Power Management of the LMP91000 Potentiostat Board

Joseph Mire
Electrical and Computer Engineering
University of Alabama
Tuscaloosa, United States
jmmire@crimson.ua.edu

Masudul Imtiaz
Electrical and Computer Engineering
Clarkson University
Potsdam, United States
mimtiaz@clarkson.edu

***Abstract*—Potentiostats are traditionally confined to laboratory environments due to their high cost and dependence on external computers and power sources. An open-source wireless electrochemical sensing system was developed to enable portable electrochemical sensing in the field. However, the design consumes significant power, reducing its battery life and therefore its applicability in the field. This work presents modified firmware for the electrochemical system to increase power efficiency and add helpful features for battery management and wearable applications. This firmware improves the time to use 150 mAh by approximately 13 times, from 16 hours to nearly 9 days. The developed firmware powers up the circuit board with a press of a button. After the boot-up sequence, it stays in a low-power mode for a set time period until it wakes up briefly and collects data. A Real-Time Clock (RTC) was implemented to log timestamp data before the device goes to sleep. Additionally, the battery management software monitors the battery voltage and fully shuts down the board when the battery voltage gets low. The open-source potentiostat board and firmware files are available through https://github.com/sigmondkukla/lmp91000/tree/master. The smartphone application source code can be found at https://github.com/sigmondkukla/lmp91000-app**

***Index Terms*—Potentiostat, Portable electronics, Power efficiency**

I. Introduction

Potentiostats can be used in a variety of environmental testing scenarios, including pesticide detection and heavy metal detection. These experiments are largely performed using expensive desktop potentiostats in controlled environments, yet there are limited options for portable field testing systems. Several systems exist to try to address this gap. In 2011, CheapStat was developed as a low-cost electrochemical system [1]. In 2020, KickStat was introduced as a more modern, smaller version of the potentiostat system [2]. With all of these developments, few systems fully integrate a small, low-cost, wireless, and long-lasting system. The electrochemical sensing system used in this research is a continuation of the open-source lmp91000 project on GitHub by Sigmond Kukla [3]. It utilizes an EFR32 microcontroller paired with an IMU and an LMP91000 potentiostat integrated circuit. The EFR32 uses the low-powered Bluetooth LE communication standard to transmit data and receive commands. A smartphone application has been developed for ease of use in the field without a dedicated laptop or desktop computer. The application allows the user to switch between Cyclic Voltammetry (CV), Square Wave Voltammetry (SWV), Differential Pulse Voltammetry (DPV), and Chronoamperometry (CA) experiments. Each of these experiments can have its parameters modified in the smartphone application. As shown in Figure 1, this board is relatively small and is powered by a small lithium battery. This potentiostat board costs around $25 in physical materials, compared to laboratory potentiostats, which often start at $200. It operates on a small 150 mAh battery, which limits the device's operating time.

Work performed while participating in a Research Experience for Undergraduates (REU) at Clarkson University.

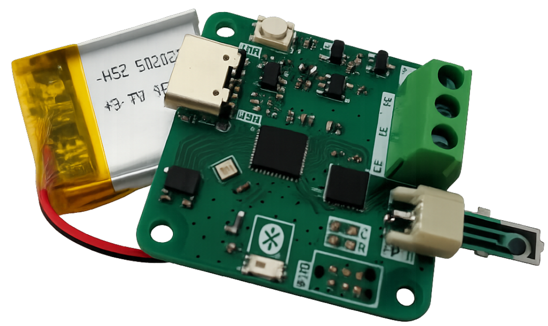

Fig. 1. Open-Source Electrochemistry Board

In this paper, a new power-optimized firmware for the LMP91000 potentiostat board is introduced. The EFR32 microcontroller has various power modes that disable or limit certain functionality. These power modes can be utilized to conserve power while not actively conducting experiments. This firmware puts the EFR32 microcontroller into an ultra-low-power mode when not in operation and wakes it up when the onboard IMU detects motion or a timer expires. This allows the device to operate significantly longer than running in full-power active mode the entire time of use. A software-tied latch to completely detach power from the board's power rails was also programmed to disable external battery power drain.

