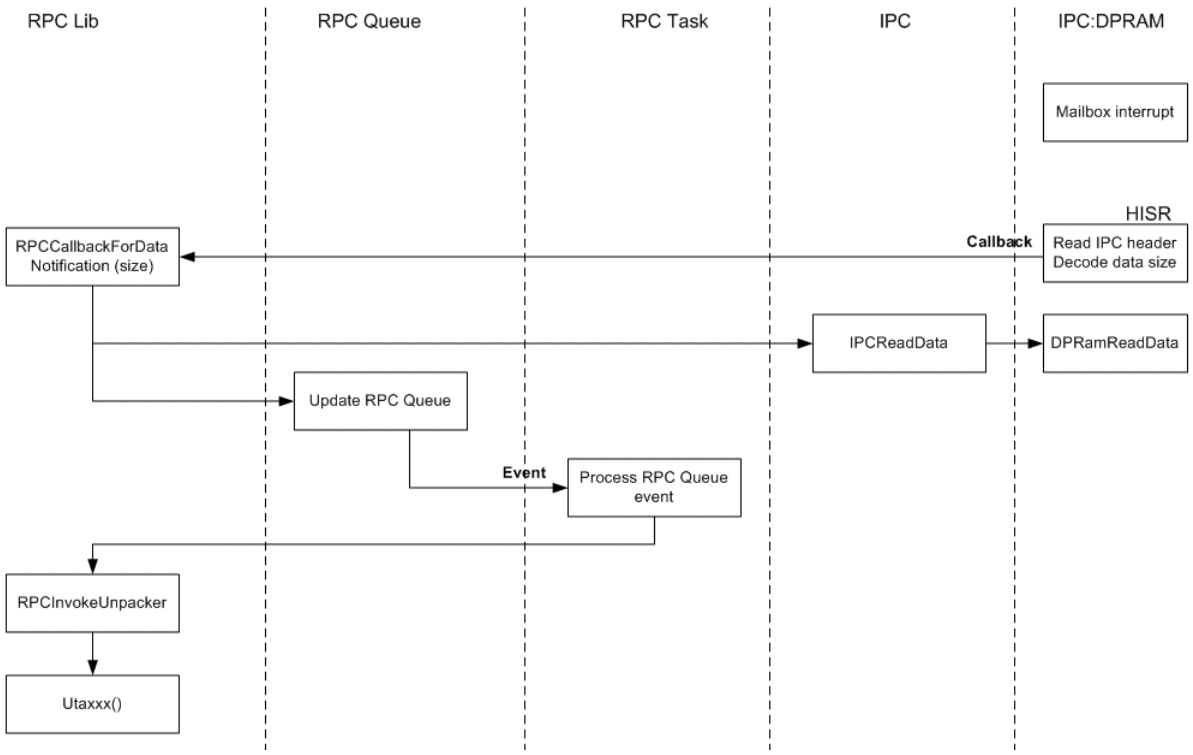


Figure 3.6: Data Transfer from Remote to Host

When Host IPC driver gets interrupt .It will read data sent by Remote IPC driver and invokes the RPC callback for data notification. RPC driver will then reads the data using IPC read and send the data to RPC task for processing using rpc queue. When RPC task receives this data it will invoke the Rpcunpacker to deserialize the data .After deserialzing the data Rpcunpacker will call appropriate APIcallback on Host side.

3.5 RPC based RIL

The purpose of Android RPC based RIL is to replace the AT based RIL by enabling remote execution of IMC (C function based) APIs from the Android side. For each of the RIL request in the android Vendor RIL Library, Aggregator library functions are called. These aggregator functions will have functionality to remotely execute the corresponding UTA Calls using RPC mechanism and handling the responseIndication coming from the Modem and it will be communicated to the above Android Telephony framework through RIL layer.

Android's Radio Interface Layer (RIL) provides an abstraction layer between Android

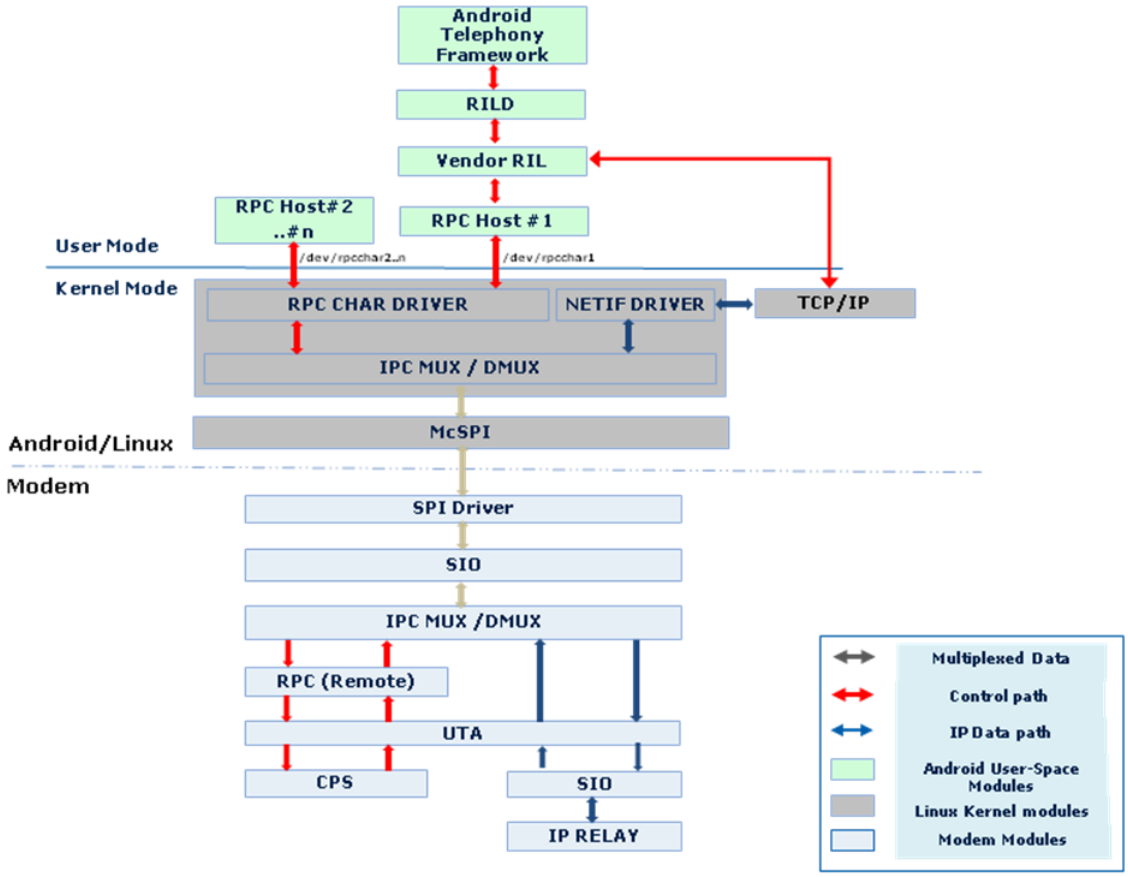


Figure 3.7: AP-BP Android RPC based RIL and IP-RAW Data flow

telephony services (Android telephony) and radio hardware (Modem). The RIL is radio agnostic layer supporting various Radio access technologies such as GSM-GPRS/UMTS. Android RIL consists of two primary components:

- **RIL Daemon** :- The RIL daemon initializes the Vendor RIL, processes all communication from Android telephony services, dispatches calls to the Vendor RIL as solicited commands and indicate solicited command response, unsolicited indication from vendor RIL to the android telephony framework.

- **Vendor RIL** :- Vendor RIL from IFX is RPC based. It contains a set of Aggregator library functions that invokes the necessary UTA function calls to perform different Radio operations. These UTA calls are executed remotely on the MEX side using RPC Mechanism. Aggregator function will give the response back to the vendor RIL to communicate suitably to the Android telephony framework via RILD.

Vendor RIL primarily does the following operations:

- Handling the RIL request originating from the Android telephony layer (via RILD) and call the appropriate aggregator library function handlers.
- Return back response information coming from aggregator library function handlers to the Android telephony layer in a suitable format via RILD.
- Processing and indicating unsolicited notification from the modem suitably to the above Android telephony layer via RILD.

