UNIVERSITY OF CALIFORNIA,
IRVINE

Low-Power Firmware Design Techniques for Wearable Emergency Health Monitoring System

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Subramanian Meenakshi Sundaram

Dissertation Committee:
Professor Pai H Chou, Chair
Assistant Professor Mohammad Al Faruque
Professor Rainer Dömer

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
AGC	Automatic Gain Control
AIC	Analog Interface Chip
CODEC	Coder Decoder
DMA	Direct Memory Access
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
ECG	Electrocardiogram
EMS	Emergency Medical Services
FAT	File Allocation Table
FIR	Finite Impulse Response
GPIO	General Purpose Input Output
I2C	Inter Integrated Circuit
I2S	Inter-IC Sound
IIR	Infinite Impulse Response
ISR	Interrupt Service Routinue
LDO	Low DropOut
MCU	Micro controller Unit
MMC	Multi Media Card
NTFS	New Technology File System
PLL	Phase Locked Loop
RTC	Real-Time Clock

RTOS	Real-Time Operating Systems
SAR	Successive Approximation Register
SD	Secure Digital
SDHC	Secure Digital High Capacity
SPI	Serial Peripheral Interface
UART	(Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

ACKNOWLEDGMENTS

I would first like to thank Professor Pai Chou, my advisor, for his continuous support and guidance throughout my Master's Thesis. Without his guidance and persistent help this dissertation would not have been possible. I wish to thank the members of my thesis committee: Professor Mohammad Al Faruque and Professor Rainer Dömer for generously offering their time, support, guidance and good will throughout the preparation and review of this thesis. I offer my warmful thanks to Dr. Ruey-Kang Chang from LA BioMed for his guidance throughout the project accomplishments, and helping us in testing the project. I would also like to thank Eva Villa at LA BioMed for helping conducting the field tests of the device. I would like to extend my thanks to all of my family members, friends, and colleagues, especially Jun Luan, Ph.D., and Rohit Zambre, Ph.D., who have helped and supported me during my Master's program at UC Irvine.

This work was sponsored in part by the NIH STTR Phase I Grants 1R41HL127868-01 ("Asthmagram") and 1R41GM113463-01A1 ("BlueBox") through Neovative, Inc., a Phase II Grant 2R42HL112435-03 ("ECG") through QT Medical, Inc.. The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors.

ABSTRACT OF THE THESIS

Low-Power Firmware Design Techniques for Wearable Emergency Health Monitoring System

By

Subramanian Meenakshi Sundaram

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 2017

Professor Pai H Chou, Chair

Embedded wearable medical systems design has been a continuously developing research field. The requirements associated with wearable devices posts severe restrictions on size and battery capacity. Even though battery technology is improving continously, the battery weight and capacity are the issues that influence heavily on how wearable medical devices can be used. Medical devices often require real-time processing capabilities and this demands high throughput. The available energy resource of these system within the posted constraints doesn't directly meet the demanding requirements. It requires rigorous resource and power management from the firmware to make it feasible. In this thesis we discuss the power constraints of our wearable emergency medical care monitoring device, BlueBox, and our approach in designing low-power firmware for making the optimal use the resource towards fulfilling the demanding requirements.

Chapter 1

Introduction

1.1 Motivation

Emergencies can happen at any time. From minor issues to severe trauma, it is essential that the response teams that come to help them are equipped with the necessary technology to save lives. There is an increasing demand for Emergency medical services (EMS). To respond to this growing demand, the EMS communities require adaptable and sustainable systems. First Responders or paramedics, as they are generally called in United States, provide rapid assessments and treatment of the sick while transporting them to a hospital for further evaluation. Considerable knowledge, skill, and suitable technology are required to provide quality ambulatory emergency medical services. High-quality emergency medical services and paramedics are important parts of any health care system. Paramedics often are first to arrive at the scene, and are allowed to defibrillate, to intubate endotracheally, and to administer life-saving drugs (epinephrine endotracheally, glucose intravenously, etc.). They treat patients during transport in the ambulance. Many studies of pre-hospital services place greater emphasis on technology factors, human efficiency and continuous refinement of standards of practice. EMS providers are expected to render effective life-saving

evaluation, intervention, stabilization and treatment that are consistent with existing standards and codes. Paramedics must be properly trained and suitably equipped with tools and systems to ensure that the highest quality of care, safety and reliability are attained before, during and after a medical emergency event [29, 12, 13, 14].

These situations require responders to effectively care for patients with limited resources and medical infrastructure within the limited space inside the ambulance, often under intense time pressure. For years, emergency medical service providers conducted patient care by manually measuring vital signs, documenting assessments on paper, and communicating over hand-held radios. The ability to automate these tasks could greatly relieve the workload for each responder, increase the quality and quantity of patient care, and more efficiently deliver patients to the hospital. Steady advances in wireless networking, medical sensors, and interoperability software create exciting possibilities for improving the way we provide emergency care. BlueBox is a project that explores and showcases how these advances in technology can be employed to assist patients and paramedics in times of emergency.

1.2 Related Work

1.2.1 Medical Information Tag

Tia Gao et al., developed MiTag (Medical information tag) 1.1, a cost-effective wireless sensor platform that automatically tracks patients throughout each step of the emergency response process, from disaster scenes, to ambulances, to hospitals [20].

The miTags can increase the patient care capacity of responders in the field. The miTag supports sensor add-ons – GPS, pulse oximetery, blood pressure and ECG. The miTag system is out-of-the-box operational and includes the following key technologies: 1) cost-effective sensor

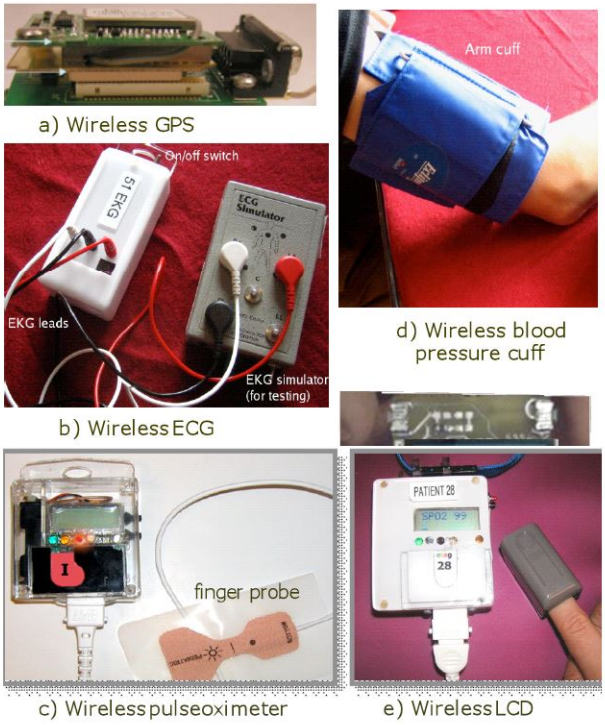


Figure 1.1: Medical Information Tag. [20]

hardware, 2) self-organizing wireless network, and 3) scalable server software that analyzes sensor data and delivers real-time updates to hand-held devices and web portals. The drawback of this setup is its bulky size and uncomfortable usability. The miTag system has wireless blood pressure cuff with an additional 9 V battery to power the cuff which makes it bulky. The miTag uses wireless sensors (ECG, GPS, Pulse oximeter), each of which has its own radio and battery, which makes each individual sensor bulky. This distributed nature of the system adds lot more work for paramedics to manage each one of them individually, which sometimes could be a pain.

1.2.2 Advanced Health and Disaster Aid Network (AID-N)

The Advanced Health and Disaster Aid Network (AID-N), developed by Tia Gao et al. [19] at The Johns Hopkins University Applied Physics Laboratory, explores and showcases how these advances in technology can be employed to assist victims and responders in times of emergency.

AID-N uses open-standard software and best-of-breed hardware to deliver a scalable, open, and reliable infrastructure that paves the way for the development of new capabilities and the extension of existing technologies. Wearable sensors designed by the CodeBlue project at Harvard University is used in AID-N. The AID-N system consists if a pulse oximeter sensor on patient's fingure that measures heart rate and blood oxygenation (SpO_2) level of the patient. It uses Wearable sensors to sense and record vital signs into an electronic patient record database. The vitals are transmitted to a local base staion (a tablet) that displays the vital in real time. Compared to MiTag, this system requires very little setup as it has a single wearable setup. AID-N is integrated with pre-hospital patient care framework MICHAELS. Due to the chaotic nature of emergencies, AID-N system faces the challenge of operating in situations that challenge instrumentation designed for use in the controlled environment of a clinical situation.

1.3 Objective

The miTag and the AID-N systems presented above has challenges when it comes to the usability under a chaotic environment of emergency. Time is a critical resource during emergency and usability issue consumes this critical resource. Also the basic idea of building such devices is to help paramedics provide emergency care while maintaining record of the patients condition. But the above mentioned system does not provide space for paramedics to log about the patients response to their emergency treatment. BlueBox, the proposed system, addresses these issues. We developed BlueBox analogous to Black Box device on airplanes, to be a low-cost, low-power wearable device that automates monitoring patient's vitals and paramedic's voice logs throughout each step of the response process. The idea is to design an easy-to-use wearable hardware system and integrate software solutions to record the patient's vitals and the paramedic's logs in the storage, which allows the doctors and physicians at the hospital to review. Access to this information could allow them to provide better care.

BlueBox platform supports recording of a variety of data including electrocardiogram (ECG), body temperature, respiration pattern (through chest movements monitoring and breathing audio), environment temperature sensors, and the paramedics' speaking voice logs about patient condition. BlueBox relieves the workload for paramedics by automating the task of data recording that helps improve quality of patient care and deliver patients to the hospital with a clear picture of patients condition from the time emergency.

1.4 Thesis Goals

The goal of this thesis is to build low power firmware to realize the functionalities of the BlueBox device within the resource constraints. BlueBox is a wearable, battery-operated, easy-to-use device. Wearable medical devices have strict size and weight constraints while having a functional and behavioral specifications that demand the system to have high performance and function for hours. Battery capacity is directly proportional to its weight as lithium polymer battery tend to have a fixed energy density. Smaller the size and weight, lower the capacity of the battery. Battery capacity is a huge bottleneck for these systems as the system requires to do active sensing and storage for hours. So these systems require an extensive power management strategies to be deployed in hardware and software levels. This problem can be cast as a hardware-software co-design problem.

Hence, in this thesis project, the goal can be stated as follows: to build low-power firmware for BlueBox system so that the device can acquire data at the required frequency and to store the data in local storage while functioning for at least 4 hours.

The methods applied to accomplish the project can be elaborated as the following steps:

- signal requirements and constraints.

- and system design requirements to fulfill the behavioral specification and mechanical constraints.
- design to optimize the required resource and to implement the low-power data acquisition firmware.
- interfacing software for data retrieval and data visualization on the PC.
- conducting experiments with the firmware and the software, along with the BlueBox system.

1.5 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 gives a short overview of the BlueBox system. It also describes the significance and requirements of the signal that needs to be recorded by the BlueBox system. This chapter also presents some design requirement confronted with some constraints.
- Chapter 3 provides the final system overview and presents the hardware architecture. This chapter also talks about the important subsystems, sensors and battery of the BlueBox system.
- Chapter 4 explains the firmware architecture of the BlueBox system and describes its core functionalities. This chapter also describes a procedure for importing and visualizing the acquired data.
- Chapter 5 gives a background on general sources of power consumption and covers in details the power management techniques implemented in BlueBox firmware.
- Chapter 6 presents the system setup, testing and verification of the BlueBox system.
- Chapter 7 concludes this thesis with ideas for future enhancement of the device.

Chapter 2

BlueBox

The previous chapter provided the motivation and objective for a emergency monitoring device. This chapter discusses the design considerations for implementing one. Section 2.1 gives a short overview about the BlueBox system and the requirements of the signal that the system is designed to monitor. Relevant design decision and background are also described in Section 2.2. Section 2.3 elaborates more on the functional requirements of the system by explaining some of the design considerations, requirements, and constraints encountered in the design of BlueBox.

2.1 Overview

BlueBox project is primarily focused on providing a very efficient and easy-to-use wearable device with a short setup time while providing the required functionality to record all the required vitals and data. It could save paramedics a lot of their time which they can invest in providing care for patients. BlueBox as mentioned is a wearable, low-power and low cost device that is supposed to be of size 4 cm $\times$ 6cm.

BlueBox records ECG, Body temperature, chest rise (using accelerometer sensors), and en-

Table 2.1: ECG signal requirements.

Specification	Minimum	Target
Leads	4	4
Channel	1	1
Sampling rate	250 Hz	500 Hz
Resolution	8 bits/sample	16 bits/sample

vironment temperature. It has two microphones where one is a contact microphone to record the patient's breathing pattern and the other is used by paramedics to log the patients condition. The functionality of BlueBox is to record all the data into a persistent storage in the device to be retrieved later for analysis and can be a good indicator for understanding the patient's medical condition and response to emergency treatments before providing further care. BlueBox device can be used when patients are in transit to the hospital in an ambulance or while in the emergency room. The paramedic's logs and the vital signs are synchronized, and such synchronized data streams can provide potentially yield insight about the patients condition from the time of emergency.

2.2 Signal requirements

2.2.1 The Electrocardiogram (ECG)

The ECG has become a routine part of any complete medical evaluation and has been used as a diagnostic test since its discovery over 70 years ago [28]. Because electricity is conducted through the heart muscle (known as the myocardium), many (but not all) types of damage to the heart tissue can be detected with an ECG. The ECG waveform allows one to infer information about electrical activity associated with different aspects of a heart beat and is therefore of particular value for assessing an individual's health. A summary of the electrical specifications relating to ECG quality is presented in Table 2.1.

Table 2.2: Accelerometer signal requirements.

Specification	Minimum	Target
Range	± 4 g	± 8 g
Resolution	8 bits	16 bits
Sampling rate	4 Hz	12 Hz

Table 2.3: Temperature signal requirements.

Specification	Body temperature	Environment temperature
Range	$0-50°C$	$-40-125°C$
Resolution	± 0.1	± 0.1
Sampling rate	2 Hz	2 Hz

2.2.2 Accelerometer

Signal requirements for the accelerometer are not as stringent as that of the ECG. The most important considerations are the acceleration range, resolution, and sampling rate. A ± 4 g range is sufficient to cover the acceleration generated by individuals [17]. A summary of the specification relating to accelerometer is presented in Table 2.2.

2.2.3 Temperature

The continuous monitoring of patients' body temperature is often necessary during induced-hypothermia and general anesthesia, or when employed in the care of infants and premature babies. Intensive care units along with emergency rooms have also adopted patient temperature as a part of vital sign monitoring procedures. There are two different temperatures to be measured: body temperature and the environment temperature, and their requirement is driven by the use case. They still share some common considerations such as the range being measured, resolution, accuracy, and reliability. Table 2.3 summarizes the specification relating to the temperature.

Table 2.4: Audio signal requirements.

Specification	Minimum	Target
Sampling rate	8 KHz	12 KHz
Resolution	16 bits/sample	16 bits/sample

Table 2.5: BlueBox Behavioral Specification.

Specification	Minimum	Target
Usability duration	3 Hours	4 hours
Memory	1 GB	2 GB
Battery	300 mAH	300 mAH

2.2.4 Audio

The system is primarily required to record two audio signals. First, paramedic's log recording feature requires the system to record the human talking voice that has a frequency range between 300 and 3400 Hz. Another audio to record is the patient's respiration/breathing pattern. Table 2.4 depicts the specification applicable to both types of the audio signals.