II. Experimental Setup

This device was programmed using Simplicity Studio 6 connected to Visual Studio Code as the integrated development environment. The custom potentiostat board was connected to the BRD4002A Wireless Pro Kit. Code was written, compiled, and flashed with Visual Studio Code. The positive battery pin of the electrochemistry board was connected to the VMCU pin of the BRD4002A Wireless Pro Kit to measure current. Simplicity Studio's Energy Profiler tool was used to graph the current versus time. As seen in Figure 2, the electrochemistry board was connected to the debug board with a Tag-Connect Debug Header. KiCad was used for schematic references.

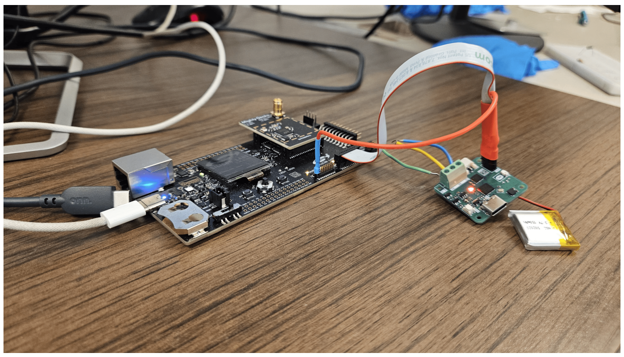

Fig. 2. Programming and Debugging Setup

III. Power Modes

The EFR32BG24 from Silicon Labs is the main controlling microcontroller used in this project. This microcontroller has five different power modes, each using different amounts of power and activating different parts of the microcontroller. The power modes from most active to least active are the following: EM0, EM1, EM2, EM3, and EM4 [4]. EM0 will be used interchangeably with active mode. This runs the microprocessor at full power. EM0 mode is used when the board is collecting data. EM1, EM2, and EM3 modes are not focused on in this project because EM0 and EM4 modes satisfy all requirements for this application. EM4 mode is the closest the microcontroller can be to a completely off state and uses the least amount of power. In this mode, most of the segments of the EFR32BG24 are deactivated, dropping the current of the MCU to less than a microampere. There are two defines for the sleep/awake time in the main application firmware file. They are AWAKE_TIME and SLEEP_TIME. These defines set how long the device stays in active mode and how long before it fires a clock interrupt to wake up. See figure 3 for a diagram of the states that the electrochemistry board can be in. Each time the microcontroller wakes up, a reset is triggered. The number of resets is stored in the backup random access memory (BURAM) for data logging. This memory persists through EM4 sleep but not through full power resets.

As seen in Figure 4, the electrochemical system has power latch circuitry to keep the device powered and to convert the battery voltage to a constant 3.3 volts. The power latch is held high when the device is on. When the battery is connected to the board, nothing is powered up. The microcontroller does not have any power, and no power modes are active. Pressing the button powers the enable pin of the linear regulator, powering on the board. The enable pin of the linear regulator is connected to both the button and a pin on the EFR32BG24. During initialization, the power latch is set high, so the board can hold power to itself without the button constantly being pressed. Additionally, the pins are configured for retention during and after EM4 sleep so all of the GPIO pins stay in the same state. This is especially important for the power latch pin. Otherwise, when the device goes into sleep, the power latch will not hold, cutting off power to the linear regulator.