RPC module on the Android side will pack each of the UTA calls and the corresponding parameters of the aggregator function as a packet and send it to the peer RPC layer running on the MEX via IPC. RPC layer on the MEX will unpack the packet and execute the corresponding UTA function. The response from UTA is packed into a packet and sent back from MEX RPC to the RPC layer on Android, where the corresponding aggregator function is invoked.

In case of AP-BP setup, Android RPC layer will interface with an IFX proprietary Linux kernel driver. This driver has the following functionalities:

- Exposes a character driver interface to the RPC module running on the Android.
- Provides network driver functionality by interfacing with Linux TCPIP stack and handle IP datagram.
- Provides a Multiplexing/De-multiplexing capability for IP data and RPC PGEN packets and sendsreceives packets tofrom SPI driver. In the remaining document this MultiplexingDe-multiplexing sub-module has been referred to as IPC MUX.

Similarly on the MEX side, there is a corresponding IPC MUX module which has the following functionalities:

- Interfaces with the USIF driver (Configured as SPI Slave) via the UtaSerial device API.
- Provides a MultiplexingDe-multiplexing capability for IP data and RPC PGEN packets. Data from USIF driver is de-multiplexed and send to RPC module (PGEN packets) or to SIO via UtaTerminal device API (IP data).

Chapter 4

PnP Analysis Tool: SocWatch

To get platform behavior related to power consumption for an SOC, instruments like Power Monitor and NI DAQ (Data acquisition systems) are used. They generate raw data in terms of power numbers which needs to be processed and correlated to get actual system behavior. This process is tedious and time consuming. Instead specific data related to system behavior can be directly captured on the system itself. SoCWatch is one such tool which resides on the host system to measure specific system behaviors related to power consumption. Moreover the tool automates the post processing thus giving precise required data.

SoCWatch is a command line tool for monitoring system behaviors related to power consumption on Intel architecture-based platforms. It analysis different aspects of the Intel Atom platform by monitoring CPU Power States, hardware accelerators like graphics, video, camera etc., other peripheral devices like audio, modem. It can capture bandwidths between the platform components like CPU-Memory, ISP-Memory IO-Memory etc. This tool provides extensive visibility into platform power states along with individual devices power states. These metrics provide insight into system behavior from energy efficiency perspective.

SoCWatch tool runs on the host platform where actual data collection is to be done. It can be configured to collect multiple features at the same time where collection duration can also be specified. The overhead of the tool is minimal, but depends on the collection type and features involved. The average overhead on the CPU can be between 2% to 5%.

SoCWatch tool at the simplest level can be described as; SoCWatch collects platform

related information from the hardware and processes it. The data to be collected depends on the feature requested.

4.1 SoCWatch Architecture

SoCWatch tool works on top of operating system, where at high level it can be divided into 2 parts. The application part where the user interface lies at Ring 3 (user space), and the driver part which communicates with the hardware at Ring 0 (kernel space).

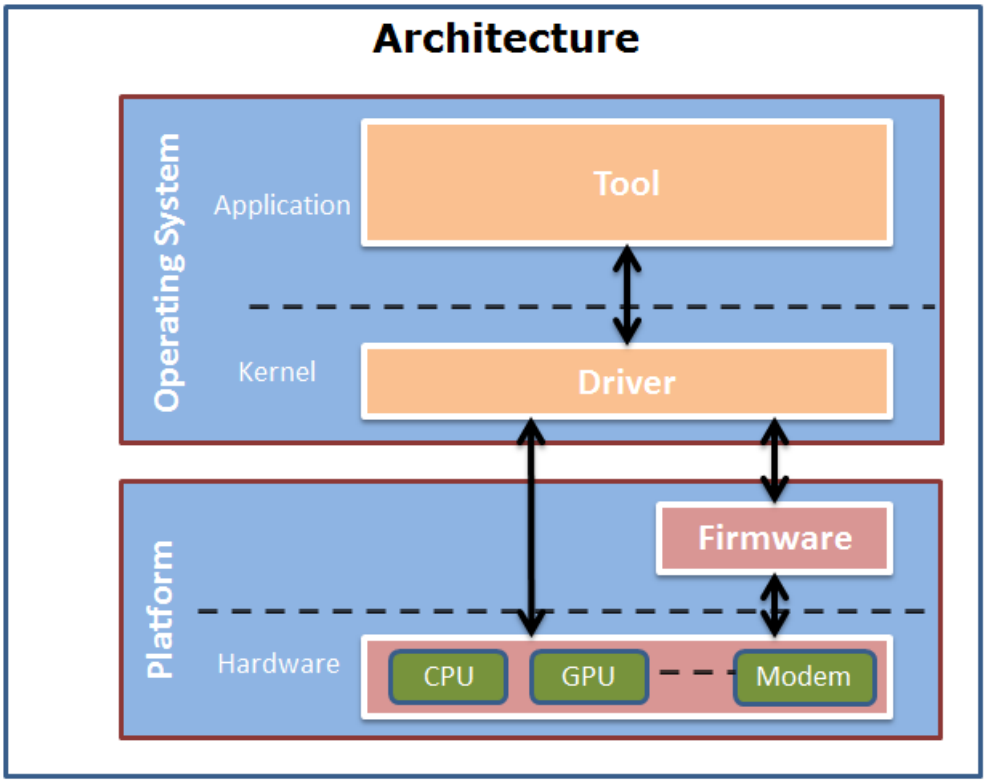


Figure 4.1: High Level SoCWatch Overview(Source Internet)

The SoCWatch Application directs the driver to collect data, which collects data from the hardware. And returns it to the application part, where the post processing is done.

4.1.1 SocWatch Application Part

The application side is the user interface which works at Ring 3. It takes command line input from the user and parses it to set appropriate parameters and preparing the tool for the collection. SoCWatch Configuration file is a database consisting of the hardware

address of various MSRs, memory locations required for the metric collection is read. After the initialization of the tool, application part controls the collection of data done by the SoCWatch driver. At the end after the collection, processing of the data is also done by application part.

4.1.2 SocWatch Driver Part

The SoCWatch Driver resides in the Linux kernel. It gets the metadata from the application part about the data to collect such as address, data size and no of records to capture. It collects the specified data from the underlying hardware using Linux tracepoint mechanism. After the complete collection the data is passes to the application part.

4.2 SoCWatch Operation Mode's

Features are the actual metrics that can be collected using SoCWatch. SoCWatch has defined 3 modes for collection data. Features can be captured in one of the following modes.

4.2.1 Snapshot Mode

In Snapshot the metric is captured at the starting and ending of the run. In this mode the counters are tracked by the hardware thus there is no overhead induced during the actual collection.

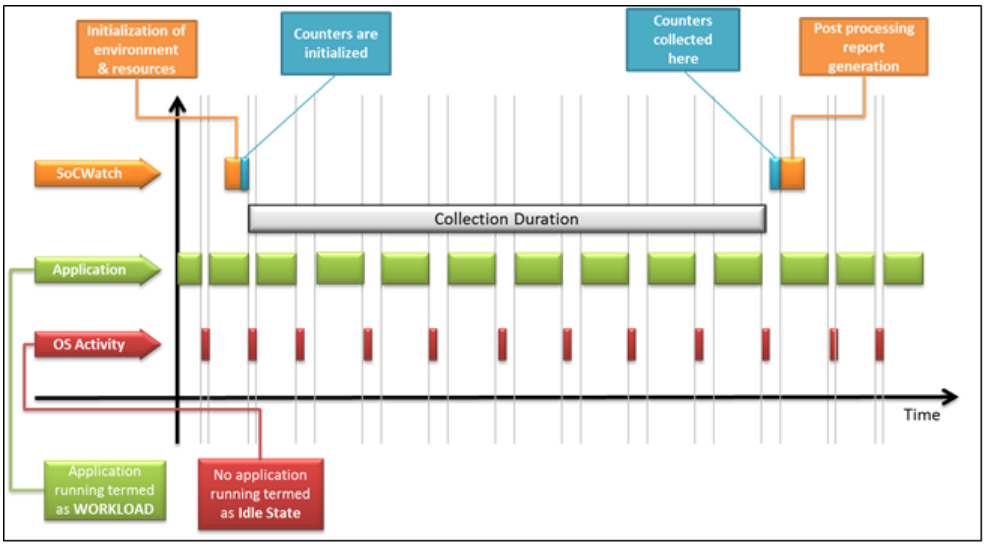


Figure 4.2: SoCWatch Snapshot Mode (Source Internet)

4.2.2 Polling Mode

In polling mode the metric is captured after regular interval of time for the complete run duration. The polling interval can be specified. Some features cannot support snapshot mode such as Pstate, GPU Cstate/Pstate (hardware limitation).
NOTE: Polling mode increases CPU overhead. Polling overhead may vary based on the features and the polling interval.