2.3 System Requirements

2.3.1 Hardware Requirement

This section presents a summary of the desired system specifications. Both minimum and desired specifications are listed. Further details of each specification are provided in Table 2.5.

The signal requirements and the use case provide the system-level electrical requirements. To meet the requirement, ECG is sampled at 500 Hz, the two audio channels are sampled at 12 KHz, the accelerometer is sampled at 12 Hz, and the two temperatures are sampled at 2 Hz. From the target sampling frequency and the resolution of all the signals, audio accounts for 48000 bytes/second, ECG accounts for 2000 bytes of data every second. Accelerometer and temperature data together

sum up to 84 bytes per second. Approximately 50 kilobytes of data is produces every second. For 4 hours, 720 MB of data is generated. The system should have enough storage to save the data and also needs have enough processing bandwidth.

To record for 4 hours, over 720 MB of space is required (not including file system overhead). The data can either be transmitted to a base station, cell phone, or be saved on board. Wireless transmission of the data to the base station is power intensive and also requires the user to have an additional device located nearby all the time. All the data needs to be stored on board, so a high-density memory must be used. A MicroSD card is well suited to this application. However, writing data to a MicroSD card is a power-hungry operation, and latency of write operations are in the hundreds of milliseconds. Most low-power digital signal processors have low on-chip memory, so the system used for the application should have enough memory for buffering data. For a test duration of 4 hours, the system should have a battery capacity of several hundreds of milliamp-hours (mAh) is required. A 300-mAh battery, for example, could support an average current of 75 mA for up to 4 hours at a supply voltage of 3.7 V.

2.3.2 Firmware Requirement

The firmware requirements for BlueBox is that it must be able to sample the signals at the required frequency as mentioned in the previous sections and store them, and it must be able to function for 4 hours. Additionally, the bare-metal firmware must be able to manage the resource to control the system and to keep the power consumption as low as possible. A low-power digital signal processor (DSP) is well suited for this task. The hardware system has to be designed in such a way that the DSP has enough control over all the peripherals and sensors in the hardware in order to perform extensive resource and power management. The low-power DSP must have enough communication resource to communicate with the sensors (accelerometer, SD card, audio codec, etc.), built-in capability for power management (e.g., configurable power domain, clock gating, power

gating, dynamic voltage and frequency scaling), enough debugging opportunities and sufficient on-chip memory for buffering data.

2.3.3 Build-Quality Requirements

The most important requirement is that the device be small in size (40 mm $\times$ 60 mm), robust, comfortable and comply with the medical guidelines. The device should have defibrillator protection feature. Additionally, the device must be splash- and sweat-proof. These requirements for the device must be achieved while maintaining the signal quality requirements as discussed in Section 2.2.

2.4 Design Requirement and Constraint

In general, the requirements associated with a wearable device places severe restrictions on size and power consumption. Even though battery technology is improving continuously and processors are rapidly improving in terms of power consumption, battery life and battery weight are issues that will have a marked influence on how wearable medical devices can be used. Medical devices often require real-time processing capabilities, and thus demand high throughput. Power consumption is becoming the limiting factor in the amount of functionality that can be placed in these devices.

The complexity of embedded wearable systems designs mainly lies in the many considerations that the engineer must take into account. There is no system with unlimited resource available, such that the system can perform extremely well without suffering from resource shortage. On the other hand, there is no system that does not follow the user requirements when it tries to accomplish its tasks. In the field of embedded system designs, engineers are usually constrained by the trade-offs between resource availability and required performance. When it comes to wearable systems, engineers have to take size and power into consideration and make trade-offs between

resource availability, size, power and performance. This introduces challenges into the design. The characteristics and the potentials of a system are determined by the resources it has. System requirements are always to be fulfilled by the designers. These requirements are to be satisfied if the system is desired to function as it should. With a limited amount of resources provided by the system, an engineer has to think carefully about how to use the existing resources in such a way that they can meet the need to carry out the required tasks.

In general, wearable embedded systems design is a hardware-software co-design problem as the system optimization consists of scheduling and binding [16]. Scheduling has to be done to meet the deadline of each task that has to be accomplished by the system, while binding has to be done to meet the availability of resources on the system. In other words, scheduling is related to performance while binding is related to resource. For battery operated devices, power management becomes an important aspect of functional requirement and software design, as it heavily influences schedulability and binding. Functional requirements, including power management, and the way it is planned to be accomplished with the software, provide the necessary information about schedulability and binding for the design optimization. Thus requiring a hardware-software co-design.

The main problem of such design is to find the most efficient combination of resource usage, optimal performance and battery capacity. The basic rule is to use the available resources as efficiently as possible, while trying to achieve, at least, the required performance, i.e., latency or execution time under the power budget. This means all deadlines have to be met with whatever resource existing on the system with minimum power consumption.

As mentioned in the previous section, BlueBox has an size and weight constraint, which is considered during the design to optimize the resources of the system to fulfill the constraints. It is required to run for a minimum of four hours on battery and is optimized to have the minimum required resources to acquire, process and store the large number of high frequency signals, mentioned in the previous section, with the required performance. Battery capacity is directly

Table 2.6: Typical power consumption of components

Component	Typical power consumption
TMS320C5515 DSP CPU	100 mW
SDHC card, 50 MHz @ 3.3 V	495 mW
AIC3204 Audio Codec	21.9 mW
ADS1292R ECG front end	1.4 mW
MPU9250 Accelerometer Sensor	12.21 mW
Environment temperature sensor	1 mW
Microphone	3.3 mW
Body temperature sensor	0.2 mW

proportional to its weight as lithium polymer batteries tend to have a fixed energy density. The battery consideration is primarily based on its size, weight, and heat dissipation. So, here the battery capacity is traded off to meet the size and medical guidelines.

Table 2.6 lists the major components used in BlueBox and their power consumption. The best battery that fulfill the size constraint has capacity of 300 mAh @ 3.7 V, which is equivalent to 1.11 Wh of energy. The theoretical power consumption of system sums up to 635 mW approximately. At this rate, the system will deplete the battery in approximately 100 minutes.

From the above calculations it is evident that energy is a scarce resource in this system and that it is very important to do rigorous power management to extend the battery life to meet the requirement. So, BlueBox with a limited power budget posts the interesting challenge of optimizing the power consumption of the device while meeting all deadlines fulfilling the functionalities with the constrained resources on the system and keep the device running for a minimum of two hours.

The thesis proposes solutions to the above mentioned problem statements with a structured approach to extensive power management through the firmware. The further chapters first give the general context of the hardware and software architecture, and then dive into the proposed power management techniques.

Chapter 3

Hardware Architecture

This chapter first provides an overview of the architecture of the BlueBox system, followed by all the components and peripherals, including sensors, used in the system and their significance. The purpose is to give a foundational understanding of the architecture necessary for designing and architecting the firmware.

3.1 System Overview

The BlueBox prototype consists of a central board, processor, storage element, contact and environmental microphones, accelerometer, and the body and environment temperature sensors. The additional electrodes attach to the central board via a DB-9 connector. The contact microphone is attached through a standard 2.5 mm audio jack. The body temperature sensor is attached through solder pads on the board. Figure 3.1 shows the overall system architecture for BlueBox.

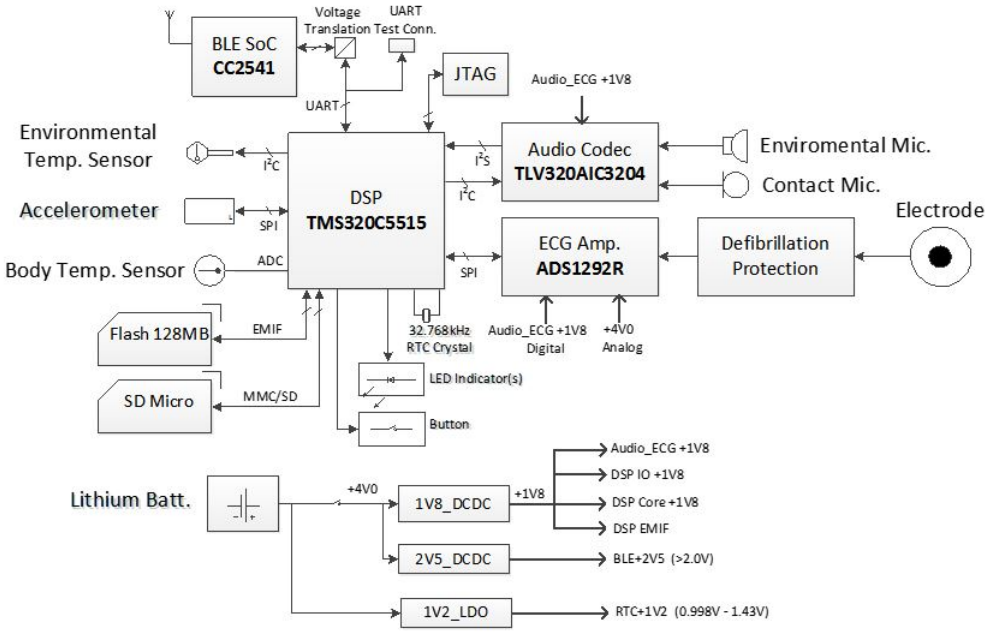


Figure 3.1: BlueBox Hardware System Architecture

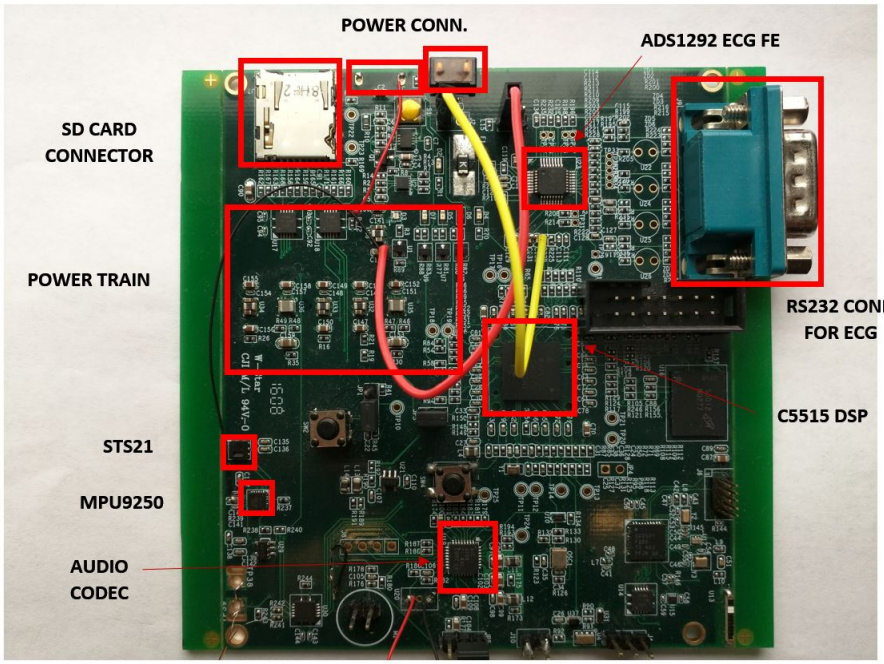


Figure 3.2: BlueBox Prototype Hardware

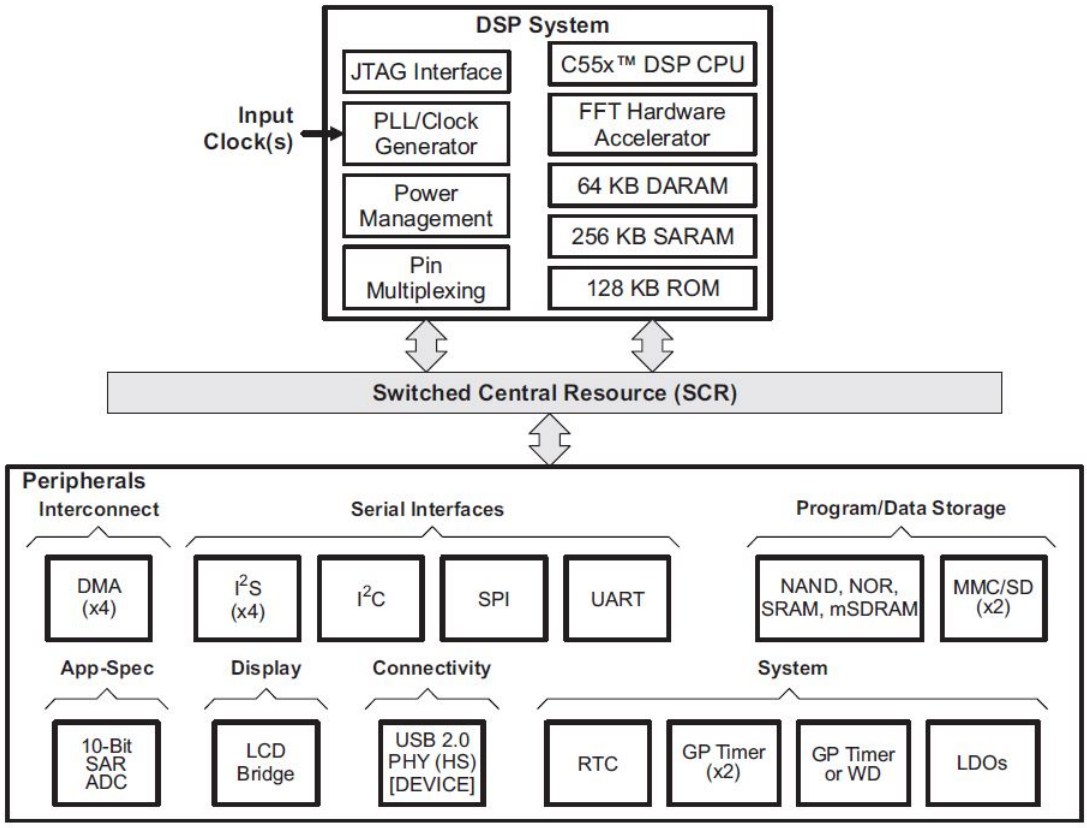


Figure 3.3: DSP Architecture. [7]

3.2 Digital Signal Processor

TMS320C5515 is a fixed-point DSP based on the TMS320C55x CPU processor core, or C55x for short. This DSP architecture achieves high performance and low power through increased parallelism and total focus on power savings. Figure 3.3 shows the DSP architecture.

3.2.1 Clock Frequency

The DSP has a low power software programmable Phase Locked Loop (PLL) clock generator that supports 60-, 75-, 100-, and 120-MHz clock rate.

3.2.2 On-Chip Memory

The DSP has 320K bytes of zero-wait-state on-chip RAM and 128K of on-chip ROM.

3.2.3 Peripherals

The DSP has general-purpose input and output functions along with the 10-bit Successive approximation register (SAR) Analog to Digital converter (ADC). Together, they provide sufficient pins for status, interrupts, and bit I/O for LCD displays, keyboards, and media interfaces. Serial media is supported through two Multimedia Card/Secure Digital (MMC/SD) peripherals, four Inter-IC Sound (I2S bus) modules, one Serial Peripheral Interface (SPI) with up to 4 chip selects, one I^2C multi-master and slave interface, and a Universal Asynchronous Receiver/Transmitter (UART) interface.

3.2.4 DMA

The device includes four Direct Memory Access (DMA) controllers, each with 4 channels, providing data movement for 16-independent channel contexts without CPU intervention. Each DMA controller can perform one 32-bit data transfer per cycle, in parallel and independent of the CPU activity.

3.2.5 Power

Furthermore, the device includes three integrated low-dropout (LDO) regulators (DSP LDO, ANA LDO, and USB LDO) to power different sections of the device. The DSP LDO can provide 1.3 V or 1.05 V to the DSP core (CVDD), selectable on-the-fly by software as long as operating frequency

ranges are observed.