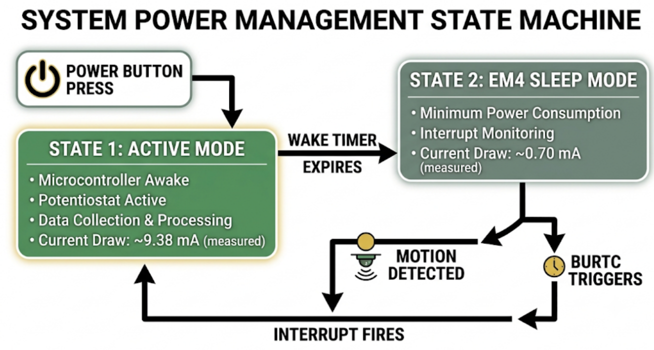

Fig. 3. Power State Diagram

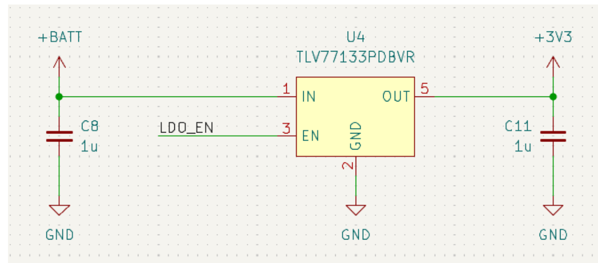

Fig. 4. Power Latch Schematic

IV. Real-Time Clock

When placed in EM4 mode, the EFR32BG24 needs an interrupt to wake up to active mode. This firmware uses two interrupts to wake up the device. The first is the Back-Up Real Time Counter (BURTC). This clock uses less than one microampere [4]. There are three possible sources for this clock. From least power consumption to highest power consumption, they are Ultra Low Frequency RC Oscillator (ULFRCO), Low Frequency RC Oscillator (LFRCO), and Low Frequency Crystal Oscillator (LFXO) [5]. Originally, the ULFRCO clock was selected as the source. This clock runs at 1000 hertz. This clock was found to be inaccurate, so the LFRCO was selected. The clock outputs the time (along with the battery voltage) as seen in Figure 5. While more accurate than the ULFRCO, as Figure 6 shows, this clock was found to have a 27-second-per-hour drift at room temperature. This 27-second-per-hour drift is not ideal for timekeeping. The LFRCO source has a precision mode to increase the accuracy of the clock, but is not supported in EM4 mode and therefore is not appropriate to use in this project. The LFXO is specifically designed with wall-clock timekeeping in mind. Unfortunately, this clock requires an external 32.768 kHz crystal oscillator to operate. The printed circuit board used in this project does not have that crystal, but it can be implemented in a future board revision. The LFXO and LFRCO clocks both operate at the same frequency, allowing a clock source change from LFRCO to LFXO with minimal code revisions.

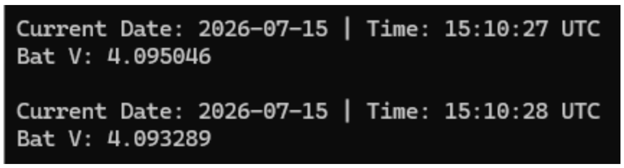

Fig. 5. RTT Terminal Graphic

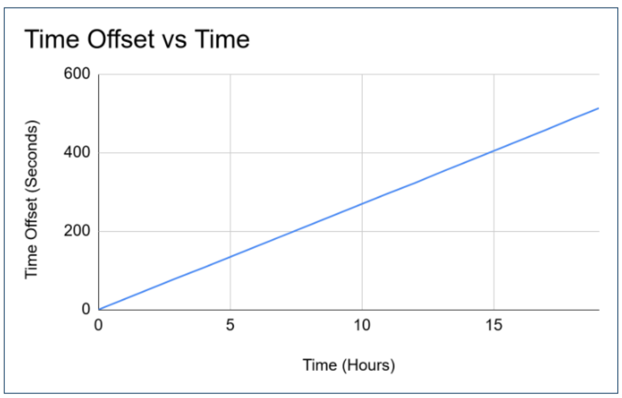

Fig. 6. Time Offset vs. Time Chart

The LFRCO was coded with a 32768 prescaler. This allows the BURTC time counter to increment by one tick every 32768 oscillations, or one second. This gives the BURTC a precision of one second, making time calculations relatively simple. The BURTC is initialized when the board first powers up. The BURTC counts the number of seconds the board is powered on until the time is updated with the current time. This BURTC uses the Unix time standard to track time. This standard counts the number of seconds since January 1, 1970. A mathematical function converts this time to universal standard time. The device was configured to use the RTT terminal interface because it allows bidirectional data. The device prints out time to the RTT terminal in this format: Current Date: Year-Month-Day — Time: Hour:Minute:Second UTC. The time can be updated via an external computer connected to the device. A Python script was developed to retrieve the Unix time from the host computer and send it to the microcontroller via the RTT terminal. The EFR32BG24 takes the time from the RTT terminal and updates its internal clock. Before the device goes into EM4 sleep, the current Unix time plus the sleep time is saved as a 32-bit number. When the BURTC reaches that number, an interrupt fires, waking up the device.