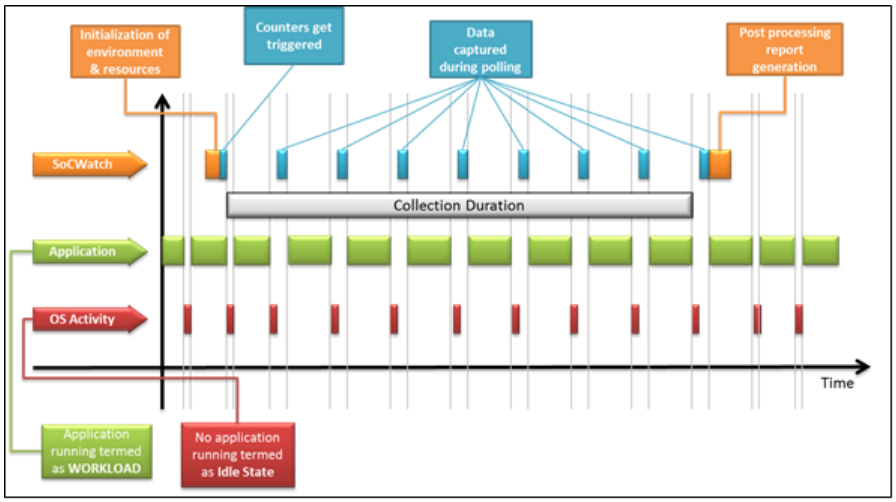


Figure 4.3: SoCWatch Polling Mode (Source Internet)

4.2.3 Tracing Mode

Tracing mode is an event based mechanism, where the data is captured whenever there is a transition from one state to another for the feature. Here for demonstration purpose; during run only 3 data traces are collected.

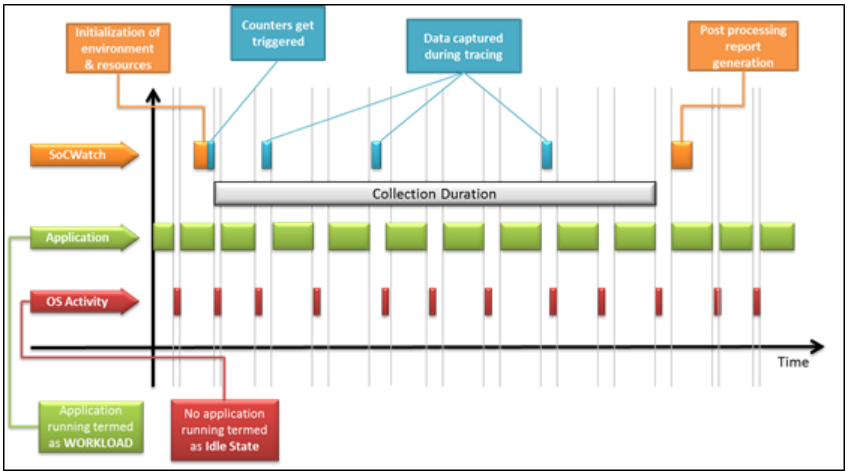


Figure 4.4: SoCWatch Tracing Mode (Source Internet)

4.3 Feature Framework

Each feature has to follow the framework defined by the state machine. As you can see in the figure the framework can be broadly divided into 3 parts.

Before the actual collection, the feature opted has to be initialized. The initialization includes the extracting of the hardware MSRs address from SoCWatch configuration file that are passed to driver to collect data from. And other required platform information. This is taken care in the Initialize().

The StartPreamble() registers and allocates memory to the MSRs, memory locations to be read during the run. Here it is check whether the feature opted or not. And if opted appropriate collection setting are done, and features to capture are set.

During the Data capture phase there is no feature specific task, the driver captures the data and stores it in Ring 0 for the duration of the run. Later the captured data is transferred to Ring 3.

StopPostamble() does the post processing on the data captured. The processing is actually extracting information from the captured data. The data is parsed and required information is extracted for post processing.

LogReport() is actual report generator. The report can be a consolidated summary of the run or a trace file consisting of data captured at regular interval.

Deinitialize() DE allocates all the application memory initialized for the feature.

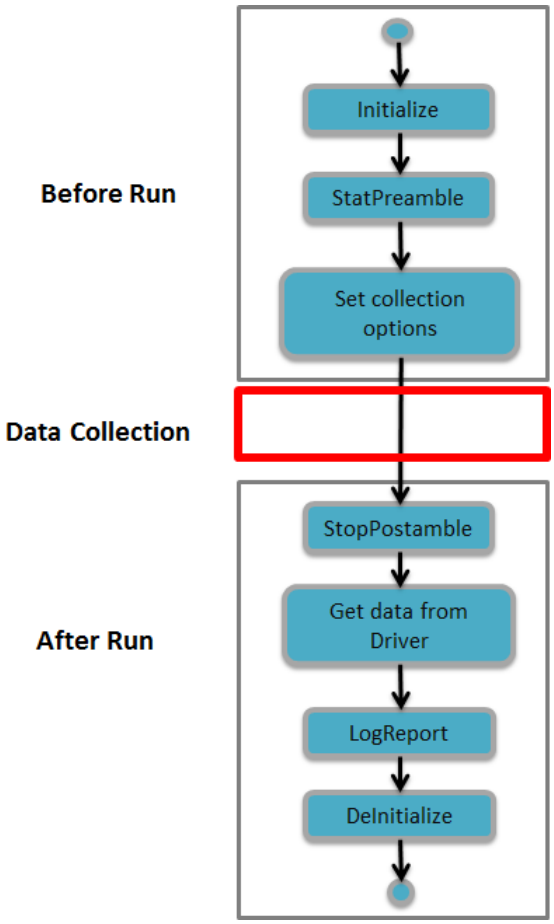


Figure 4.5: Feature State Transition (Source Internet)

This is the complete execution cycle for a feature in SoCWatch. Some of the features have additional functions limited to themselves, used in supporting the actual framework.

Chapter 5

Implementation

Integrating modem power capturing capability to SoCWatch would be beneficial in the following ways:

1. Enable better platform power projections for future architectures.

2. Produce first level of power debug for CP & AP. By easily generating all the statistics, the engineers working on AP and CP interface would be able to quickly communicate with each other and root-cause the issues.

3. Platform PnP optimization: A cohesive approach to platform PnP analysis, where CP and AP statistics are analyzed at the same time would help in more efficient optimization for scenarios like browsing and audio/video streaming over 3G4G network.

5.1 AT Interface Based Implementation

Capturing the power data of the modem is made possible by implementing a custom AT command, which will give us requested data, from the modem firmware. AP Power Trace hides power control driver details and provides information: the BBs sleep, idle and its active times. AP trace mechanism omits a large part of the complex and detailed information as Power Control Trace could provide, but gives this data of statistical nature. But the data provided needs to be decoded at the AP side before it can be understood completely. SoCWatch does the decoding part.

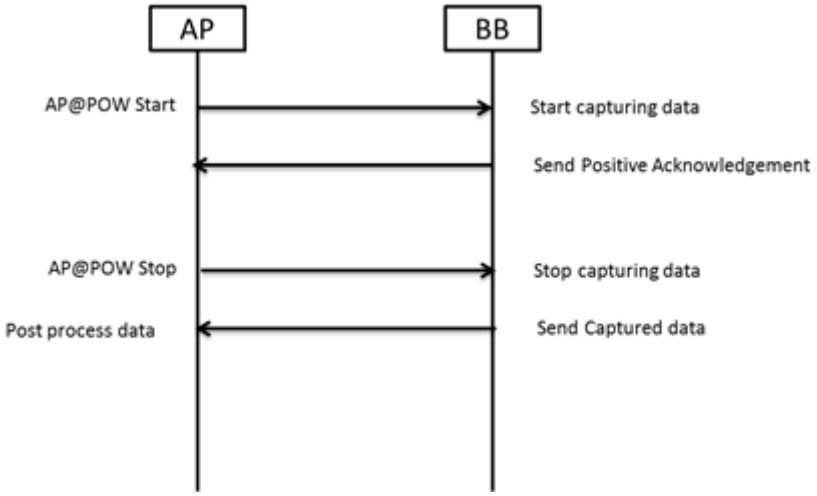


Figure 5.1: AP Power Trace Mechanism

5.1.1 AP Power Trace Modes

AP Power Trace operates in 2 modes

1. Asynchronous mode
2. Cyclic mode

Asynchronous Mode

Asynchronous mode corresponds to Snapshot mode in SoCWatch. In the mode of operation AP sends a start command to BB. BB on receiving this command starts capturing the time BB resides in corresponding power state. This goes on till AP does not issue stop command to BB. Which in response stops tracing and sends the captured data to AP.