3.3 AIC3204 : Audio Codec

TLV320AIC3204 (or AIC3204 for short) is an analog interface chip (AIC) from Texas Instruments. It is a flexible, low-power, low-voltage stereo audio codec with programmable inputs and outputs, PowerTune capabilities, fixed predefined and parameterizable signal-processing blocks, integrated PLL, integrated LDOs, and flexible digital interfaces. It has both recording and playback capabilities. The record part of the AIC3204 covers operations from 8 kHz mono to 192 kHz stereo recording, and contains programmable input channel configurations covering single-ended and differential setups, as well as floating or mixing input signals. It also includes a digitally-controlled stereo microphone preamplifier and integrated microphone bias. It integrates A/D and D/A conversion functions and achieved high-precision A/D and D/A converter in low cost using Σ-Δ technology. The chip is small and compact and is available in 5 mm $\times$ 5 mm 32-pin QFN package.

Figure 3.4 displays the audio codec's architecture. It has capabilities of recording stereo audio signals. It has two channels that it can record. One channel is used for respiration recording and the other for paramedic's log recording. Each channel of the stereo audio ADC consists of a signal-processing engine with fixed processing blocks. The signal processing blocks available are first-order infinite impulse response (IIR), scalable number of bi-quad filters, variable-tap finite impulse response (FIR) filter and automatic gain control (AGC). The choice between these processing blocks is part of the PowerTune strategy to balance power conservation and signal-processing flexibility. Less signal-processing capability reduces the power consumed by the device.

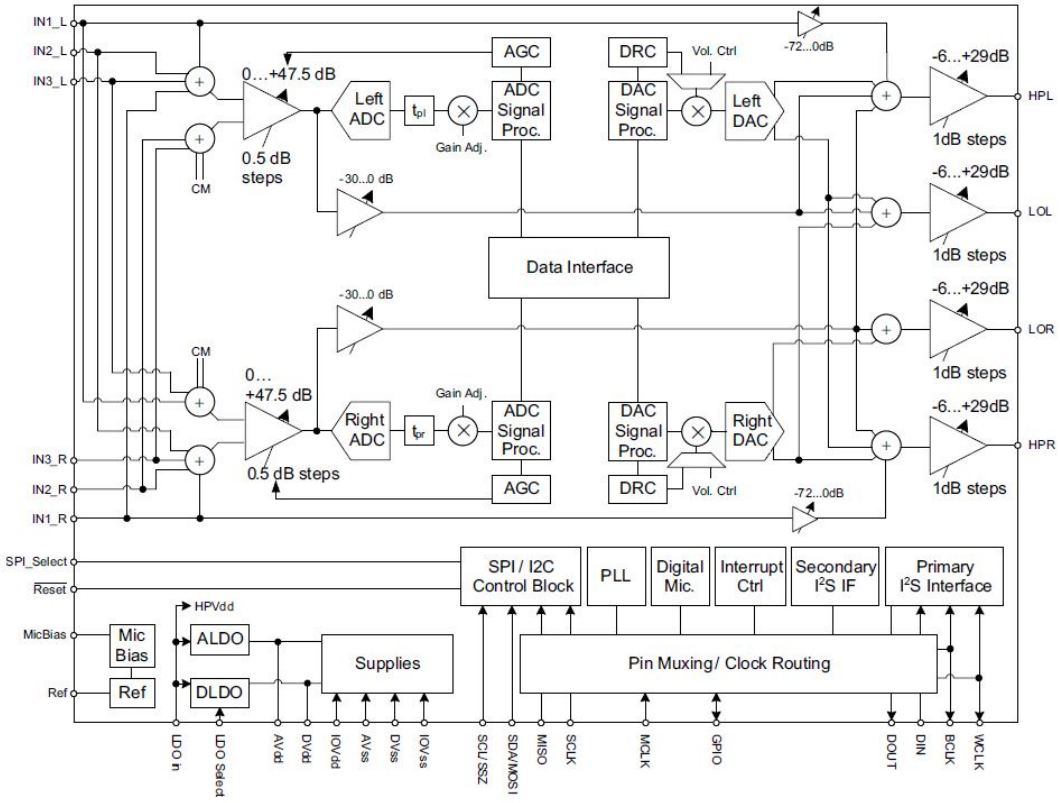


Figure 3.4: AIC3204 Block Diagram. [9]

3.4 ADS1292R : ECG Frontend

ADS1292R is the ECG analog front end chip to which the electrode output is connected to. ADS1292R is a two channel, simultaneous sampling, 24-bit, delta-sigma ($\Delta\Sigma$) analog-to-digital converters (ADCs) with a built-in programmable gain amplifier (PGA), internal reference, and an on-board oscillator. ADS1292R incorporate all features commonly required in portable, low-power medical electrocardiogram (ECG), sports, and fitness applications. ADS1292R enables the creation of scalable medical instrumentation systems at significantly reduced size, power, and overall cost. ADS1292R has a flexible input multiplexer per channel that can be independently connected to the internally generated signals for test, temperature, and lead-off detection. Additionally, any configuration of input channels can be selected for derivation of the right leg drive (RLD) output signal. The ADS1292R operate at data rates from 500 SPS up to 8 kSPS. The device is packaged in a 5-mm $\times$ 5-mm, 32-pin thin quad flat pack (TQFP). Operating temperature is specified from -40°C to $+85^{\circ}$C. ADS1292R is interfaced with the DSP through SPI.

3.5 MPU9250 : Accelerometer sensor

MPU-9250 is a 9-axis motion-tracking device that combines a triaxial gyroscope, triaxial accelerometer, and a triaxial magnetometer with a Digital Motion Processor^{TM} (DMP). With its dedicated I^2C sensor bus, the MPU-9250 directly provides complete 9-axis MotionFusion^{TM} output. MPU-9250 features three dedicated 16-bit analog-to-digital converters (ADC) for digitizing each of the gyroscope outputs, accelerometer outputs, and magnetometer outputs. For precision tracking of both fast and slow motions MPU-9250 provides a user-programmable accelerometer full-scale range of $\pm$2 g, $\pm$4 g, $\pm$8 g, and $\pm$16 g. The package size down to a footprint and thickness of $3 \times 3 \times 1$ mm, to provide a very small yet high-performance low-cost package. Figure 3.6 shows the architecture of the MPU9250.A valid accelerometer data is available 30 ms (typical)

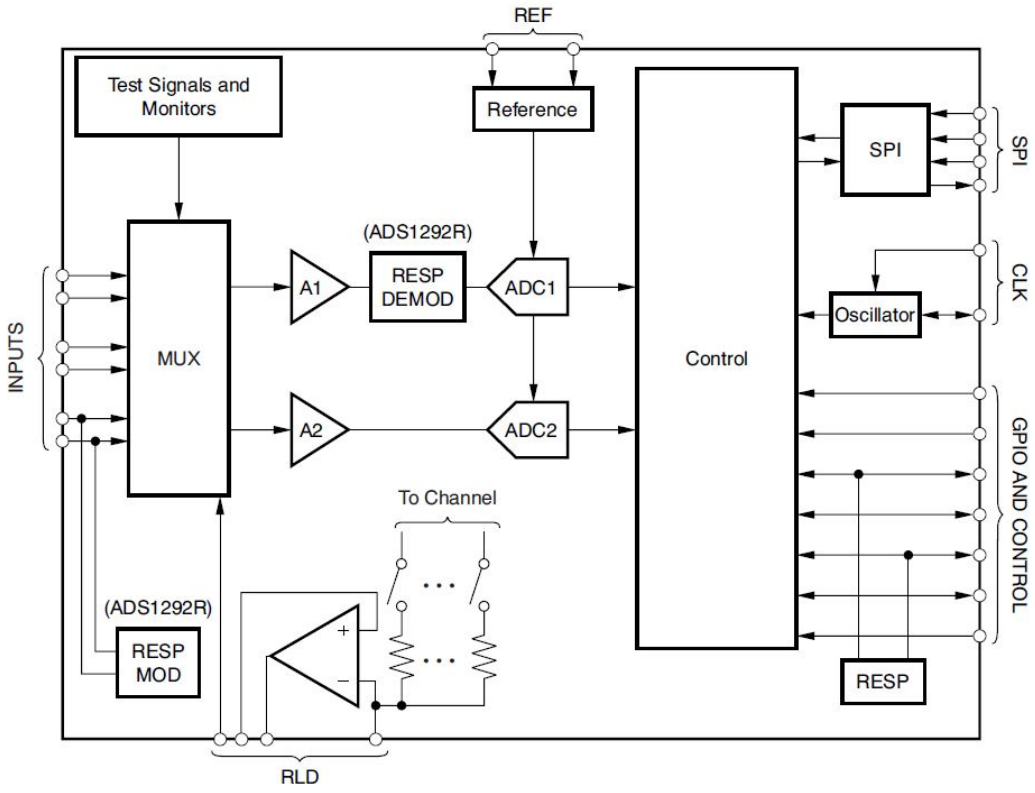


Figure 3.5: ADS1292R Functional Block Diagram. [4]

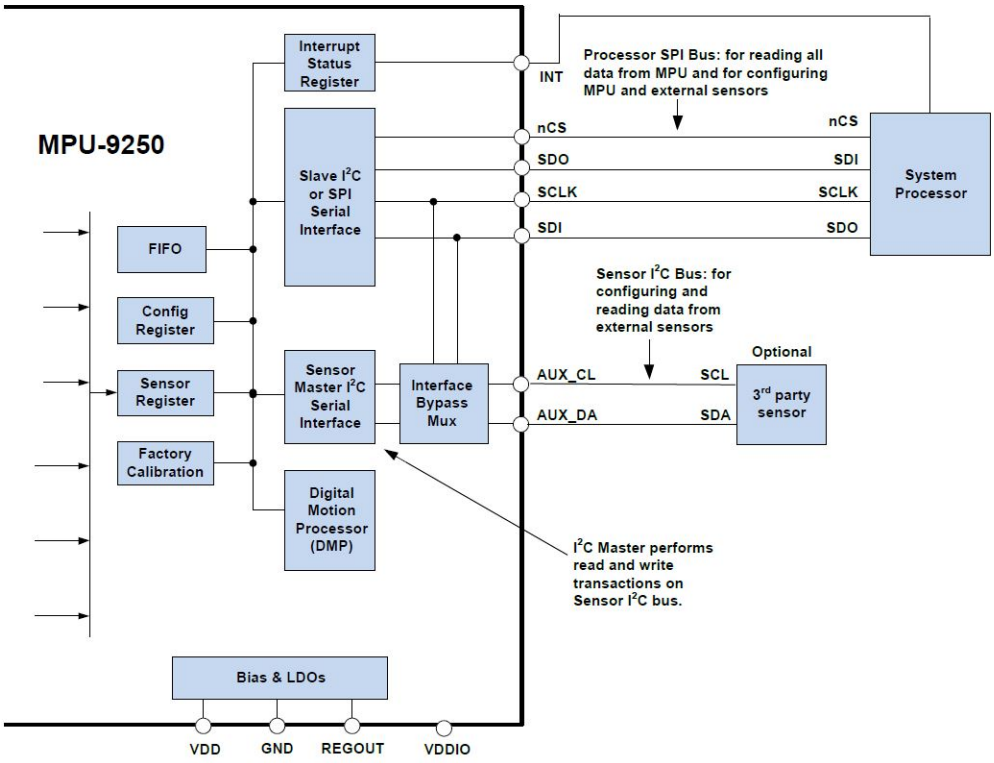


Figure 3.6: MPU9250 Architecture. [8]

after wakeup from sleep.

The MPU-9250 contains a 512-byte FIFO register that is accessible via the Serial Interface. The FIFO configuration register determines which data is written into the FIFO. Possible choices include gyro data, accelerometer data, temperature readings, auxiliary sensor readings, and FSYNC input. A FIFO counter keeps track of how many bytes of valid data are contained in the FIFO. The FIFO register supports burst reads. The interrupt function may be used to determine when new data is available.

The MPU9250 has multiple user-configurable power modes. The modes of interest are Sleep mode at 8 μA of power consumption and Low-Power Accelerometer Mode, which consumes 450 μA.

3.6 Temperature sensor

3.6.1 STS21 : Environment temperature sensor

STS21 is a low-power, fully self-calibrated digital temperature sensor well suited for applications with high demand on temperature accuracy. With dedicated I^2C bus, it directly provides the temperature output. The sensor comes in a package sized 3×3 mm footprint and 1.1 mm height. The sensor is powered up to the chosen supply voltage VDD (between 2.1 V and 3.6 V). After power-up, the sensor needs at most 15 ms for reaching idle state, i.e., to be ready accepting commands from the DSP. Current consumption during start up is 350 μA maximum. Whenever the sensor is powered up, but not performing a measurement or communicating, it is automatically in sleep mode (idle state). STS21 measures temperature with different resolutions ranging from 14 bits to 11 bits, where the 14-bit resolution takes 85 ms for conversion whereas a 11-bit resolution takes somewhere between 9 and 11 ms.

3.6.2 MA100: Body temperature sensor

MA100 is a Biomedical Chip Thermistor assemblies that are designed for use in applications involving both intermittent and continuous patient temperature monitoring. Although low in cost and small in size, these are highly stable, precision thermo-chips provide reliability, tight interchangeable tolerances, geometries, and fast response time that are often required.

3.7 Memory

The BlueBox system has on-chip memory and the persistent Micro-SDHC (Secure Digital High Capacity) card storage. Because a MicroSD card is power intensive and takes hundreds of mil-

Figure 3.7: Lithium Polymer battery

liseconds to write to, the DSP's on-chip memory is used as buffer to store data intermittently.

Once there is enough data in the buffer it is written to the SD card and later the data can be accessed by a computer. All the heavy-lifting data transfers are done with the help of Direct Memory Access (DMA) peripheral interconnect to take load off the DSP. SD card is interfaced to DSP using MMC/SD interface. The SD card used for the application is a high capacity card that can support a maximum clock frequency of 50 MHz. These high capacity cards are not byte addressable. They are organized in 512-byte sector or block and thus are block addressable. BlueBox uses Kingston 8-GB Micro SDHC card. The SDHC card is very power hungry. It consumes 150 mA when operated at 50 MHz during active read/write operation and consumes 30 mA during inactive periods.

3.8 Battery and Power management circuitry

A GMB 302547 Lithium ion polymer rechargeable battery, shown in Figure 3.7, is used to power the system. It has a capacity of 300 mAH at 3.7 V. It is 4 cm long, 2 cm wide and 0.2 mm thick, and it weighs 8 grams. The battery is placed on top of the central board housing. The central board has a battery charging circuitry that can charge the battery completely in 2 hours. The power management circuitry provides a power-train that feeds the different power domain of the DSP as shown in Figure 3.1.

Chapter 4

Firmware Architecture

This chapter describes the firmware architecture of the BlueBox system and describes its core functionalities. This chapter also talks about importing and visualizing the acquired data.

4.1 Overview

The firmware for the system is written in C and assembly using Code Composer Studio IDE. TI provides DSP/BIOS (in later versions called SYS/BIOS) a real-time operating system for their DSPs. This operating system provides a wide range of system services to an embedded application including preemptive multitasking, memory management, and real-time analysis. The RTOS is feature-rich and helps speed up development and testing of firmware. However, a potential drawback of using the RTOS is that it trades performance for structure of the DSP software for quicker and easier development. For a resource and power constrained application, it is required to squeeze out the performance from the system for firmware to have complete control over the underlying hardware. Thus, firmware for the system is implemented on bare metal using C and using the chip support library (CSL) for C5515 provided by TI.