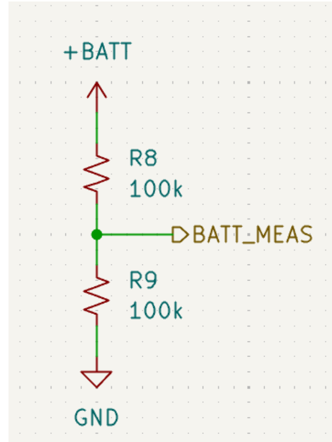

Fig. 7. Voltage Divider Schematic

V. Battery Voltage

As shown in Figure 7, the board has circuitry for a voltage divider to measure the battery voltage. Before the modified firmware, the device featured a battery voltage reading function. However, the voltage reading was very inaccurate compared to the actual voltage measured with a multimeter. The issue was traced down to the ADC clock frequency. The frequency was too fast for accurate readings because the sampling window was too short to fully charge the microcontroller's internal sampling capacitor. The ADC clock frequency was 16 MHz. When the frequency was decreased to 1 MHz, the function reading was accurate. The ADC clock frequency needs to be close to 16 MHz for the electrochemical experiments. Therefore, a new solution was needed. The actual voltage and the function voltage were compared across different points of the battery's voltage range. After calculating the multiple data points, it was found that a multiplier of 2.18 was necessary for the battery voltage function to give an accurate value to account for the fast ADC frequency.

After the voltage function was corrected, a function was made to shut down the device when the battery gets too low. Lithium batteries get damaged when the voltage gets too low. More importantly to this project, the 3.3-volt linear voltage regulator requires the battery voltage to be over 3.3 volts for the output voltage to be a proper 3.3 volts. Therefore, a device shutoff function was added to safely disconnect the power latch once the battery voltage was measured to be 3.35 volts. This prevents the battery from external power drain.

VI. Motion Triggered Wake

When the device is in EM4 sleep mode, Bluetooth is disabled, and there is no direct communication with the device. If the sleep time is set for a long time, there would be no easy way to communicate with the device until it wakes up. The IMU was enabled on this device to provide wake-on-motion functionality. This allows the device to wake up and communicate via Bluetooth LE after someone picks it up. The IMU has two interrupt pins on its integrated circuit package [6]. One of these pins was routed to the PC07 pin on the EFR32BG24. The IMU was configured to send an interrupt to the microcontroller when motion is detected. This wakes up the device from EM4 sleep to active mode. This is a more convenient way to communicate with the device than waiting for a timer to wake it. Additionally, some future applications of this project may include a wearable version. This code allows the device to wake when a person moves so the device can collect data.

VII. Results

This board uses an average of 9.38 milliamperes while in active mode. The mode is important for Bluetooth transmission and data collection. In EM4 sleep mode, the device uses approximately 0.70 milliamperes. This means that the device takes 8.9 days to use 150 milliampere hours while in sleep mode compared to just 16 hours in active mode. When duty cycling the device to be awake for ten seconds and go to sleep for one hour, the device takes 8.6 days to use 150 milliampere hours. See Figure 8 for a visual representation of these numbers. This duty cycling with EM4 sleep allows for multi-day field deployments. Figure 9 shows the current vs time graph of the PCB with the device being in active mode for about a second and sleep mode for about two seconds. The spikes approximately every 100 milliseconds when the graph is high are from the Bluetooth transmission. Figure 10 shows the same graph with the IMU disabled. With the IMU disabled, active and sleep power consumption is lower, but wake-on-motion functionality is disabled.

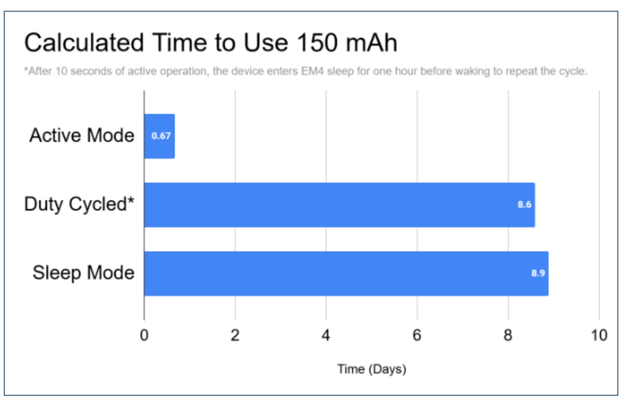

Fig. 8. Battery Life

VIII. Future Work

This project was a continuation of an open-source project. While this outlines many of the feature updates and improvements of this electrochemistry project, there are still improvements that can be made. Many of these improvements can be made by updating the mobile application. When the device is powered on, the time is reset to January 1, 1970. The device needs an external computer to connect to it to update the time.