If BB reports any error during start or stop commands SoCWatch reports the type of error and stops capturing the feature.

NOTE: this does not affect the other features being captured at the same time.

AP command to start AP Power Trace in Asynchronous mode

atpow:start_ap_trace (cycle_time, trace_select)

cycle_time = 0 (specifies mode of operation as Asynchronous to BB)

trace_select = 0xFFFFFFFF (capture all possible data Deep Sleep time, Idle time, Active Time)

NOTE: there can be multiple variations for trace_select specifying AP Trace to capture only Deep Sleep Time and Active Time or Idle Time and Active Time. But SoCWatch always specify to capture all the 3 times.

AP command to stop AP Power Trace in Asynchronous mode

atpow:stop_ap_trace ()

No parameter is specified to stop AP trace.

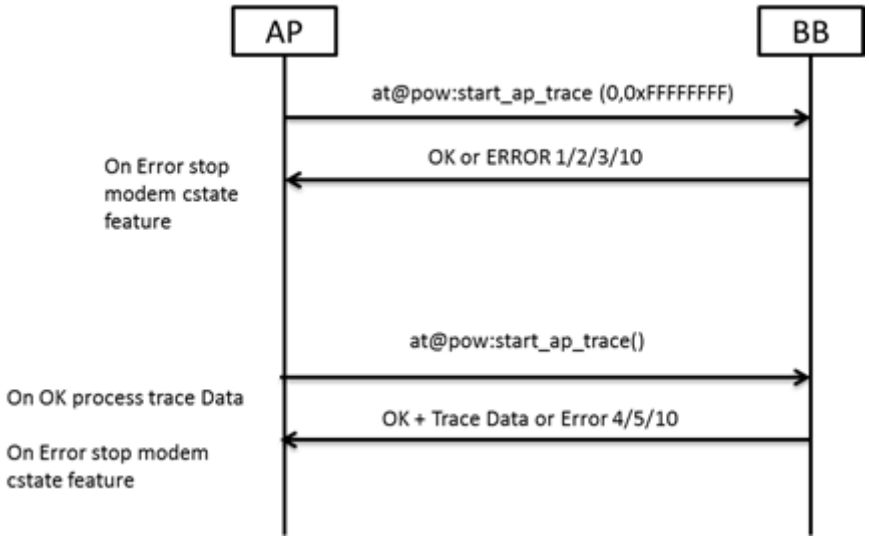


Figure 5.2: AP Power Trace in Asynchronous Mode

Cyclic Mode

Cyclic mode corresponds to Polling mode in SoCWatch. Cyclic mode AP Power Trace is similar to Asynchronous mode. But her we specify cycle_time parameter in the start command other than 0. This cycle_time corresponds to the time interval after which BB should send the trace data until stop command is specified by AP. The cycle time by default is 100 ms. But it can be set by n option given via command line to SoCWatch.

The minimum time interval supported by BB is 1 ms, while maximum time interval is 2.5 seconds. The command sequence followed in cyclic mode is same as that in asynchronous mode, but the only difference is that we need to specify cycle_time other than 0.

AP command to start AP Power Trace in Cyclic mode.

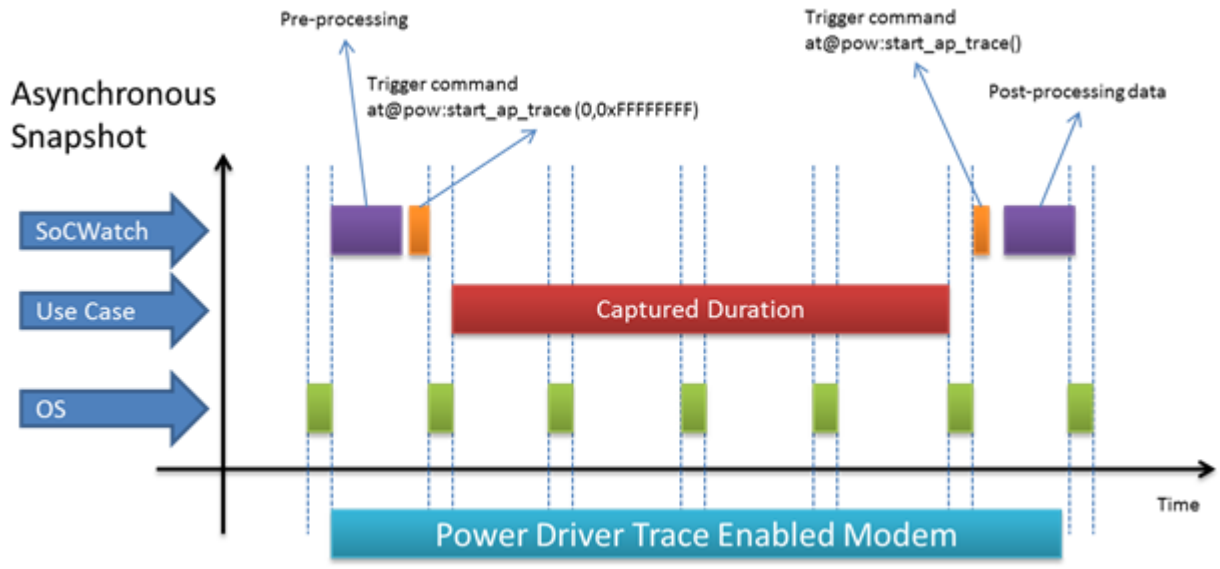


Figure 5.3: Modem Snapshot Mode in SoCWatch

atpow:start_ap_trace (cycle_time, trace_select)

cycle_time = greater than 0 (specifies mode of operation as Cyclic to BB)

trace_select = 0xFFFFFFFF (capture all possible data Deep Sleep time, Idle time, Active Time)

NOTE: there can be multiple variations for trace_select similar to Asynchronous mode.

AP command to stop AP Power Trace in Cyclic mode

atpow:stop_ap_trace ()

No parameter is specified to stop AP trace.

5.1.2 AP Trace Decode By SoCWatch

Whenever start AP trace command is issued by AP, BB initializes the counters, and starts populating them based on BB Cstates. When stop AP Trace command is issued BP stops incrementing counters and returns the counter contents.

The Response given by BB on successful trace will be of following format

POW:OK,SLEEP_TIME:{count},SLEEP_OVERRUN:{count},
IDLE_TIME:{count},IDLE_OVERRUN:{count},ACTIVE_TIME:{count},
ACTIVE_OVERRUN:{count}