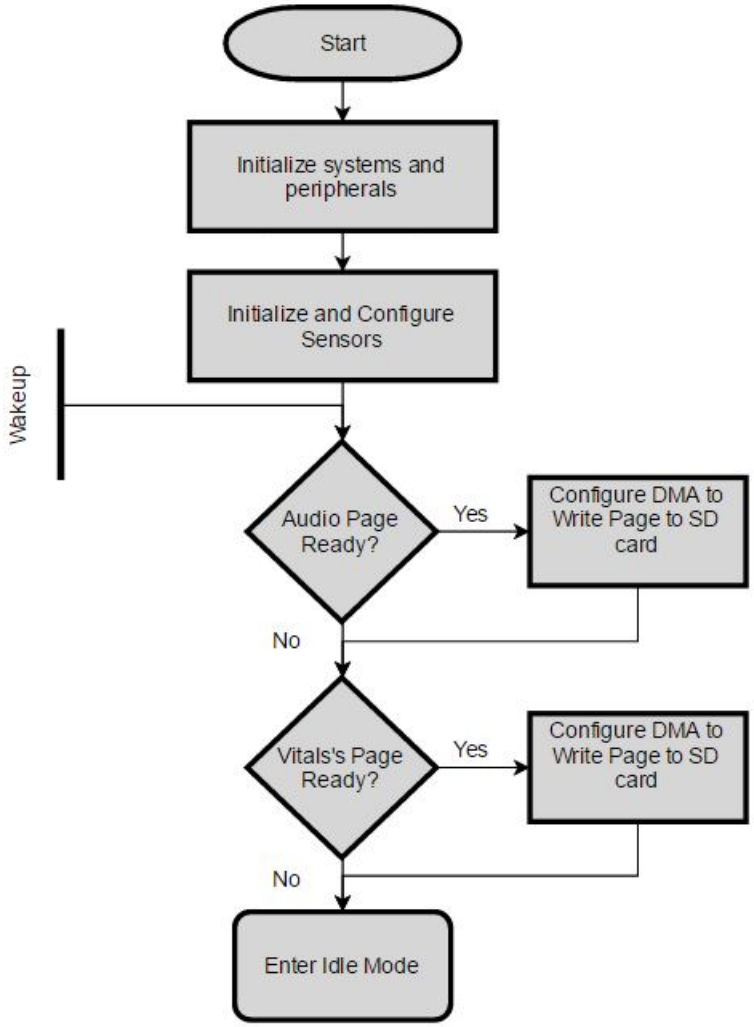


Figure 4.1: Main thread

The functions of firmware can be categorized as follows:

- Initializing System and Sensors
- Data collection and storage
- Power management

Figure 4.1 gives the overall flow of the firmware.

4.2 Initializing System

The first thing the firmware does after boot-up is to initialize the clock by configuring the phase locked loop (PLL) control registers. PLL is configured to generate a 100 MHz clock to tick the system. After the clocks have been set up, the multiplexed I/O pins are configured to function as a specific peripheral or general-purpose input/output (GPIO) pins. Then, the clock and power to those I/O pins and peripherals are enabled. After the I/O pins are configured, the communication peripherals such as SPI, I^2C, and I2S are configured. Their mode of operation (master or slave), frequency of operation, etc., are initialized and configured. DSP's I2S module is configured as I2S master (master supplies bit clock and word clock) and the Audio codec AIC3204 is configured as I2S slave.

4.3 Initializing sensor

Audio Codec AIC3204 is configured, through the I2C interface, to sample the audio signal at 12 KHz on each of the two channels and configured to amplify the signal with automatic gain correction feature enabled. The codec is also configured to filter the audio signal using first order IIR filter and 5 Bi-quad filters to improve the audio quality.

The ECG analog front-end ADS1292R is configured to sample ECG signal at 500 samples per second. ADS1292R is configured to generate a hardware interrupt to inform the DSP when the sample is ready so that the DSP can read the data from the ECG chip over SPI. The environment temperature sensor STS21 is configured to sample the temperature with an 11-bit precision. This data is acquired by polling. For every 250 ECG interrupts the STS21 is polled once to collect the temperature data. The Analog-to-Digital converters (ADC) on board are configured to sample the body temperature once every 250 interrupts of ECG.

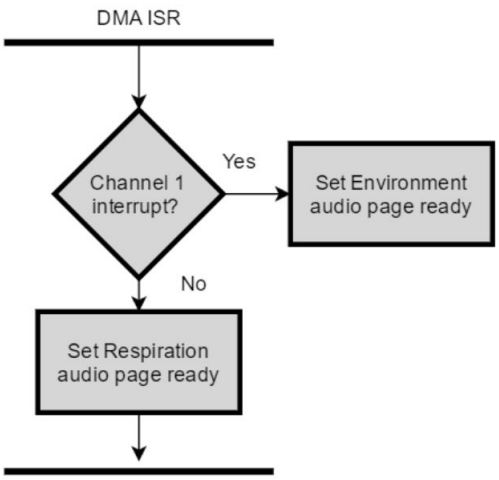


Figure 4.2: DMA Interrupt Service Routine

4.4 Data Collection and storage

The firmware is designed for concurrent data acquisition using dedicated hardware resources for the signals of interest. These hardware resources can perform certain required but very defined functionalities like data buffering, filtering, and data moving without intervention by the DSP. These hardware accelerators requires the DSP to configure, for them to operate. The designed firmware is primarily preemption based where the DSP configures, assigns work, to the peripherals and hardware accelerators to concurrently acquire the signals. Upon completion of the assigned work, i.e., data acquisition, these accelerators interrupt the DSP, which configures DMA to write the collected data in the form of audio and vitals pages (explained in Section 5.7.2) into the SD card. This sequence of interrupts and services happen periodically to acquire and store the signals.

There are two non-maskable interrupts that preempts the DSP. One is the DMA interrupt, generated upon acquisition of audio signal of pre-configured size and other one is the hardware interrupt generated by the ECG front end signaling that the ECG data is ready. The flowcharts in Figure 4.2 and Figure 4.3 give the functionality of DMA interrupt service routine (ISR) and the hardware interrupt ISR, respectively. The ECG ISR is responsible of acquiring the available

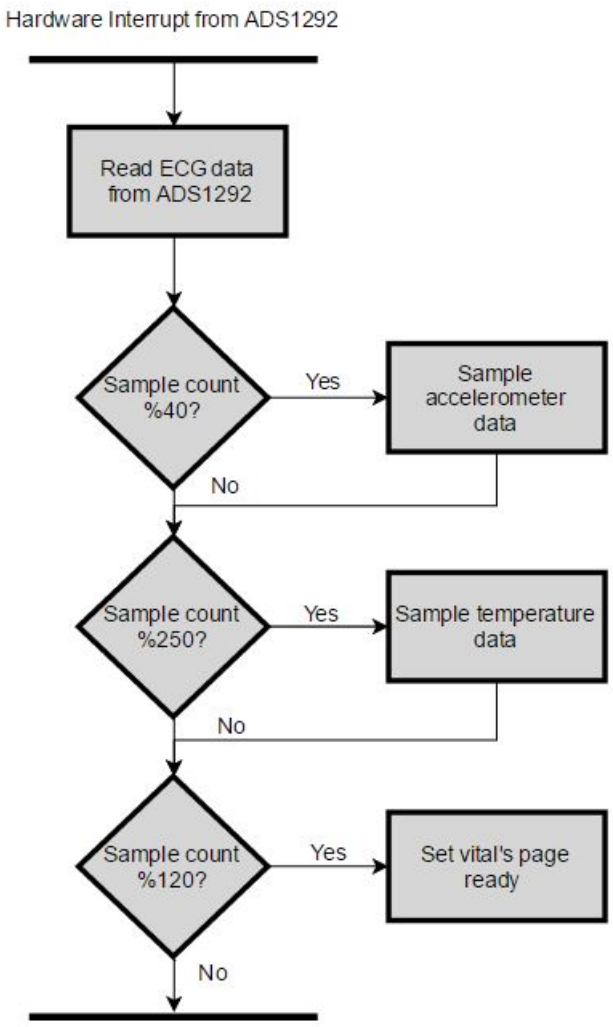


Figure 4.3: Hardware Interrupt Service Routine

ECG data from the Analog front end and also for acquiring the body temperature, environment temperature, and accelerometer data.

The following would provide insight and explain in detail the data storage mechanism with the context of audio signal acquisition. The two audio channels are sampled at 12 KHz. The audio signal acquisition and storage is the most intensive task of the entire system and also very power hungry. Involving DSP to do this complete task would completely utilize the DSP's computational bandwidth while allowing no space for DSP to do other useful work. Also, the overhead of doing

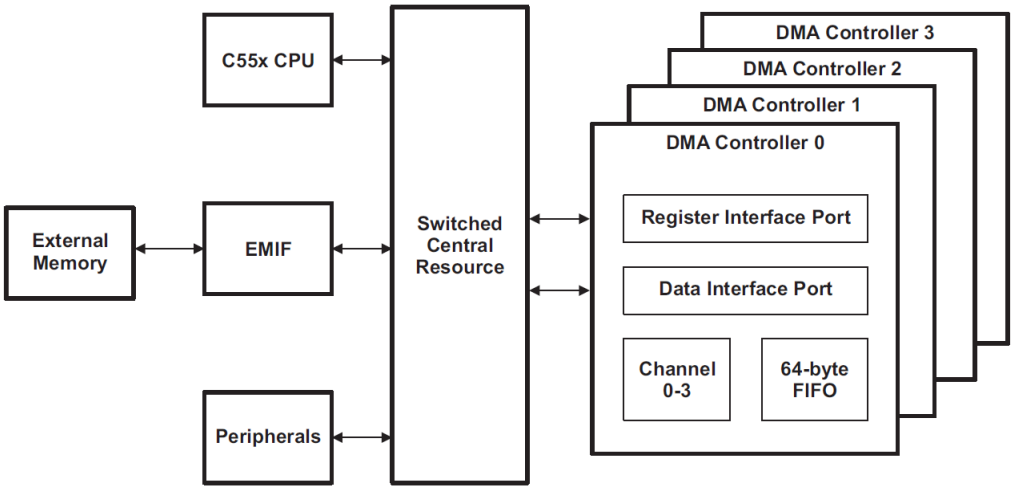


Figure 4.4: Conceptual Block Diagram of DMA controller. [5]

this operation through DSP is very high. TMS320C5515 includes 4 DMA controllers with four DMA channels each. It can move data among internal memory, external memory, and peripherals without intervention from the DSP and in the background of DSP operation. So, DMA data transfer is used in audio signal collection and storing. Figure 4.4 shows the DMA in the DSP.

The DMA is configured to collect the audio signal from the I2S peripheral and cache it in a buffer. The I2S receive event triggers the DMA to collect the audio signal. When the DMA fills up the allocated buffer, it interrupts the DSP to initiate the SD card write using another available DMA channel, and it takes care of the further action. The system continues acquiring audio signals even while writing data into the SD card. This is done through using DMA in ping-pong mode. Figure 4.5 shows the ping-pong operation mode.

When the ping buffer data is being copied to the SD card , the DMA acquires the incoming data and fills it in the pong buffer. So DMA used in ping-pong mode enables the system to collect the audio signals and store the previously cached signal simultaneously.

As mentioned earlier, the measurement of temperature signal is triggered for every 500 interrupts of ECG. This adds an overhead of checking if the interrupt counts every time the ECG

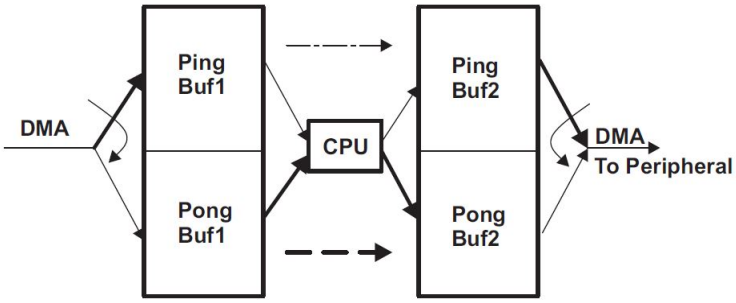


Figure 4.5: Ping-Pong Mode for DMA Data Transfer. [5]

Figure 4.6: Flow of data in during record

interrupt is serviced. The measurement of temperature within the context of ECG interrupt increases the worst-case execution time of the ISR, which is not preferred in general for typical application. One could argue that the system could have used a timer and generate a timer interrupt every one second and trigger the temperature measurement instead of tightly coupling it with the ECG ISR. The reason for not going with this approach is the cost in terms of power for running a dedicated timer for the temperature measurement. As long as the tasks could be finished on available clock cycles without disturbing other functionality of the system, it is fine even if the worst-case execution time of the ISR increases.

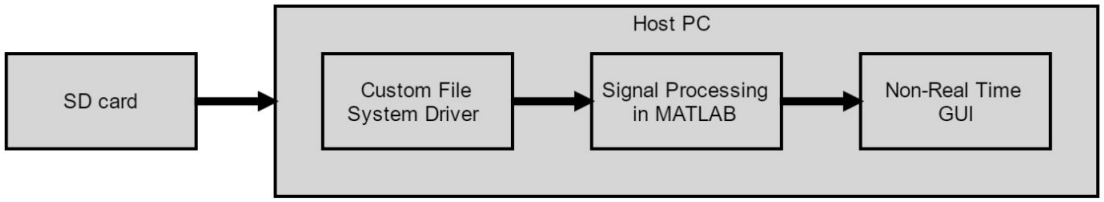


Figure 4.7: Flow of data during import

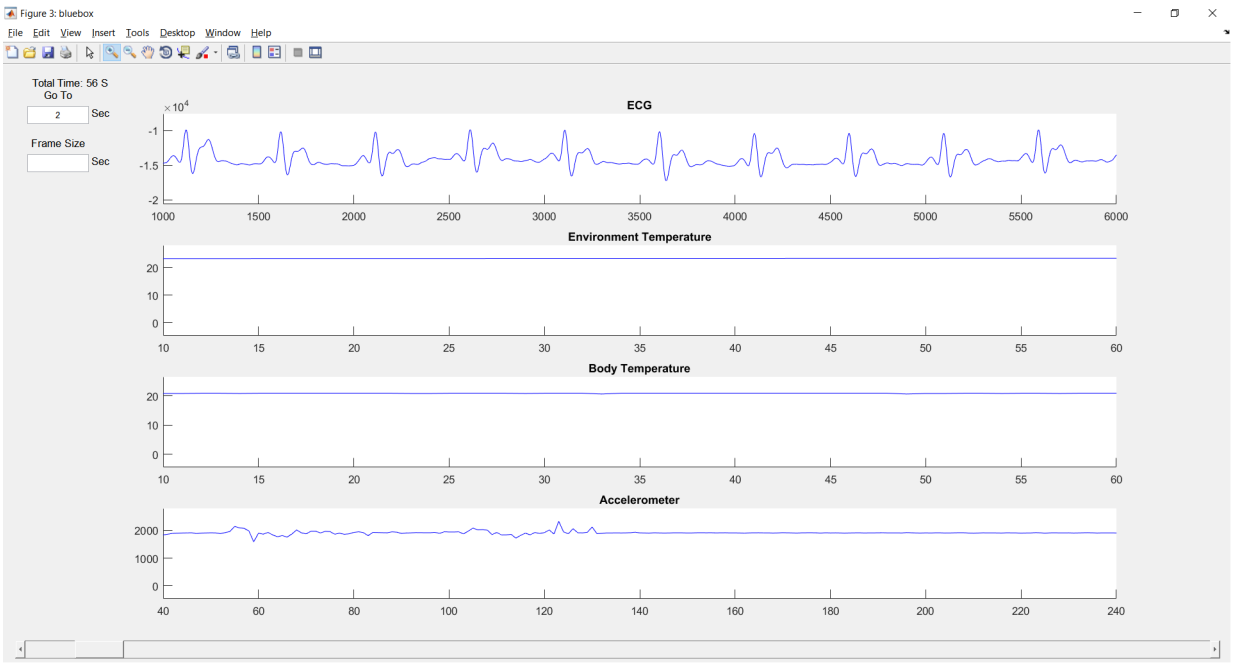


Figure 4.8: MATLAB Graphic User Interface

4.4.1 Importing the Data

Once the data has been saved to the Micro SD card, it can be read directly by a computer. A custom file system is used to store the data, so it cannot be directly recognized on a computer as one of the supported file systems. A filesystem driver for the custom filesystem is developed in Python, which could parse the data on the SD card and create files of each of the signals. These files are read by a MATLAB script that does post processing and generates data vectors corresponding to each signal. For example, in case of the body temperature sensor, the data in the file is the raw ADC values, the calibration details of the body temperature is embedded into MATLAB code, so it does convert raw ADC temperature data into a meaningful value through the calibration information.