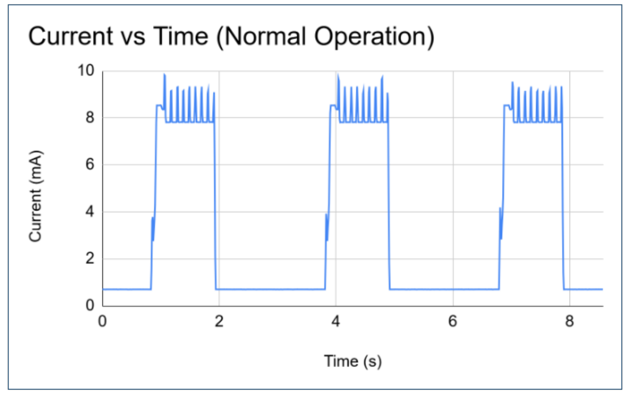

Fig. 9. Current vs Time (Normal Operation)

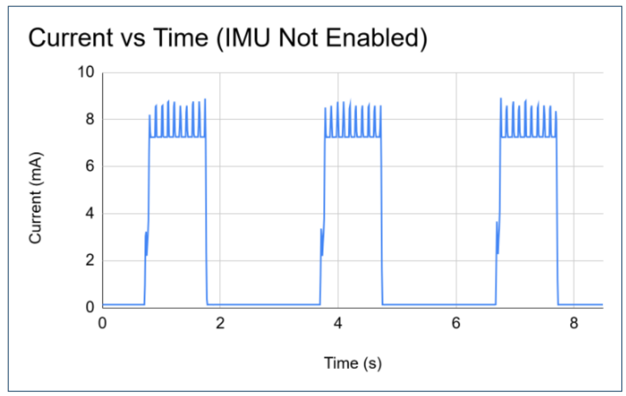

Fig. 10. Current vs Time (IMU Not Enabled)

A future improvement could be an automatic time update via the mobile application or a graphical button. Additionally, the battery voltage can be shown in the application. A battery percentage can also be extrapolated from the voltage and time, or via an external chip.

A necessary improvement is adding an LFXO crystal to the circuit board. This will allow for accurate timekeeping and time-stamping. The firmware is already written to allow this crystal to function with minuscule changes to the code. Unfortunately, a crystal could not be added in this research due to time constraints.

IX. Conclusion

The open-source electrochemical sensor system had limitations in several areas. Battery life was improved by approximately 13 times. The real-time clock and timestamping allow for more advanced data collection and sampling. Motion detection enables functionality for wearable applications. The power-latch-with-button logic allows the device to turn completely on and off in a user-friendly way. All of these features are open-sourced and hosted on GitHub for anyone to implement. These improvements make the device more applicable and attractive for real-world use.

REFERENCES

[1] A. A. Rowe et al., “CheapStat: An Open-Source, ‘Do-It-Yourself’ Potentiostat for Analytical and Educational Applications,” PLoS One, vol. 6, no. 9, p. e23783, Sept. 2011, doi: 10.1371/journal.pone.0023783.

[2] O. S. Hoilett, J. F. Walker, B. M. Balash, N. J. Jaras, S. Boppana, and J. C. Linnes, “KickStat: A Coin-Sized Potentiostat for High-Resolution Electrochemical Analysis,” Sensors, vol. 20, no. 8, p. 2407, Apr. 2020, doi: 10.3390/s20082407.

[3] S. Kukla, B. Uzunoglu, M. J. A. Khondkar, T. Sweeney-Fanelli, S. Andreescu, and M. H. Imtiaz, “Lessons Learned from Designing a Wireless Universal Electrochemical Sensing System,” in Proc. IEEE 16th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON), 2025, pp. 0143–0149, doi: 10.1109/UEMCON67449.2025.11267680.

[4] Silicon Labs, “EFR32BG24 Wireless SoC Family Data Sheet.” [Online]. Available: https://www.silabs.com/documents/public/data-sheets/efr32bg24-datasheet.pdf

[5] Silicon Labs, “EFR32xG24 Wireless SoC Reference Manual.” [Online]. Available: https://www.silabs.com/documents/public/reference-manuals/efr32xg24-rm.pdf

[6] STMicroelectronics, “LSM6DSV: 6-axis inertial measurement unit (IMU) with embedded sensor fusion, I3C, OIS/EIS for smart applications,” DS13476, Rev. 5, Aug. 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/lsm6dsv.pdf