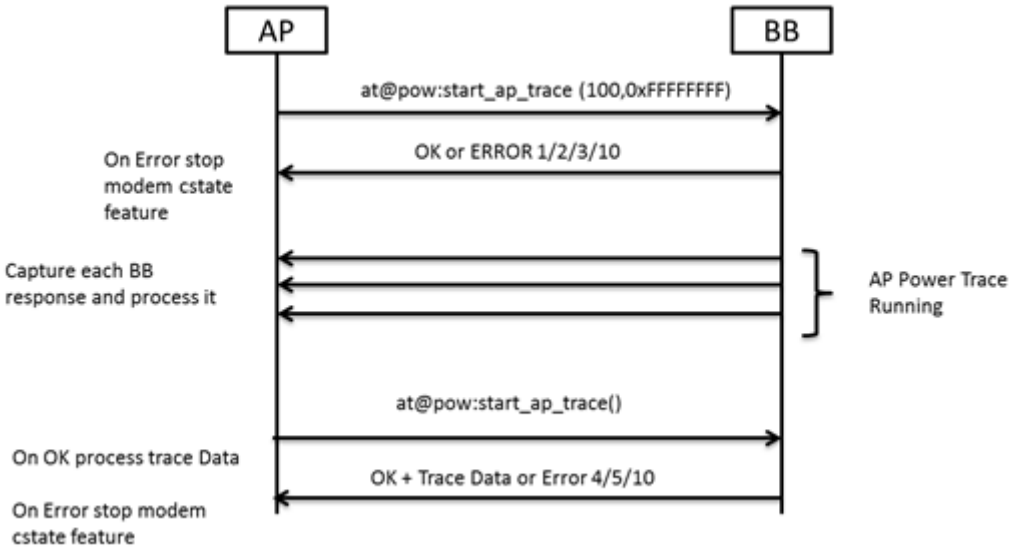


Figure 5.4: AP Power Trace in Cyclic Mode

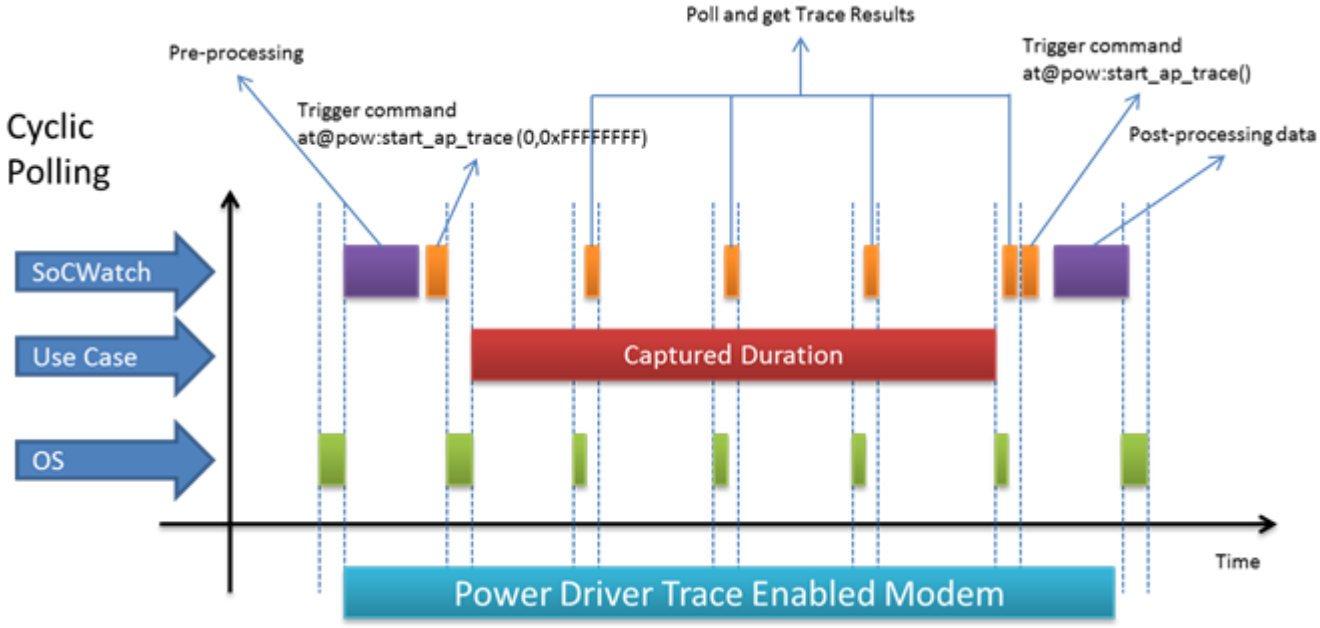


Figure 5.5: Modem Polling Mode in SoCWatch

We segregate the Sleep, Idle and Active time with corresponding overruns. 1 overrun is equivalent to 0x100000000 counts i.e. $2\hat{3}2$.

SLEEP_TIME is multiple of TDMA frame, which is 4.615 ms.

SLEEP_TIME = counter_value * 4.615

IDLE_TIME and ACTIVE_TIME are raw System time counts based on 26 MHz clock

IDLE_TIME = counter_value $(26 * 10\hat{6})$

ACTIVE_TIME = counter_value (26 * 10$\hat{6}$)

After normalizing all the counter reading we can get the total AP Trace time

AP_Trace_Time= SLEEP_TIME + IDLE_TIME + ACTIVE_TIME

Can we can calculate percentage residency

SLEEP_PERC = SLEEP_TIME AP_TRACE_TIME *100

Similarly for IDLE_TIME and ACTIVE_TIME.

This happens in snapshot mode. While in polling mode same technique is followed but the difference between two consecutive polled records is considered.

5.1.3 AP Modem Trace Module Integration in SoCWatch

The flow of function call to capture modem c-state data in SoCWatch is as follows.

1. Initially we open modem as a device via a predefined port gsmtty10.
2. If modem is connected we check if the modem model version is compatible or not. The modem model should be XMM7160.
3. If modem is compatible we will first stop any previous AP Trace command. (This is to stop any already running AP Trace that was not stopped probably by SoCWatch only).
4. Now the actual AP Trace either in snapshot mode or trace mode is started.
5. After the run duration AP Stop Trace command is issued.
6. And later on post processing on the received data is done.

NOTE: Steps 1,2,3 are performed during modem feature initialization, while rest are executed during the actual data collection.

5.1.4 Error Handling for AP Modem Trace Feature

Whenever an error is encountered will communicating to and from modem the feature capture is stopped and corresponding error is displayed to the user.

5.2 RPC Based Implementation

To understand the Modem behavior it's necessary to monitor various modem specific metrics. These insights give a better understanding of the functioning of modem and help in optimizing it's working.UTA is one such methodology used for monitoring Modem metrics.

UTA technique gives a unified and efficient mechanism to access various Modem metrics. UTA implements is a command line tool which resides on the Application processor and communicates with the modem to monitor a metric. It registers the metric along with a call back function with the modem. Modem sends the data which is handled by the registered callback function.

5.2.1 SoCWatch UTA Architecture

From SoCWatch perspective UTA Handler and the communication between AP & BP is Black Box. UTA Library provides the interface to access the necessary uta calls for the uta functionality to work. Similarly the RPC used for the communication between the Application and Communication processor is managed by the UTA Library (API).

UTA API has two main components as shown in the diagram; The UTA metric that actually does the metric capturing and the RPC for communication between the processors.

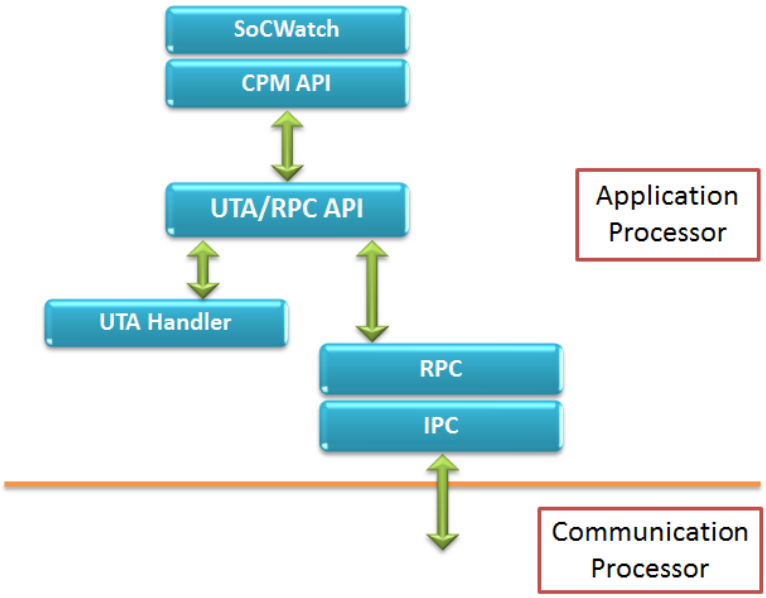


Figure 5.6: UTA Based SoCWatch Architecture

UTA API

The UTA Metric engine is the actual one on the modem side which collects the data on the modem and sends its back to the AP. The metric data is collected from various components of the Modem Stack. Reporting of metric is done either periodically or on event occurrence depending on the metric type. The API's provided by UTA are for Registering and De-Registering the Metric. When a Metric is registered, corresponding call to the modem is made which starts collecting the metric and send it back to the callback function. The syntax is