MATLAB plots the data and visualizes it as shown in Figure 4.8 so that the doctors and physicians can relate and see all the signals together. The tool also allows them to jump to the time of interest and tune the time window shown by the tool.

Chapter 5

Low-Power Firmware Design Techniques

Chapter 5 first gives a background on general sources of power consumption in a digital system and describes some of the basic techniques used for power management. Section 5.4 describes the design flow of power management techniques for BlueBox system. Section 5.5 to 5.7 explains the details of those techniques.

5.1 Background

In order to design successfully for low power consumption, it is important to have a through understanding of the sources of power dissipation, the factors that affect them and the methodologies and techniques that are available to achieve optimal results. Throughout this chapter, we discuss about power consumption and methods for reducing it. Energy is the time integral of power; if power consumption is a constant, energy consumption is simply power multiplied by the time during which it is consumed. Reducing power consumption only saves energy if the time required to accomplish the task does not increase too much. The chapter presents the sources of power dissipation and describes the important low-power methodologies and power optimization techniques

available. Low-power design can be applied on various abstracted levels, such as system level, architectural level and technological level. This chapter could as well be utilized as a quick review on low power design.

5.2 Sources of power consumption

Most components are currently fabricated using CMOS technology[21, 26]. Main reason for this bias is that CMOS technology is cost efficient and inherently lower power than other technologies. So the sources of power consumption talked in this chapter is based on CMOS component model. There are a number of sources of power consumption in CMOS, which can be subdivided into *static* and *dynamic* power dissipation. The main difference between them is that dynamic power is frequency dependent, while static is not. Dynamic power dissipation is primarily caused by the switching of the CMOS devices (MOSFETs) when logic values are changed (known as *capacitive power* or *switching power*). The amount of power that is dissipated is directly related to the switching activity (which is the number of logic transitions per clock cycle in the entire circuit), the clock frequency, the supply voltage, and the capacitive load the circuit drives [27]. Another source of dynamic power dissipation is *short-circuit power*. CMOS is comprised of both PMOS and NMOS devices. During a logic transition, the PMOS and NMOS devices are simultaneously turned on for a very short period of time, allowing a short-circuit current to run from V_{dd} to ground[15, 30]. This behavior is inherent to CMOS switching. The more dominant component of dynamic power is capacitive power. This component is the result of charging and discharging parasitic capacitances in the circuit. Every time a capacitive node switches from ground to V_{dd} an vice-versa energy is consumed.

Static power, which is a constant factor and has nothing to do with the switching activity, is caused by *leakage power*. In an ideal situation, a static CMOS circuit (i.e. one that does not switch) does not consume any power because there is no direct path from V_{dd} to ground. In real life there

will always be leakage currents, since CMOS devices are not perfect switches. The total power dissipation can be described by the following formula:

$$P_{total} = \alpha(C_L.V_{DD}^2.f_{clk}) + I_{sc}.V_{DD} + I_{leak}.V_{DD} \qquad (1)$$

The Greek letter α represents the *switching activity* in the circuit, expressed in a value between 0 (no switching activity at all) to 1(maximum switching activity). C_L is the capacitive load, driven by the circuit. As can been seen in the formula, the dynamic power consumption depends on V_{dd} square, which makes the supply voltage a very important factor in low-power design, as will be explained later. I_{SC} and I_{leak} represent the total short circuit and leakage current, respectively. It is important to realize that that I_{SC} is a variable, while I_{leak} is not. I_{SC} depends on the charge carried by the short-circuit per transition, the cycle time, and the total number of transitions

$$I_{SC} = Q_{SC} \cdot f \cdot \alpha \qquad (2)$$

A better way to represent the formula would be like this:

$$P_{total} = \alpha(C_L \cdot V_{DD}^2 \cdot f_{clk}) + Q_{SC} \cdot f_{clk} \cdot V_{DD} + I_{leak} \cdot V_{DD} \qquad (3)$$

It is a misunderstanding that reducing power consumption will always lead to a reduction in energy as well. When we refer to power, we refer to the momentary electric energy that is dissipated, measured in Watts. Energy is measured in Joules, or Watts per second. Thus, the energy consumption depends on the power consumption and the time it takes to perform a task. For example, if we have two circuits A and B, and $P_A = 2P_B$, and $2D_A = D_B$ (where D refers to the delay of the circuits), then EA = EB. Thus, even though the power consumption of circuit B is twice as low as circuit As, no energy is saved. When we take another look at formula (3), reducing α, V_{DD}, and C_L will always reduce power and energy consumption. Lowering f_{clk} reduces only power, not energy [18].

5.3 Basic Low-Power Design Methodologies

The following methodologies are most powerful ones and are the most basic ones for any system. They include voltage scaling, frequency scaling, clock gating and power gating. All these methodologies helps to reduce the power consumption but comes at a cost. For example frequency scaling could affect the execution time of the application, power gating causes a boot up latency whenever the system or peripheral needs to be powered on. It purely depends on the application requirement to decide on the trade-off If the application can tolerate the cost of a methodology, then it can be used to reduce the power consumption. So, the BlueBox system adopts a few of these basic methodologies but not all of those. It is still important to have a basic idea of these basic methodologies and the insight they provide on power consumption.

5.3.1 Static voltage scaling

One way to decrease the power consumption significantly, is to decrease the supply Voltage. As mentioned in the previous section 5.2, dynamic power consumption depends quadratically on V_{DD}. Voltage scaling is therefore the most effective method to limit the power consumption. However, when V_{DD} is lowered, it comes at a price: the delay of the logic increases. In systems where we desire a high throughput, and ask for the maximum performance of the technology being utilized, voltage scaling is not an option. If V_{DD} would be lowered, we would not be able to meet the performance requirements. In many situations however, we do not ask for the maximum performance and we can safely lower V_{DD} in order to save power. Even though delay is increasing, the power-delay product is improving when V_{DD} is decreased, since power decreases quadratically while delay increases less fast. Delay scales with:

$$\frac{V_{DD}}{(V_{DD} - V_{TH})^2} \qquad (4)$$

5.3.2 Dynamic voltage scaling

Instead of a fixed supply voltage during circuit operation it is also possible to dynamically adjust the supply voltage based on the current required performance of the circuit. The required performance is often not always the same. When the required performance of the circuit is momentarily reduced, we can afford lower supply voltages. This is called dynamic voltage scaling (DVS). Apart from lowering the supply voltage, it is also possible to lower the clock frequency. A reduction in V_{DD} will always increase the delay to some extent, but if the cycle time is still much higher than the delay of the circuit, energy is wasted. Therefore dynamic voltage and frequency scaling (DVFS) can be employed to save energy to the maximum.

5.3.3 Frequency Scaling

Apart from V_{DD}, there is another variable in the equation of section 5.2 that intuitively suggests possibilities for reducing power: frequency. Of course, the clock frequency is bound by the desired throughput of the system. However, a system does rarely operate at its maximum throughput all the time. Often, the desired throughput is much lower than its maximum performance. Then, it is possible to lower the operating frequency in order to save power (and thus also energy), which is called frequency scaling. Sometimes voltage scaling and frequency scaling are employed simultaneously, such as in cell phones, when they are in stand-by mode.

5.3.4 Clock gating

In the previous sections we have referred to (sub)systems that do not always operate at their maximum performance. It is also possible that parts of a system are idle for a period of time: then no useful computational work is performed. Still, there is power consumed. A subsystem being idle does not necessarily mean that the subsystem is not performing any computations. It only means

that the results are not being utilized. This is possible when the subsystem is still fed with data, but the result is discarded, because it is not needed at that moment. But even when the subsystem is not performing any computational work, there is still the static power dissipation (leakage power) of the subsystem. What is more, flip-flops dissipate some dynamic power every single clock cycle, even when the in- and outputs remain the same. If there are large registers present in these subsystems, this power dissipation can become quite significant. And, finally, there is the power dissipation of the clock network in the subsystem. Clock networks are very expensive in terms of power. A major portion of the total power consumption of the system is dissipated in the clock network (mainly in the clock buffers/drivers) [24]. Considering the above, there is a lot of power that can be saved when a subsystem is idle. One way to achieve this goal is to apply clock gating. This essentially means that the clock signal of the subsystem is cut off. This will save the power dissipated in flip-flops and the clock network. If the combinational logic in the subsystem is fed by registers at the inputs, the logic will stop switching [11]. It will, however, not save the leakage power.

5.3.5 Power Gating

Power gating provides basically a solution to the same problem mentioned in clock gating section. Power gating has however an important advantage over clock gating: it is capable to save static power of idle blocks as well, since it cuts of the power supply instead of the clock signal. In order to do that, blocks need to be placed onto separate power domains, which can be powered on and off. When a block is asleep it costs some time to wake it up again, the same as it costs some time to put a block to sleep. This introduces additional delays. Also during wake-up and going to sleep, still some leakage power is dissipated which makes power gating not perfect. There is always a cost associated with the state transition. The essential criteria for implementing power gating is the total leakage power component and how many and how often blocks are idle. The leakage power highly depends on the technology being utilized and the impact of the leakage power highly

depends on the system frequency being utilized. If the leakage component is significant and many blocks are idle for longer periods of time, power gating may be efficient. One should however be aware of the fact that power gating is much more difficult to implement than clock gating and leads to significantly higher costs (mainly because of all the switches that are required). It is also important to realize that power gating is much more invasive than clock gating. While clock gating does not affect the functionality of the system, power gating does. It affects inter-block communication and, as mentioned before, adds time delays to safely enter and exit power gated modes [24].

5.4 Design flow

The design flow of any system consist of design at various levels of abstraction. Abstraction hides complexity to enable more structured design and implementation of complex systems. The aspect of the system that needs to be optimized helps in designing the various levels of abstraction. When a system is designed with the emphasis on power optimization, then the design must embody optimization at all levels of the design flow. In general, the support for energy management needs to be incorporated at the system level and the architectural level. An important aspect of the design flow is the relation and feedback between the levels. Havinga et al has come up with three layers of abstraction: system, architecture, and technology [23, 31, 22]. Table 5.1 describes the abstraction levels and related examples for energy reduction. ‘

At the highest level, the design decisions have the most influence. Therefore, the most effective design decisions derive from choosing and optimizing architectures and algorithms at the highest levels. It has been demonstrated by several researchers [32] that system- and architecture-level design decisions can have a great impact on power consumption. However, when designing a system it is a problem to predict the consequences and effectiveness of high-level design decisions because implementation details can only be accurately modeled or estimated at the technological level and

Table 5.1: Abstraction level and related example for energy reduction.

Abstraction level	Examples
System	processing method energy manager scheduling light weight filesystem system partitioning
Architecture	parallel hardware Direct Memory Access FFT hardware
Technological	clock gating clock frequency control voltage control

not at the higher levels of abstraction. Furthermore, the specific energy reduction techniques that are offered by the lower layers can be most effective only when the higher levels are aware of these techniques, know how to use them, and apply them.

The above-mentioned layers can be considered as vertical layers of any digital system. The BlueBox system could be broken down into three main subsystem as shown in Figure 5.2 based on its application semantics: sensor, processor, and storage subsystem. These sub-systems work together and are interdependent on each other for their functionalities. Each of these subsystems contains all three abstraction layers mentioned above. Our low-power design flow targets each of these three horizontal layers of the subsystems individually and the interdependent vertical layers within them. It is interesting that effect on a single subsystem creates space for further power savings for its dependent subsystem. For example an optimization in storage subsystem provides opportunity to keep the DSP idle for longer time.

Our approach with this multi-dimensional design space offers us a large range of possible trade-offs. Also, our modeling of the system with horizontal and vertical layers made it fairly easy to think and work on power management. The following sections discuss our approach to minimizing power consumption in sensor, processor, and storage subsystems.

Table 5.2: Sub-Systems of BlueBox

Sub-System	*Sub-Systemcomponents*
Sensors	Accelerometer Environment Temperature sensor Audio Codec Ecg Analog Front end Body temperature sensor
Processor	DSP core GPIO Peripherals DMA Memory
Storage	SD card EMIF

5.5 Power Management of Sensor sub-system

A sensor is a device that converts real-world physical parameters into electrical signals that a computer can understand. They are the most important active part of the BlueBox system. As shown in Table 5.2, BlueBox employs five different sensors, most of which are actively measuring the physical signal in real-time, as opposed to a triggered measurement.

They also do have a processing capabilities embodied with them. They contribute to the significant amount of power consumed by the system. The first, and perhaps most direct, approach of our low power consumption design targets the sensor itself. BlueBox employs power mode management, smart sensing, and computational offloading to reduce the power consumption of the DSP. The following subsections describe the three simple techniques that enabled low power sensing.

5.5.1 Sensor power mode management

Most of the sensors designed for low power consumption have one or more lower-power states such as a shutdown mode or an extremely low-power operating mode [3]. In many instances, this

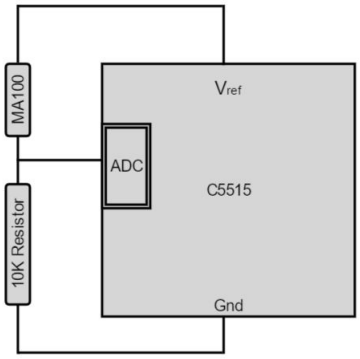


Figure 5.1: Simplified version Body Temperature measurement circuitry

type of sensor operation is directly controlled by the firmware designer in the end application. It is also important to note that there is always a cost associated with the low-power modes. The cost could be reduced performance, increased latency, delayed boot-up, etc. Also, transition of modes consumes power too. So, it is the application semantics that decides the best modes of operation and how often transitions need to be triggered.

Considering the body temperature sensor (MA100) from BlueBox system. It is a simple thermistor. The voltage drop across this thermistor is a value proportional to the body temperature. The simplified circuitry of MA100 can be seen in Figure 5.1.

The C5515 DSP supplies the reference voltage to the thermistor MA100. As body temperature is measured only twice in a second, the reference voltage could be supplied only during the measurement and can be turned off the rest of the time to save the power dissipated by the measurement circuit during the unused time.

From Section 3.5, it can be noticed that the accelerometer sensor (MPU9250) takes 30 ms to wake up from sleep and generate a valid accelerometer value. BlueBox cannot afford to spend this time inside the ISR. So, it is a good idea to keep the MPU9250 under low-power accelerometer mode, which consumes 19.8 μA providing the required response time.

5.5.2 Smart Sensing

Smart sensing is a method for achieving low power that uses integrated 'smart' digital logic in the sensor. "Smart Sensing" is the added digital capability that can be used to enable the sensor to perform its own internal power management or help reduce the power consumption of the system. This approach has been used in BlueBox through the smart capabilities of accelerometer.

The sensor's embedded logic detects events and notifies the DSP over interrupt. Unlike the previous low-power method that has no power consumption at its lowest level, this one has a baseline power consumption, so it can automatically power itself on. With a higher frequency of use, a device with more internal intelligence eventually becomes more efficient.

An example of such technique in the BlueBox, for reducing power consumption is the use of accelerometers smart first-in, first-out (FIFO) buffer. An 8-bit or 12-bit configurable 32-sample FIFO allows buffering of the data, so a host system can power on the sensor and read the data at a slower rate. As a result, the DSP can be kept in a lower-power state more of the time, achieving lower power consumption for itself and lower duty-cycle operation for the DSP whose power consumption is approximately ten times higher than that of the accelerometer. By reducing the DSPs power-on time by a very small amount, the entire system might still consume less power even though the accelerometer consumes more power. The system-level advantages more than compensate for the slightly higher power consumption in the sensor.

5.5.3 Computational offloading

In this third approach for low power sensing, the DSP utilizes the local computing capability of the sensor. Termed "local compute capability," this system-level methodology can offload the computation from system application DSP. For example, a DSP that uses 100 to 1000 times as much power, a local sensor with the compute capability can perform the required computations

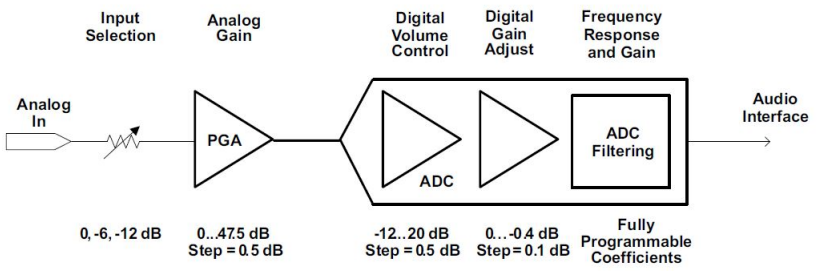


Figure 5.2: Signal Processing blocks of Audio Codec. [9]

and allow the DSP to go into lower-power modes delivers a significant power savings at the system level.

This design methodology is used in BlueBox to reduce the power consumption of audio data acquisition through the use of local compute capability of the Audio Codec. Audio signal captured through the electret microphones contains some noise with it and is required to filter the signal to improve the audio quality and attenuate high-frequency noises. The process of digital filtering at the DSP could be a good solution as the filter can be designed specific for the application and the signal of interest. However, it consumes more computational bandwidth and power, which is not desirable for the BlueBox application. So, BlueBox uses the signal processing support extended by the audio codec AIC3204 as shown in Figure 5.2. The codec has a fixed number of filters with configurable coefficients. So, this application-specific filtering circuit in the codec would be more efficient, perform faster, and consume significantly less energy than doing the same filtering process in the general-purpose processor. Offloading the digital-filtering computation to the codec saves a significant amount of computational bandwidth and power.

5.6 Power Management of DSP subsystem

Selecting a processor is one of the most critical decisions when we are designing an embedded system. This selection is based on the features required to satisfy the control functionality of the

final product and the raw computing power needed to fulfill those system requirements. There are several points to be aware of when selecting a processor. The overall power consumption of an MCU is defined by its power consumption in different modes, typically active and standby (including sleep, idle, standby, etc.), and taking into account the power consumed to transition from one mode to another.

Active power consumption by a processor is the power consumed when the processor is running. As almost all controllers are based upon CMOS logic, power is consumed primarily during the switching of transistors. The other major factor that determines the run time of the system is the standby power consumption. Most applications can spend a significant amount of time in standby mode. Standby current is the sum of leakage current, current consumed by power management circuits, clocking systems, power regulators, RTC, I/Os, interrupt controllers, and so on. It varies from controller to controller, based upon the particular features and peripherals supported in standby mode. The power consumed in standby mode is significantly less than in active mode. So, the basic idea is to put the processor into standby mode whenever possible and try to keep the processor idle as long as possible to reduce the power consumed by the processor. The following section discusses the techniques used in the BlueBox system to reduce the processor power consumption.

Timing analysis is the first step towards the understanding of the processor functionality in terms of processor utilization [25]. The processor utilization U is defined as the total amount of time spent idle by the processor.

$$U = 100\% - (\%T_{\text{idle}}) \tag{5.1}$$

where the idle time T_{idle} is the time during which the processor does not do any useful work.

Normally, this is a loop that spins the processor busy-waiting for an event. The Code Composer Studio IDE, which is used for firmware development, provides the timing analysis of the code execution in DSP. Once the average idle time is known, we can measure the active utilization when the system is under various states of loads using Equation (5.1). BlueBox employs *intelligent waiting*, *event reduction*, and *intelligent peripheral shutdown* techniques to reduce the power consumption of the DSP. The following sections explains each of the technique in detail.

5.6.1 Intelligent Waiting

The TMS320C5515 DSP includes various run-time power modes that can be used to scale down power consumption. The most common is idle mode of operation, in which the instruction-executing portion of the processor core shuts down while all peripherals and interrupts remain powered-on and active. Idle mode consumes substantially less power than when the processor is actively executing instructions. A key aspect of idle mode is that it requires little overhead to enter and exit, usually allowing it to be applied many times every millisecond. Anytime the DSP is spinning in a loop or blocked waiting for an event, it should be placed into idle mode to conserve power. Since any interrupt can wake the DSP from idle mode, use of this mode enables the DSP to intelligently wait for events in the system. The BlueBox system is dependent on external interrupts generated by the ECG analog frontend and the audio codec through the I2S bus. From Figure 4.1, it can be seen that the main thread of the BlueBox goes to idle mode after writing the pages to SD card, if they are available. When an interrupt occurs, the DSP wakes up and invokes the interrupt service routine (ISR), which checks if any page is available to write and then goes back into idle mode to save power.

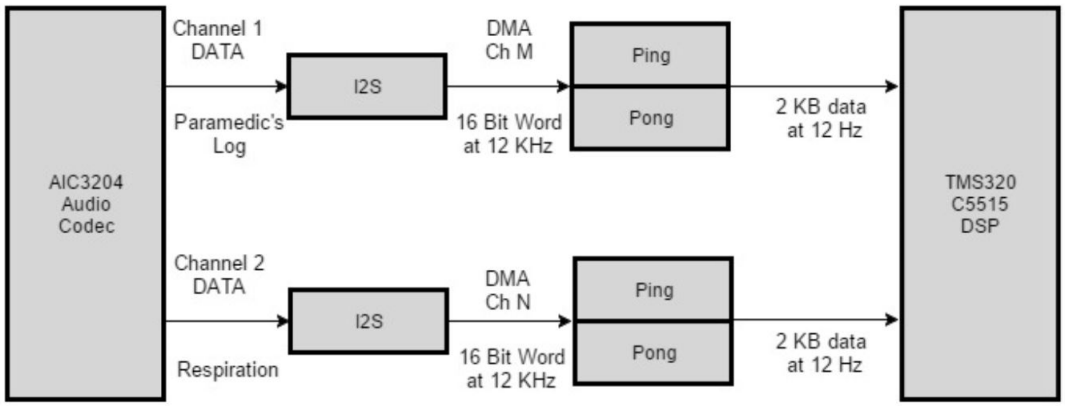


Figure 5.3: Event Reduction through DMA for Audio acquisition

5.6.2 Event Reduction

Another important technique employed is event reduction. Whereas intelligent waiting enables the processor to enter its idle mode as often as possible, event reduction attempts to keep the processor in idle mode for as long as possible. It is implemented by analyzing the application firmware code and system requirements. It is also required to make sure that the effect of optimization of processing the interrupt/event does not affect other functionalities. The interrupts associated with the BlueBox are periodic external interrupts to trigger the processor to read and collect the data. There are totally 24000 audio interrupts generated every second by the audio codec. If the processor handles all the interrupts, it will not have suffient time to go to idle mode and have to be active all the time. This is where Direct Memory Access (DMA) comes into play to help the processor. DMA allows the processor to remain in idle mode for significant periods even while data is being sent to or received from peripherals. DMA should be used in peripheral drivers whenever possible. The savings are quite impressive. The DMA employed with the audio codec reduces 24000 interrupts per second to 24 interrupt per second. Figure 5.3 explains how the reduction is achieved.

The receiver DMA ping-pong buffer is of 2 KB size, and it takes 1000 interrupts to move the data into the buffer. When the buffer is full, the DMA interrupts the processor to take further actions. In the mean time, the DMA starts servicing the interrupts filling up the pong buffer. Therefore, during the situations where the processor is more prone to face the external interrupts, the event reduction approach using the DMA will be the solution.

The intelligent waiting and the event reduction techniques together have brought down the DSP active time, utilization, to roughly 20.92%. The details of the calculation are presented in Section 6.2.

5.6.3 Intelligent Peripheral Shutdown

All the above methods are for the power consumption when the device is running. This method of intelligent shutdown is useful when the peripheral is clock or power gated when it is not in use. The C5515 processor continues to power the general purpose input output pins and peripherals during idle mode. As inputs, these I/O pins can be used as interrupts to wake up the device; as outputs, they can be used to configure to external peripherals. Careful configuration of how these pins are configured can have large effect on sleep mode power consumption. The DSP has separate power domains which provide power to different portions of the device. The separate power domains allows the firmware to select the optimal voltage to achieve the lowest power consumption at the best possible performance. There are 13 different power domains each of which powers a specific domain. Figure A.1 gives the brief gist of the power domains. Body temperature is measured twice in a second. Figure 5.1 shows the simplified body temperature measurement circuitry. It can be understood that the ADC peripheral of the DSP will be used only twice every second. So BlueBox firmware enables the analog power domain that powers the ADC only during the measurement. The analog power domain and also the ADC peripheral are shut down during the rest of the time.

5.7 Power Management of Storage Subsystem

Storage contributes to a significant fraction of power consumption of many embedded systems. With respect to data acquisition and logging system, the fraction is somewhere between 30 to 50%. From Table 2.6, we can notice that power consumption of the SDHC card is about half of that of the entire system when a read or write action is performed. It is thus required to manage the storage sub-sytem with utmost care. So, the basic strategy is to write or read only when it is required and to keep the SDHC card utilization as low as possible. Storage subsystem involves the SDHC card and MMC/SD interface from the DSP. As explained in Section 3.7, the SDHC card is block addressable. Blocks of size 512 bytes can be written or read from the SD card. So, the processor has to buffer or cache blocks of data and write them to the SD card.

When recording five different signal streams onto the SD card, BlueBox system organizes them into two categories: the audio signal is in its own category, while ECG, accelerometer, and temperature data are in the other. (2/21/2017) so that is four signal streams, not five? what about time stamp? Separate buffers are associated with each of these categories, and the respective data is written into them and eventually flushed to the SD card. The reason for separating out the audio into its own category is its data rate of 48 KB/s, which is higher than the other signals combined. So, it makes sense to have dedicated buffer for audio while having the other signals share a buffer.

The ECG analog front end interrupts the DSP at a frequency of 500 Hz. Each sample collected during the interrupt is appended to the buffer. Because the accelerometer only needs to be sampled at 12 Hz, a counter within the interrupt ensures that the accelerometer is sampled only once every 40 interrupts. Similarly, since the temperature is to be sampled at 2 Hz, a counter within the interrupt service routine is used to take a sample every 250 interrupts. Each time the signals are sampled, values from each of the sensors are stored in the respective position in the buffer.

5.7.1 Saving Data to MicroSD Card

Because the BlueBox must be able to function continuously, the interrupt services must continue to operate even when data is being written to SD card. So, DMAs are used to write the data to the SD card as it can do the task without the interference of the DSP. When possible, sampled data is saved directly to a primary data buffer. When the primary data buffer is full DMA is instructed to initiate the transfer of the data from primary buffer into the SD card persistent storage. In the meantime data collected is stored in the secondary buffer and the cycle goes on.

There are lot of different ways of writing and organizing the data in the SD card. One way is to use existing filesystem that is recognizable by the general operating system, other way is to write the raw data and to design a n application specific custom filesystem. There are pros and cons associated with designing a new filesystem. Using a existing standard filesystem like FAT, NTFS etc., are really simple to implement as standard libraries are present but they were all designed for user comfortability and are designed for higher end computational devices. There is a huge overhead associated with storing and reading of data. For example, in order to write a block of data into a file, first it takes multiple SD card reads to figure out to which block address the data should be written then the actual processes of writing happens. This process keeps the SD card busy for longer duration, which is very undesirable for our application. It is required to come up with a simple custom filesystem that could suffice the necessity of application. The problems of custom filesystem is incompatibility with standard operating systems. To read data from the custom formatted SD card, it is required to develop a filesystem driver which is capable of reading this SD card. It requires some time to implement and test the custom filesystem. The important thing here to notice is that the custom filesystem could potentially save time and power it consumes to write data to the SD card.

5.7.2 File System

A Secure Digital High Capacity(SD) card basically has a SD controller module and a memory core. Memory core is where the data is stored in. The memory core of a SDHC card is divided into blocks of 512 bytes and they are block addressable. BlueBox caches the data in a local buffer, which are, in general multiples of block size of the SD card. For convenience of understanding, the buffer that caches the data is referred as pages. The custom filesystem developed for BlueBox has three categories of pages. They are audio page(same page layout used for both the channels of audio), vitals page(includes ECG, temperature and accelerometer data) and index page. The page size used for audio is 2Kb and for ECG its 512 bytes. Figure 5.4 shows the layout of audio page. It just contains the actual audio data and does not encode any meta-data with it. The pointers to the audio pages are present in the Index page. Each audio page accounts for four blocks, so when ever an audio page is written to the SD card, the corresponding four block address are stored in the index page.

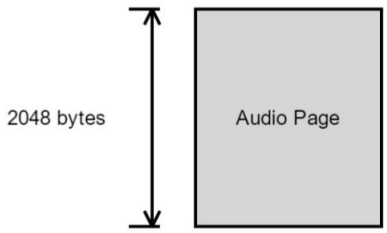


Figure 5.4: Audio Page

Figure 5.5 depicts the Vitals page. Its 512 bytes in size and contains actual data along with meta data. The first two bytes of the vitals page is an identifier, which uniquely identifies it. It includes ECG, accelerometer, temperature and time data.

The index page comprises of a 2 byte page identifier, 2 byte Audio channel identifier and 127 32bit block address as shown in figure 5.6. There is a separate index pages for left and right channel audio. The block addresses in the index pages are the pointers to next pieces of the audio data. The

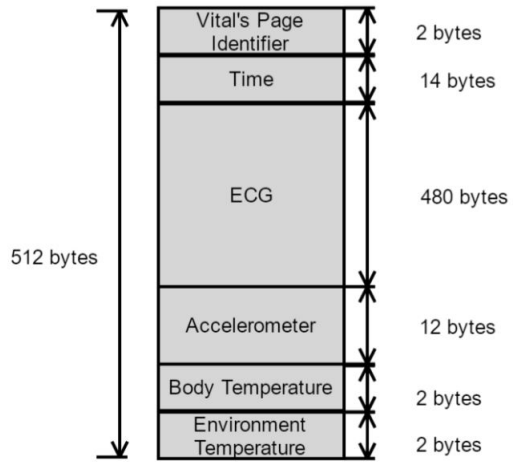


Figure 5.5: Vitals Page

audio data is reconstructed into a file using these index pages.