UTA_REMOTE_CALL(UtaRegisterHandler)(g_nInstance, &g_UMetricId, HandlerFunction)

The arguments passed to the register function are

1. Instance = 0 (default)
2. UMetricId = consists of the MetricGroupID and MetricID
3. HandlerFunction = callback Function

Similarly the De-register function makes modem stop collecting the Metric. The Syntax is

UTA_REMOTE_CALL(UtaDeRegisterHandler)(g_nInstance, &g_UMetricId, HandlerFunction)

The callback function that extracts the data from the modem response varies feature wise. Some features require normalization as per the 3GPP standards. The Syntax is

void utaMetricsHandlerFunction(int nInstance, UtaMetricsId * metric_id, UtaMetricsData* metricData)

The arguments are

1. Instance = 0 (default)
2. UMetricID = consists of the MetricGroupID and MetricID

3. MetricData = structure containing data collected

5.2.2 RPC API

An Inter-process communication technique; Remote procedure call (RPC) allows a executable in one address space to trigger execution of computer program in another address space (in here Modem). This is achieved by abstracting the coding details from the programmer. This implies that the programmer essentially write the code in the similar way he uses for local subroutine execution. Thus from the programmers point of view, wheather the subroutine is remote or local to the program in execution does not matter.

In the RPC scenario the remote server is always known, while the client initiates the RPC communication by sending the request message consisting of the procedure to execute and the parameters accepted by the procedure. Here the Application Processor is the host client while the Modem is the Remote Server. The channel used is gsmtty21 (DLC21) for the RPC communication.

The host sends a request to the remote including the function it wants to be executed and its actual parameters. The remote sends a response, including the result after execution is completed.

RPC stubs generated will be used for data transfer between processors. The tool uses the header files of drivers to create RPC stubs.

Example:

Consider this API is implemented.

U32 dummy_driver_api1 (DUMMY_DRIVER_ENUM_T paramA, Dummy_Driver_STRUCT1 paramB);

The proxy implementation file stays in the driver module which makes the RPC calls.

The proxy implementation file is written in similar lines:

#include {packer_unpacker_.h}

U32 dummy_driver_api1 (DUMMY_DRIVER_ENUM_T paramA, Dummy_Driver_STRUCT1 paramB); {

RPC_RETURN_RESULT_T rpc_result;

U32 RetVal;

RetVal = REMOTE_CALL (dummy_driver_api1)(&rpc_result , paramA , paramB);

if(RPC_RESULT_SUCCESS == rpc_result) {

return RetVal;

} }

For AP to use the RPC mechanism to connect to CP, a request call has to be made through RPC. Requests are any call that is required by the client of RPC to reach to the other side (CP side)

To call an RPC API, append the actual API with a macro: *UTA_REMOTE_CALL*

Example:

Actual Call needed on the AP side:

return_val = UtaMsCallCsJoinCallsReq(nInstance, context_id);

RPC Call to be done on the AP side:

return_val = UTA_REMOTE_CALL (UtaMsCallCsJoinCallsReq)(nInstance, context_id);

5.2.3 UTA Library

UTA library acts as an interface between the SoCWatch tool and the UTA RPC API's. It gives a unified and efficient mechanism to access various Modem metrics. UTA_Metric Engine on the modem side collects the data which is passed to UTA_Handler on the Application processor. The communication between the processor is done via RPC IPC mechanism.

CPM Library is used by SoCWatch to Register and De-register the UTA Metrics and initialize the RPC for the communication using the underlying IPC technique (Inter-Process Communication).

The SoCWatch uses interface for UTA metric Registration and De Registration, and similarly RPC API for initializing the communication channel. All the interfaces to these API are wrapped in the Library.

5.2.4 Implemented Modem Metrics

SoCWatch has considered the metrics which are important from power and performance perspective. The metrics collected are related to components and activities which are power hungry. This will help in understanding the modem power consumption model

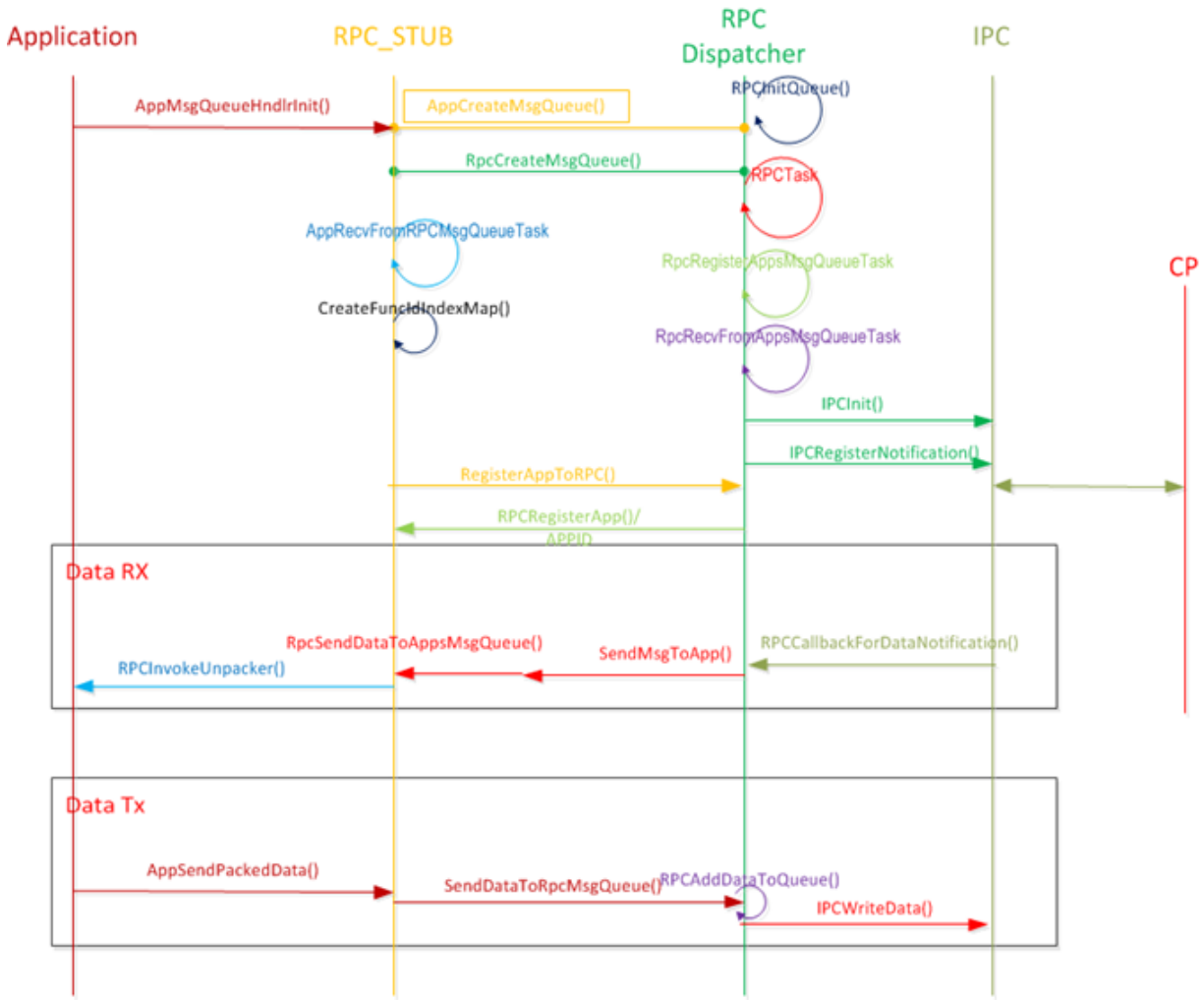


Figure 5.7: Data Transferring internally in RPC levels

Figure 5.8: SoCWatch view of CPM Mechanism

in greater depth.

The metrics that are currently implemented by the SoCWatch are

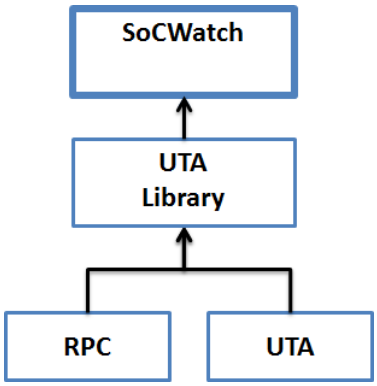


Figure 5.9: Components of UTA & RPC Based Implementation For SoCWatch

Name	**Group**	**Description**	**Collection**
RF Info	CIQ0	L1 (Radio) Layer Cell related Info	Periodic
LTE RRC States	CIQ0	LTE RRC State and cause of change	Event
UMTS RRC States	CIQ0	UMTS RRC State	Event
PUSCH Tx power	ELT2	Physical Uplink shared channel tx power for frequency hopping	Periodic
Radio Mode	CIQ0	Get the type of network/service	Event
Service State	CIQ0	Get Radio & Network Service states	Event
Physical Cell ID	ELT2	Get the physical cell ID of the serving cell	Period
RSRP in LTE	ELT2	The average RSRP (Reference Signal Received Power) of the LTE serving cell	Periodic
Cellular Power Statistics	**CPM3**	**Get Modem Power States Residencies**	**Periodic**
LTE RLC Data Transfer	CIQ0	E-UTRA RLC Data Transfer Report Upload / Download	Periodic

Table 5.1: SoCWatch Implemented Features

Metric Fields	**Type**	**Explanation**
Sleep Time	unsigned int	Time BB spent in SLEEP State
Sleep OverRun	unsigned int	Count Sleep Time Overflows
Idle Time	unsigned int	Time BB spent in IDLE State
Idle OverRun	unsigned int	Count Idle Time Overflows
Active Time	unsigned int	Time BB spent in Active State
Active OverRun	unsigned int	Count Active Time Overflows

Table 5.2: Cellular Power Metric Fields

Chapter 6

Result

The PnP Analysis tool has been tested for Modem metrics collection using Network Simulators CMW500 and Agilent PXT against 2G/GMS 3G and LTE/4G networks. These devices simulate real world working live mobile network environments.

At the same time the feature has been tested on live mobile networks. The results are approximately the same.

6.1 AT Interface Based Results

Following are the use-cases for which the features are tested.

Sr.	Use-Case	Expected Output	Actual Output	Result
1	2G Idle (no modem activity)	High Sleep Time Residency	High Sleep Time Residency	Pass
2	2G call	High Idle Time Residency	High Idle Time Residency	Pass
3	3G Idle (no modem activity)	High Sleep Time Residency	High Sleep Time Residency	Pass
4	3G call	High Sleep Time Residency	High Sleep Time Residency	Pass
5	Online video streaming	High Idle Time Residency	High Idle Time Residency	Pass
6	3G data Idle mode	High Sleep Time Residency	High Sleep Time Residency	Pass

Table 6.1: Modem testing UseCase for AT based Implementation

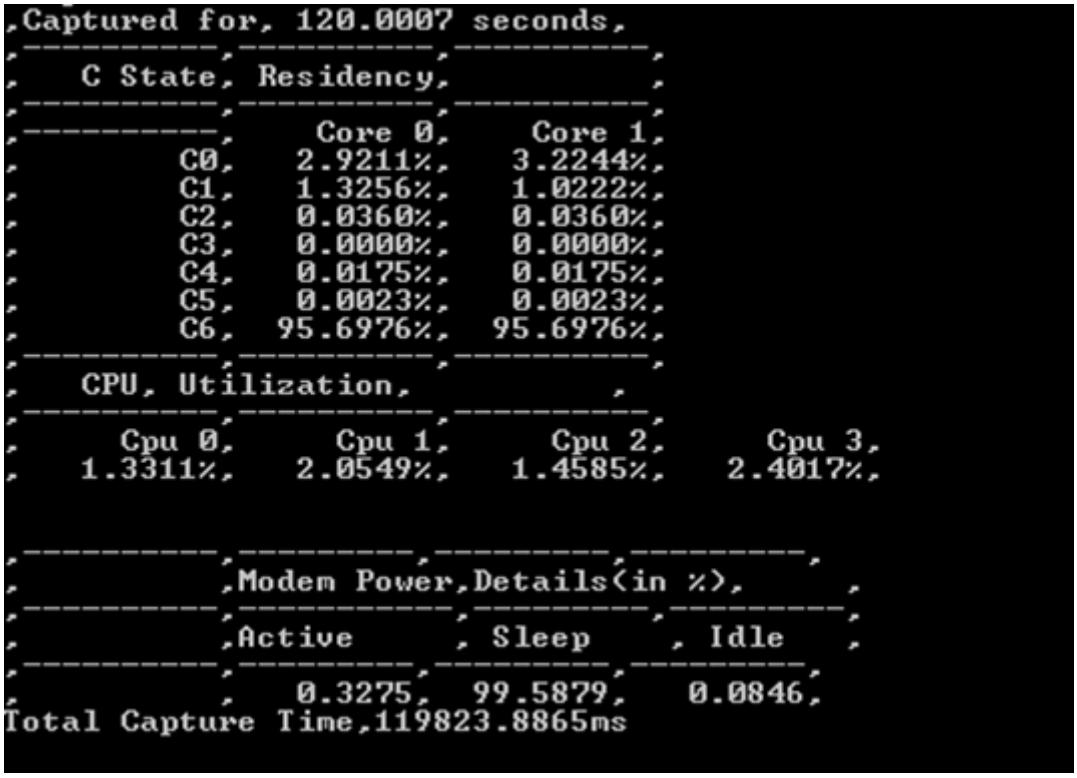


Figure 6.1: Modem Activity with 3G Idle Mode (No Data)

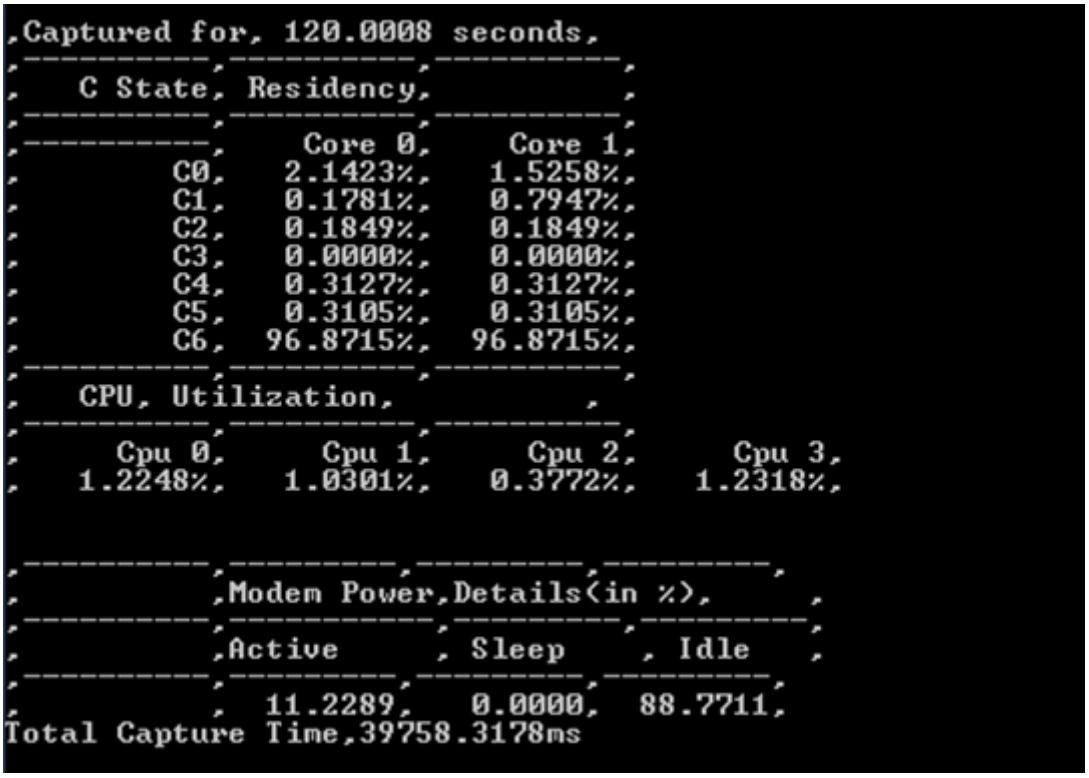
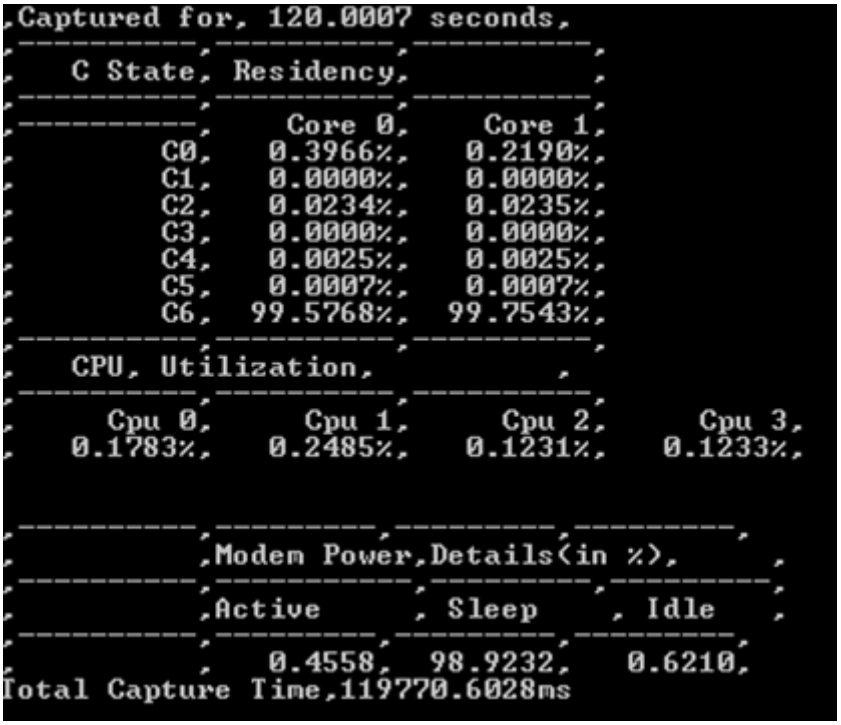