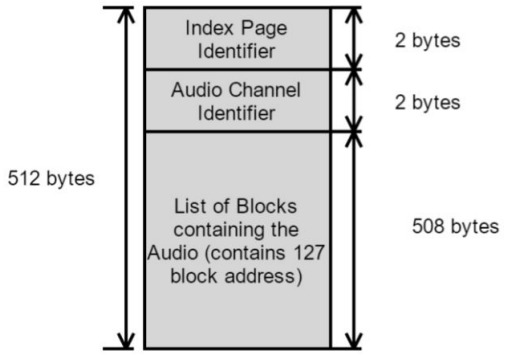
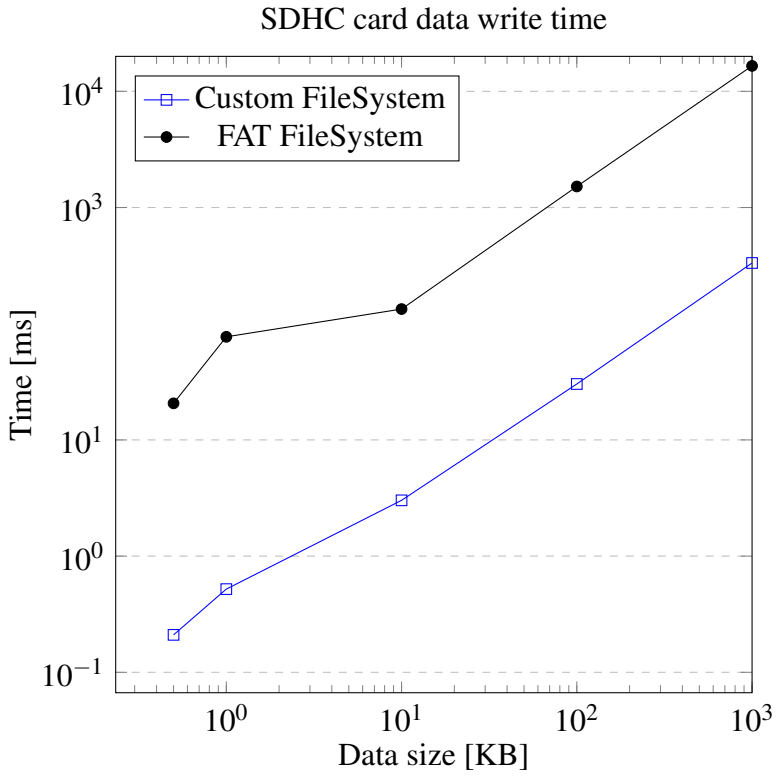


Figure 5.6: Index Page

The firmware filesystem driver keeps track of the next available free block and decides to which block does the data in the page go into. All the pages are written to the SD card sequentially. After the page has been successfully written by the DMA, the block address pointer is updated to point to the next free block. 24 pages of audio sample is written per second and 4 pages of ECG and other aggregated data is written per second. The filesystem inherently keeps track of content of each block through page identifiers and the pointers in the index page. The index page is also cached in a buffer and when it fills up it is written into the SD card. So that while reading back the data from the SD card we could organize the blocks and reconstruct the data stream.

This simple custom filesystem helped in reducing the overhead of writing data to SD card, which in general is very high with a standard filesystem that adds up bunch of meta data. The following graph 5.7.2 shows the time taken to write the data to sd card and compares the two filesystem. The x axis marks the amount of data(includes only the meaningful sensor data) and the y axis marks time in millisecond.



From the figure 5.7.2 it is evident that for the same amount of actual data the custom filesystem takes, on an average, 50x less time to write than the FAT filesystem. The custom filesystem essentially spends less energy to save the data by reducing the active utilization time by 50x.

From the experiment with SDHC card, running at 20MHz clock at 3.3 V, the write/read operation consumes 48 mA (considering only DSP and SD card active) and in standby it consumes 22 mA. BlueBox system requires to write 50 KB of data every second as mentioned in section 2.3. The utilization of storage($U_{storage}$) using the custom filesystem can be computed as follows:

$$U_{storage} = \frac{Timetakentostore50KB}{F_{DSP}} * 100\%$$

$$U_{storage} = \frac{1,514,088}{100,000,000} * 100\% = 1.5\%$$

With FAT filesystem, the storage utilization is closer to 70% whereas it is just 1.5% with the custom file system. The system draws an average current of 40.2 mA when writing 50KB of data in one second on FAT filesystem whereas the custom filesystem's average current consumption is 22.27 mA. Thus the designed filesystem reduces the system's average power consumption by approximately 44.6% within the application semantics of BlueBox.

Chapter 6

Experiments and Clinical Test

This Chapter explains the setup for testing the basic device functionalities and its performance. The Chapter also discusses the clinical test conducted at LA BioMed at Torrance using Mannequin-based Patient Simulation. The study protocol, analysis and preliminary results will be discussed.

6.1 Motivation

The motivation for the study can be broken down into the following:

- To perform preliminary testing of device functionalities and its runtime
- To perform a clinical test to evaluate the signal quality and to test the usability
- To obtain suite of signals in a data set for future algorithm optimization

6.2 Preliminary Experiments and Energy Characterization

The basic device functionality of the device is tested by simulating the required signals locally in our lab 6.1. The ECG signal is simulated using Precise Life SKX 2000 ECG simulator. The ECG is interfaced in 4 lead configuration using right arm(RA), left arm(LA), left leg(LL) and right leg(RL) drive leads. The body temperature is simulated using sodium acetate based reusable head pods. The acceleration of chest, paramedic log and breathing sound are manually simulated.

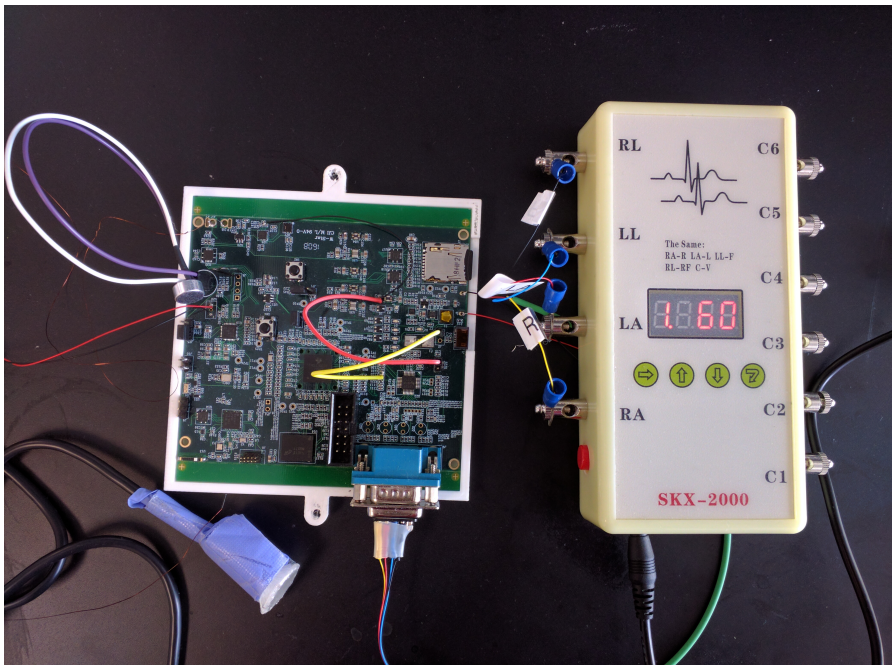


Figure 6.1: Prototype Test Environment

The system is tested with the firmware that incorporated the techniques mentioned in the previous chapter. In order to determine the power dissipation, the relative contribution of the different parts of the system needs to be considered. First, the idle mode current of the system was measured to be 22 mA. The environment temperature sensor, ECG front end chip and the accelerometer are powered directly from the battery. They are not controlled by the DSP, so the state of the DSP does not affect their power consumption. These components always contribute towards the current drawn by the system irrespective of the state of DSP. The average current drawn by the system un-

der different modes of operation are shown in table6.1. The average current drawn by the system with all the sensors and DSP (running at 100MHz) being active was measured to be 38.64 mA. The average current drawn when writing to the SD card while all the components of the system are active was measured to be 48 mA. The current drawn by the system was measured using INA231 evaluation module [6], which samples the current drawn by the system every 140 μS and logs the data.

Operation	Current
Idle	22 mA
Data Acquisition	38.64 mA
Micro SD write	48 mA

Table 6.1: Current drawn for different operation mode.

We also measured the DSP utilization to have more insight into how long the system is in different operation mode mentioned above. We used the Code Composer studio's sophisticated profile clock to count the utilization of DSP [2]. As mentioned in section 4 there are three main routines that run: main, DMA ISR and ECG ISR. The worst case execution time of DMA ISR ($WCET_{DMA}$) routine is 3000 cycles to service the interrupt and it is executed for 24 times in a second. The DMA ISR utilization (U_{DMA}) can be calculated as follows:

$$U_{DMA} = \frac{WCET_{DMA} * Numberof Interrupts/Second}{F_{DSP}} * 100\%$$

$$U_{DMA} = \frac{3000 * 24}{100,000,000} * 100\% = 0.72\%$$

Where F_{DSP} is the frequency at which the DSP is running, which is 100 MHz in our case. Similarly the main routine and the ECG ISR is profiled. The summary of the worst case utilization

of these routines are presented in table 6.2. The total utilization of the DSP in active mode is roughly 20.92%.

Routinue	Utilization
Main	18%
DMA ISR	0.72%
ECG ISR	2.204%

Table 6.2: Current drawn for different operation mode.

The Storage utilization with the custom filesystem is computed to be 1.5%. The worst case contribution of storage towards the average current drawn by the system is

$$48mA * \frac{1.5}{100} = 0.72mA$$

The average current drawn by the system considering during the idle and active mode:

$$38.64mA * \frac{20.924}{100} + 22mA * \frac{79.076}{100} = 25.481mA$$

The average current drawn by the system sums up to approximately 26.201 mA. With this rate the system should be able to run for 9 - 10 hours before recharging the battery. The acquired data was verified by primarily reconstructing the signal and confirm if the amount of samples of each signal is equivalent to the test duration. This ensures the correctness of the process of data recording. The validity of the data was verified manually from playing back the signals and validating if the reflected the conditions during the test.

6.3 Clinical Test, Setup and Protocol

6.3.1 Mannequin-based Patient Simulation

BlueBox initial clinical test is done with a Mannequin-based patient simulation environment. Taking the underlying concepts described for the desktop simulation one step further is the recreation of the real physical patient in a realistic physical clinical environment [1]. Thus, computerized mannequin stands in for the patient, and a variety of equipment can be used (either real clinical equipment or computer-driven replicas) to monitor and treat the patient. Among the functions that these mannequin-based simulators can replicate are:

- Spontaneous breathing (and the ability to breathe for the patient with a bag or ventilator)
- Pulses, heart sounds, breath sounds, pupil size, pupil response to light
- Real-time display of electronically monitored information (e.g. ECG, oxygen saturation, etc.)

Not only are these created in their normal manifestation, but all the elements of a large variety of abnormal conditions can be created such as (the list is almost endless) heart attack, severe allergic reaction, breathing difficulties, sepsis (blood poisoning), severe abnormalities of sugar metabolism etc.

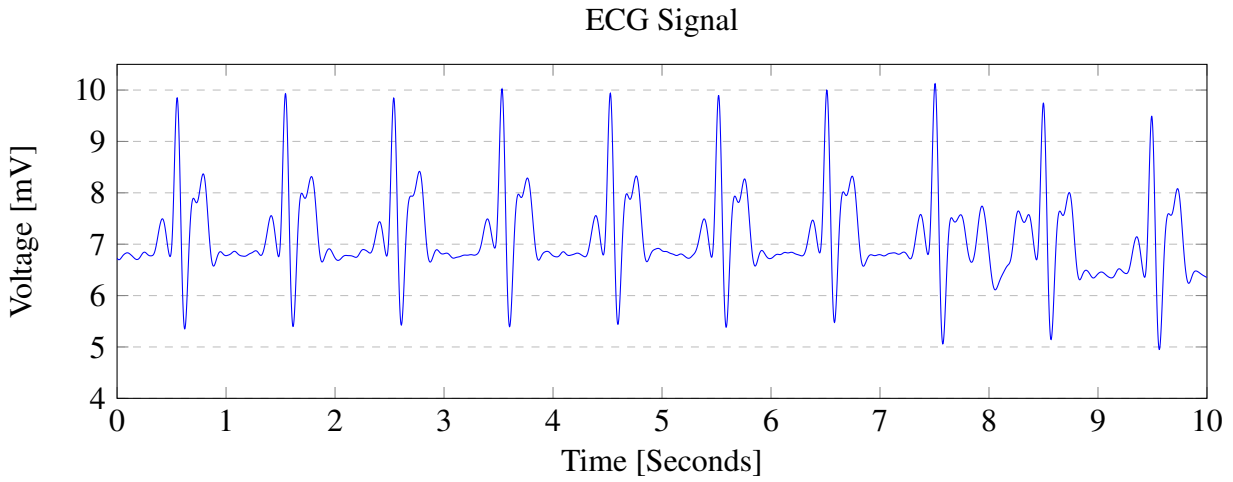
6.3.2 Test Protocol

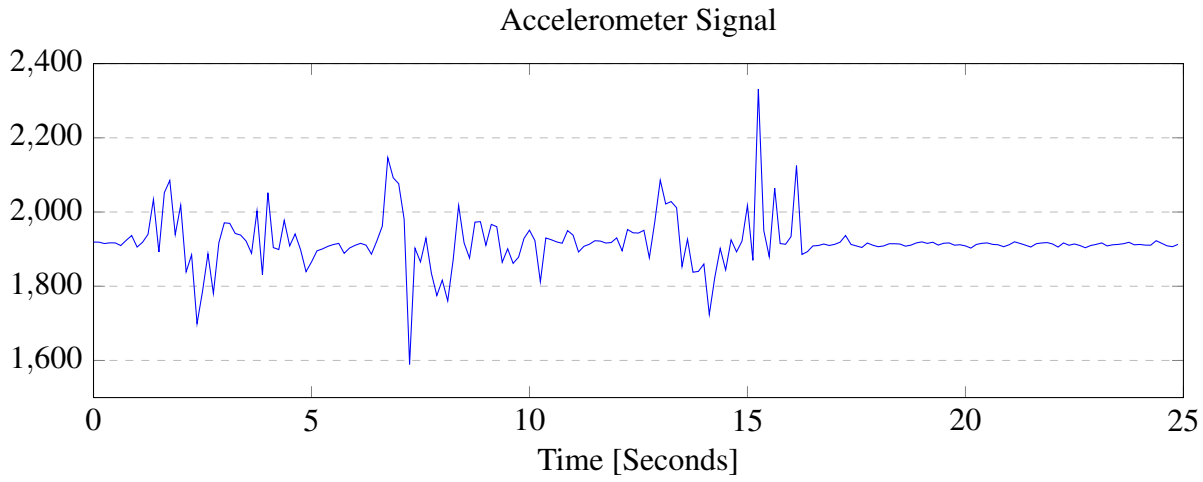
The test was done with the BlueBox device placed on the Mannequin's chest. The Mannequin wore the four electrode model in a lead II configuration. The body temperature sensor is worn near the mannequin's neck. The contact microphone to record the respiration sound is placed on the

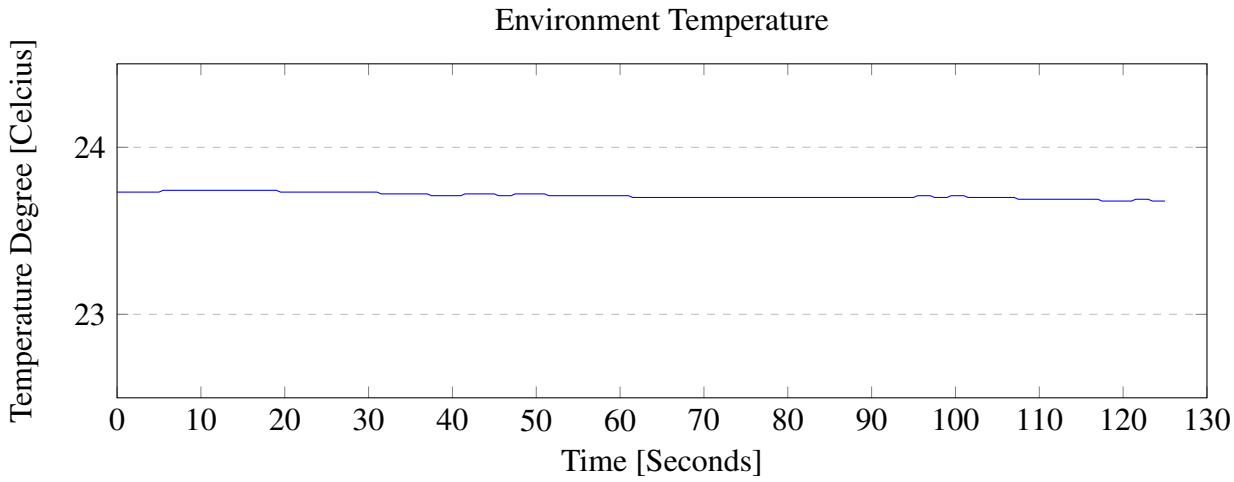
mannequin's abdomen. The environment microphone to record paramedic's comments is on the top of the device. As the accelerometer is on board, it can measure the chest rise of the mannequin. The on-board temperature sensor can measure environment temperature. The preliminary tests were conducted for different time duration ranging from 15 minutes to 2 hours. The tests were conducted under normal conditions and variety of abnormal conditions like heart attack, severe allergic reaction, breathing difficulties etc. The device is switched on using a button during the beginning of th experiment and after the test duration is completed, the data is imported into the computer to verify if the signal reflected the test conditions. The lab researchers and doctors on the IRB are required to be present for the duration of the test.