Figure 6.2: Modem Activity during a 3G Voice Call



Captured for, 120.0007 seconds,		
C State, Residency,		
	Core 0,	Core 1,
C0,	0.3966%,	0.2190%,
C1,	0.0000%,	0.0000%,
C2,	0.0234%,	0.0235%,
C3,	0.0000%,	0.0000%,
C4,	0.0025%,	0.0025%,
C5,	0.0007%,	0.0007%,
C6,	99.5768%,	99.7543%,

CPU, Utilization,			
Cpu 0,	Cpu 1,	Cpu 2,	Cpu 3,
0.1783%,	0.2485%,	0.1231%,	0.1233%,

	Modem Power, Details(in %),		
	Active	Sleep	Idle
	0.4558,	98.9232,	0.6210,

Total Capture Time,119770.6028ms

Figure 6.3: Modem Activity during 3G data Idle Mode

6.2 RPC Interface Based Results

```
****************************************

Registering...
Registered for RF60
Registered for GS34
Registered for GS46
Registered for GS67
Registered for LT01
Registered for LT02

****************************
METRIC LT03     4270396 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4270396 PhyCellID=3362  EARFCN=348      RSRP=8dBm       RSRQ=-8.5dB
NEIGHB CELL#2   4270396 PhyCellID=282   EARFCN=289      RSRP=64dBm      RSRQ=82.5dB
****************************Registered for LT03
Registered for EL06
METRIC EL06     4271658 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4271677 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4271677 PhyCellID=3618  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************Registered for LT07
METRIC EL06     4272660 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4273012 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4273012 PhyCellID=4643  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************Registered for LT09
METRIC EL06     4273662 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4274237 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4274237 PhyCellID=4386  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************Registered for LT12
METRIC EL06     4274663 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
Registered for EL04
****************************
METRIC LT03     4275517 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4275517 PhyCellID=4386  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
METRIC EL06     4275665 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
Registered Metrics...
METRIC EL06     4276667 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4276797 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4276797 PhyCellID=4130  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
METRIC EL06     4277669 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4278077 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4278077 PhyCellID=3874  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
METRIC EL06     4278671 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4279357 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-102dBm    RSRQ
NEIGHB CELL#1   4279357 PhyCellID=4130  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
METRIC EL06     4279672 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4280637 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-101dBm    RSRQ
NEIGHB CELL#1   4280637 PhyCellID=1826  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
METRIC EL06     4280637 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
METRIC EL06     4281642 TAC=35653       DL Bandwidth=10 MHz     PhyCellID=262   NumTxAntenna=2  RSRP
****************************
METRIC LT03     4281917 MCC=310 MNC=410 PhyCellID=262   EARFCN=5780     Band=17 RSRP=-101dBm    RSRQ
NEIGHB CELL#1   4281917 PhyCellID=3618  EARFCN=348      RSRP=64dBm      RSRQ=82.5dB
****************************
```

Figure 6.4: Modem Feature Collection using RPC mechanism

Name	Network	Use Case Scenario	Expected Output	Actual Output	Result
RF Info	3G	network Activity Call/ Idle	data must be aligned with simulator	data must be aligned with simulator	Pass
LTE RRC States	4G	Switch to and from Airplane/ Standby/Active	data must be aligned with simulator	data must be aligned with simulator	Pass
UMTS RRC States	2G/3G/4G	Switch to and from Airplane/ Standby/Active	data must be aligned with simulator	data must be aligned with simulator	Pass
PUSCH Tx power	4G	network Activity Call/ Idle	data must be aligned with simulator	data must be aligned with simulator	Pass
Radio Mode	3G	Switch to and from Airplane/ Standby/Active	data must be aligned with simulator	data must be aligned with simulator	Pass
Service State	3G	Switch to and from Airplane/ Standby/Active	data must be aligned with simulator	data must be aligned with simulator	Pass
Physical Cell ID	4G	network Activity Call/ Idle	data must be aligned with simulator	data must be aligned with simulator	Pass
RSRP in LTE	4G	network Activity Call/ Idle	data must be allied with simulator	data must be aligned with simulator	Pass
LTE RLC Data Transfer	4G	network Activity Call/ Idle	data must be aligned with simulator	data must be aligned with simulator	Pass
Modem C-State	2G/3G/4G	network Activity Call/Idle	data must be aligned with simulator	data must be aligned with simulator	Pass

Table 6.2: Modem testing UseCase for RPC based Implementation

Chapter 7

Conclusion

Tools are an important part for analysis of any software, including the operating systems. We have multiple tools which can achieve us brilliance in only one field of analysis. However, optimization and analysis involves many parameters which mean using multiple tools to achieve our goal. Multiple tools can cause inconvenience of time and man power to get it working. Hence the need is to integrate tools and create an automated setup for a more accurate idea to aid analysis.

In this project, we successfully integrated a module which could accurately measure modem power statistics.

The POC of RPC based modem metrics collection has proved that the legacy AT interface for modem communication is far behind when in to high Performance and low Power consumption. The work has given strong numbers for the Intel Android RIL team to argue with the current AT based modem interface.

To conclude, having a tool with the above features gives us an edge over our competitors who design and manufacture their own APs but have limited visibility in the baseband as they procure it from third party vendors. We hope that the proposed tool would be able to give a deeper insight into the platform architecture through simultaneous PnP monitoring of AP and CP via simple enhancement of AT commands. This would enable us to create product differentiation. Also, an efficient PnP data generation would enable productive communication and problem resolution between various teams at Intel, thus helping in faster time-to-market.

Bibliography

[1] Intel 64 and IA-32 Architectures Software Developer's Manual

[2] RPC-RIL Functional Specification (Intel Internal Document)

[3] AT Command Manual.

[4] AP Power Trace Functinality Specification (Intel Internal Document)

[5] XMM7160 Intel Intenal Documents (Intel Internal Document)

[6] 3GPP Technical Specification

[7] AT Commands For GSM & GPRS Wireless Modems

[8] Implementing Remote Procedure Calls by Andrew D. Birrell & Bruce Jay Nelson

[9] SoCWatch for Android* 1.4

[10] SoCWatch High Level Design Document (Intel Internal Document)

[11] Intel HAS SOC Documents (Internet Internal Documents)

[12] Android Developers Manual

[13] National Instrument M Series User Manual

[14] Mobile Device Power Monitor Manual By Monsoon

[15] ACPI Specification