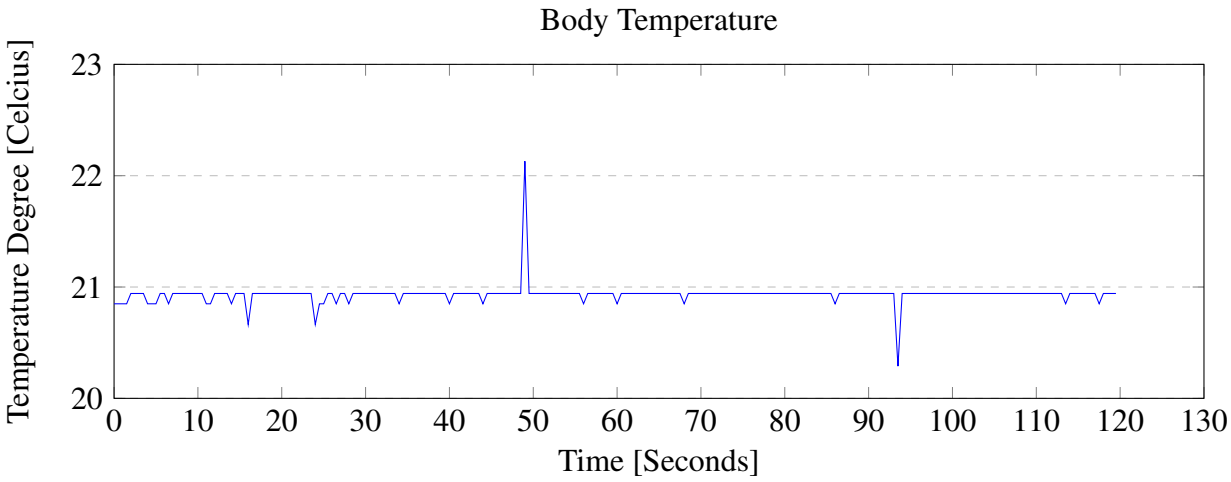
6.4 Analysis and Result

The important objective of the test is to evaluate the signal quality and performance of the BlueBox device. This is achieved by comparing the acquired signal with the originally simulated signals. The computerized mannequin has a record of all physically simulated signals (including ECG, breathing pattern and body temperature). All the vitals are compared with the original signal for evaluation. The two ECG signals are aligned in time to make a simple analysis. By analyzing these two signals physicians concluded that the acquired signal is reliable. Body temperature and chest movements were also analyzed with the same manner and are reliable. The following figures shows the samples of results of signals measured during the Clinical tests.


ECG Signal
Voltage [mV]
10
9
8
7
6
5
4
0
1
2
3
4
5
6
7
8
9
10
Time [Seconds]


Accelerometer Signal
2,400
2,200
2,000
1,800
1,600
0
5
10
15
20
25
Time [Seconds]


Environment Temperature
Temperature Degree [Celcius]
24
23
0
10
20
30
40
50
60
70
80
90
100
110
120
130
Time [Seconds]



The environment temperature is evaluated by simultaneously measuring the environment temperature using a standard device and comparing their values. This comparison confirmed the reliability of it. The two channels of audio acquired are analyzed by playing it back and manually verifying the quality. The analysis concluded that paramedic's log audio signal carries the required information along with some noise. It requires a proper post processing in order to remove noise. The contact microphone was sensitive to the paramedic's logs and it kind of made the breathing sound feeble. A microphone or a sensor that could be sensitive to the respiration and not sensitive to the atmospheric noise to be developed in order to get the required respiration information.

Chapter 7

Conclusion

Low-power firmware for wearable emergency health monitoring device has been designed and developed. The initial phase of the Clinical testing has been done using mannequin-based patient simulation environment. Further clinical testing of the device is being done currently. In this last Chapter, the ideas for the future work is presented.

7.1 Discussion

While the preliminary results suggests that the BlueBox device could be a good viable option for usage in ambulance for emergency monitoring, there are number of things that are learnt during the design process to improve the system for the future. First, the respiration audio acquired contained noise and other signals including the environment sound. The microphone used for sensing the respiration audio is sensitive to environment noise, including the paramedic's log. Attempt was made at post processing to remove the noise through complex filtering operations but still a clear respiration audio was not reconstructed. Removing the noise at the source seemed to be a good solution for the problem. Then the respiration microphone was placed inside a stethoscope chest

piece, that amplified the breathe sound and attenuated the environment noise. The revision of microphone solved the problem of breathe signal acquisition, but the chest piece of a stethoscope is heavy and is not really comfortable to wear. An alternate sensor that serves the exact purpose needs to designed in order make it comfortable to wear. The other problem was with the noise in the ECG signal that appeared due the harmonics from the source. A simple filter in the post processing filtered the noise and solved the issue.

There were two primary usability issue with the device. The first issue is that because the device does not transmit to a base station(tablet, mobile device), there is no live feedback about the signal quality and it is too hard to check if all the connections and placement of electrodes are proper. The board has to be placed on the body and run during the test period. Then the micro SD card from the board is removed and the data can be analyzed on the computer. A live feature with the radio transmission meant for debugging could help to confirm the signal quality. And the second is that because the central BlueBox board was made using a rigid PCB it makes the device sometimes uncomfortable to fit it on the patients. A flexible PCB could be a required future work to make the device take different shapes and become comfortable to wear.

7.2 Future Work

The basic proof of concept functionality of BlueBox has been shown with the mannequin-based patient simulation environment. Further clinical tests on real patients is necessary to validate the device. Further improvements of the device needs to be done to meet the medical grade requirements.

Future work should investigate on the design of proper sensor to sense the breathing pattern of the patient while making sure that the designed sensor setup is comfortable to wear. Also the future work should also research on real time transmission of data to a base station, while making

sure that battery life of the device is maintained as per the requirement. It is also required to evaluate the possibility of implementing the system on a flexible PCB within the constrained size and dimension. Additionally, a properly molded enclosure for the PCB could help improve the patient's acceptability of wearing the device.

7.3 Summary

Wearable emergency health monitoring on ambulance could give information about the condition of the patient from the time of emergency. It can provide the physicians with additional information such as the paramedic's logs on patient's response to emergency medications. In this thesis BlueBox, a wearable emergency health monitoring device, was discussed and the low power firmware for this device was developed. Previous works on such devices did not focus on adding and synchronizing the paramedic's logs unlike BlueBox. Preliminary clinical results suggest that it is a potential system to be used in ambulance for Emergency Health Monitoring.

Bibliography

[1] Mannequin-based Patient Simulation. http://cisl.stanford.edu/what_is/sim_modalities/mannequin_sim.html.

[2] Profile clock in CCS. http://processors.wiki.ti.com/index.php/Profile_clock_in_CCS.

[3] White Paper: Low-Power Sensing Energy-efficient power solutions. http://cache.freescale.com/files/sensors/doc/white_paper/LOWPOWERSENSWP.pdf?&Parent_nodeId=&Parent_pageType=.

[4] Low-Power, 2-Channel, 24-Bit Analog Front-End for Biopotential Measurements . http://www.ti.com/lit/ds/symlink/ads1292.pdf, 2012.

[5] TMS320C5515/14/05/04 DSP Direct Memory Access (DMA) Controller User's Guide . http://www.ti.com/lit/ug/spruft2a/spruft2a.pdf, 2012.

[6] Ina231evm evaluation board user's guide. http://www.ti.com/lit/ug/sbou128/sbou128.pdf, 2013.

[7] TMS320C5515 Fixed-Point Digital Signal Processor. http://www.ti.com/lit/ds/symlink/tms320c5515.pdf, 2013.

[8] MPU-9250 Product Specification Revision 1.0. https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdf, 2014.

[9] TLV320AIC3204 Ultra Low Power Stereo Audio Codec. http://www.ti.com/lit/ds/slos602c/slos602c.pdf, 2014.

[10] Tms320c5515 dsp system user's guide. http://www.ti.com/lit/ug/sprufx5e/sprufx5e.pdf, 2014.

[11] L. Benini, G. De Micheli, and E. Macii. Designing low-power circuits: practical recipes , 2001.

[12] L. Botwinick, M. Bisognano, and C. Haraden. Leadership Guide to Patient Safety. Cambridge: Institute for Healthcare Improvement. , 2006.

[13] D. R. Boyd. The conceptual development of EMS systems in the United States. *Emergency Medical Services*, 11(1):19–23, 1982.

[14] G. R. Braen. On turbulent times for emergency medicine. *Annals of Emergency Medicine*, 25(2):271, 1995.

[15] A. Chandrakasan, S. Sheng, and R. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 2002.

[16] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1994.

[17] M. M. K. Delano. A long term wearable electrocardiogram (ECG) measurement system, 2012.

[18] M. Ditzel, R. H. J. M. Otten, and W. A. Serdijn. Power-Aware Architecting: for data-dominated applications , 2007.

[19] T. Gao, D. Greenspan, and M. Welsh. Improving Patient Monitoring and Tracking in Emergency Response .

[20] T. Gao, D. Greenspan, and M. Welsh. Wireless Medical Sensor Networks in Emergency .

[21] S. Gupta and M. Kumar. Optimizing Low Power Embedded Designs , 2010.

[22] P. Havinga. Design techniques for energy efficient and low-power systems Chapter 2 .

[23] P. J. Havinga and G. J. Smit. Design techniques for low power systems .

[24] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi. Low Power Methodology Manual: For System-on-Chip Design , 2007.

[25] N. A. A. Khan and K. Ushadevi. Embedded software power optimization techniques in real time operating system, 2007.

[26] S. Labs. Balancing Performance and Power Efficiency in Embedded Systems , 2010.

[27] J. M. R. M. Pedram. Power Aware Design Methodologies , 2002.

[28] R. G. Mark. Clinical electrocardiography and arrhythmias, 2004.

[29] C. Page, M. Sbat, K. Vazquez, and Z. D. Yalcin. Analysis of Emergency Medical Systems Across the World , 2013.

[30] J. M. Rabaey and M. Pedram. Low Power Aware Design Methodologies , 1996.

[31] M. Salajegheh. Software Techniques to Reduce the Energy Consumption of Low-Power Devices at the Limits of Digital Abstractions , 2013.

[32] B. R. Sheng S., Chandrakasan A. A portable multimedia terminal. *IEEE Communications Magazine*, 30(12):64–75, Dec. 1992.

Appendix A

A.1 C5515 DSP power domains

Power Domains	Description
Real-Time Clock Power Domain (CV_{DDRTC})	This domain powers the real-time clock digital circuits and oscillator pins (RTC_XI, RTC_XO). Nominal supply voltage can be 1.05 V through 1.3 V. **Note:** CV_{DDRTC} must always be powered by an external power source. None of the on-chip LDOs can power CV_{DDRTC}. This domain cannot be regulated internally, external regulation must be provided.
Core Power Domain (CV_{DD})	This domain powers the digital circuits that include the C55x CPU, on-chip memory, and peripherals. Nominal supply voltage is either 1.05 V or 1.3 V. This domain can be powered from the on-chip DSP_LDO.
Digital I/O Power Domain 1 (DV_{DDEMIF})	This domain powers all EMIF I/O only. Nominal supply voltage can be 1.8, 2.5, 2.75, or 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
Digital I/O Power Domain 2 (DV_{DDIO})	This domain powers all I/Os, except the EMIF I/O, USB I/O, USB oscillator I/O, some of the analog related digital pins, and the real-time clock power domain I/O. Nominal supply voltage can be 1.8, 2.5, 2.75, or 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
RTC I/O Power Domain (DV_{DDRTC})	This domain powers the WAKEUP and RTC_CLKOUT pins. Nominal supply voltage can be 1.8, 2.5, 2.75, or 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
PLL Power Domain (V_{DDA_PLL})	This domain powers the system clock generator PLL. Nominal supply voltage is 1.3 V. This domain can be powered from the on-chip analog LDO output pin (ANA_LDOO).
Analog Power Domain (V_{DDA_ANA})	This domain powers the power management analog circuits and the 10-bit SAR. Nominal supply voltage is 1.3 V. This domain can be powered from the on-chip analog LDO output pin (ANA_LDOO). **Note:** When externally powered, this domain must be always powered for proper operation.
USB Analog Power Domain (USB_V_{DDA1P3})	This domain powers the USB analog PHY. Nominal supply voltage is 1.3 V. This domain can be powered from on-chip USB_LDO output pin (USB_LDOO).
USB Digital Power Domain (USB_V_{DD1P3})	This domain powers the USB digital module. Nominal supply voltage is 1.3 V. This domain can be powered from on-chip USB_LDO output pin (USB_LDOO).
USB Oscillator Power Domain (USB_V_{DDOSC})	This domain powers the USB oscillator. Nominal supply voltage is 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
USB Transceiver & Analog Power Domain (USB_V_{DDA3P3})	This domain powers the USB transceiver. Nominal supply voltage is 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
USB PLL Power Domain (USB_V_{DDPLL})	This domain powers the USB PLL. Nominal supply voltage is 3.3 V. This domain cannot be powered by internal LDOs, external regulation must be provided.
LDOI Power Domain (LDOI)	This domain powers LDOs, POR comparator, and I/O supply for some pins. Nominal supply voltage is 1.8 V through 3.6 V. **Note:** This domain must be always powered for proper operation.

Figure A.1: C5515 DSP power domains. [10]