FRAM BASED LOW POWER SYSTEMS FOR LOW DUTY CYCLE APPLICATIONS

By

Cody A. Gossel, B.S.

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electrical Engineering

University of Alaska Fairbanks

May 2019

Abstract

Ferro-Electric Random Access Memory (FRAM) is a leap forward in non-volatile data storage technology for embedded systems. It allows for persistent storage without any power consumption, fulfilling the same role as flash memory. FRAM, however, provides several major advantages over flash memory, which can be leveraged to substantially reduce sleep current in a device. In applications where most of the time is spent sleeping these reductions can have a large impact on the average current. With careful design sleep currents as low as 72 nA have been demonstrated. A lower current consumption allows for more flexibility in deploying the device; smaller batteries or alternative power sources can be considered, and operating life can be extended.

FRAM is not appropriate for every situation and there are some considerations to obtain the maximum benefit from its use. An MSP430FR2311 microcontroller is used to measure the performance of the FRAM and how to structure a program to achieve the lowest power consumption. Clock speed and instruction caching in particular have a large effect on the power consumption and tests are performed to quantify their effect.

Two case studies are considered, a feedback control system and a data logger. Both cases involve large amounts of data writes and allow for the effects of the FRAM to be easily observed. Expected battery life is determined for each case when the sample rate is varied, suggesting that average operating current for the two solutions will nearly converge when the sampling period exceeds 1000 s. For sampling periods on the order of one second operating current can be reduced from 15.4 µA to 730 nA by utilizing FRAM in lieu of flash.

Table of Contents

Page

List of Figures

List of Tables

Acknowledgements

I first thank my advisor, Dr. Dejan Raskovic, for his support and advice during my research and writing. Without his input throughout the process this work certainly never would have come to fruition. Also appreciated are the contributions of Dr. Denise Thorsen and Dr. Vikas Sonwalkar who provided valuable feedback while serving on my graduate committee.

I also thank my mentors and coworkers at Johnson Space Center. My mentors, Dr. Michael Callahan and Rachel Fisher, provided a remarkable amount of support during the early stages of my silver biocide research. Dr. Callahan spent a tremendous amount of time reviewing concepts and data with me, and Rachel's excellent project management skills helped to keep research tasks performed at JSC moving forward smoothly. A special thanks goes out to Otto Estrada, Stacey Moller, Chris Carrier, and Robert Johnson, all of which provided lab support during the water bus testing of the silver biocide control system. My JSC branch leadership, Nichole Williams and Jonathan Bowie, were always very understanding and I am grateful for the research opportunities they provided to me there.

Finally, I would like to thank Chris Wilson and Matt Bishop, who generously volunteered their time to review my writing and provide valuable input.

Introduction

Research Background

The project began with a specific microcontroller implementation intended to dose biocide into a spacecraft water system. From there, it was further expanded into specific research into general programming and design techniques for low power and low energy operation, particularly those pertinent to FRAM.

The original low duty cycle application from which the follow-on research was derived is a feedback controller designed to regulate the concentration of ionic silver in a spacecraft water system. Ionic silver is an effective biocide for residual treatment of water; however, it comes with the drawback of not being storage stable. Spacecraft water systems are traditionally constructed using metallic materials. In such a system, there are two primary sources of losses – gradual plating of the silver onto metallic surfaces within the water system [1] and occasional sudden losses when fresh water is introduced to the system to replenish water that has been used. Ultimately, this contributes to a need to monitor and actively maintain the ionic silver biocide concentration within the water system.

Long duration, unsupported missions require large water storage tanks, which provide a potential breeding ground for bacteria. At present, approximately an 85% closure of the water loop has been achieved and the current goal is to increase the closure to 98% [2]. In practical terms, this would mean that for 1 L of water consumed there would be 20 mL of losses. These inefficiencies require spacecraft to carry significant amounts of additional water on long duration flights. At 5 L of daily usage a three-month flight would need to carry 67.5 L of water to offset these efficiency losses. The water system consists of a large tank from which only relatively small amount of water will be drawn off at any given time. This arrangement creates a buffer, which helps to naturally smooth out large swings in the concentration. This allows for the use of a weak plant in the control system as well as reducing the need for high frequency monitoring.

To prevent the growth of deleterious organisms within the drinking water it is necessary to maintain an appropriate biocide concentration [3], to offset the losses a control system is required. Out of desire for efficiency, the system would be designed to minimize silver losses onto the metal surfaces. The gradual nature of the losses, taken in combination with the buffer effect of the tankage yields a very long time constant. Because the time constant of this system is very long this allows for a control system to take infrequent measurements, which is an ideal situation for a low power system.

A test device was designed to demonstrate control of the silver concentration using an ion selective electrode in conjunction with a microcontroller. The onboard ADC was used to implement a simple on-off controller and drive a dosing electrode in a recirculating, closed loop water system. The system proved ultimately successful, maintaining the silver concentration in the metallic system over a long duration. This controller is the device used to perform the testing in the Silver Biocide Feedback Controller Case Study chapter.

One interesting facet of electrolytically dosing silver biocide is that a very small amount of energy is required to drive the chemical reaction. A 0.7 V potential with a 300 μA had been shown to introduce electrolytic silver at a rate of 5.4 μg/s [4]. The combination of small energy requirements to dose silver, the long time constant of the system allowing for infrequent checks, and the success of the simple feedback controller suggested that a very low power solution could be implemented. This idea motivates the design and testing of the low power feedback controller described in the Low Power Redesign section.

A challenge to overcome with the dosing system relates to the functionality of the plant. To achieve wear leveling across the device it is necessary to alternate the direction of the dosing current. To achieve the lowest possible average current consumption the microcontroller is placed into a full shutdown mode and awoken by an external low power timer. Utilizing the shutdown mode of the controller does not allow for RAM retention which makes it difficult to track the direction the current was flowing during the last activation period.

Traditional nonvolatile memory available on a microcontroller, such as flash or other EEPROM technologies, require large amounts of energy to write and have a prohibitive wear life. Erasure of flash memory, which is necessary before writing new data, requires a large voltage compared to read or write actions [5]. Typically, this will require an onboard charge pump to generate the necessary voltage and in most implementations erasure will be performed on an entire block of memory. Energy consumption for a program action can reach 10 times the energy required for a read, while erase energy consumption can reach as high as 50 times the read energy [6]. A relatively recent advance in onboard memory, FRAM, allows for nonvolatile storage with a small energy premium over static random access memory (SRAM) and a nearly unlimited wear life [7]. The characteristics of FRAM make it a logical choice when the use of a shutdown mode is desired but there is a need to persist changing data through sleep cycles.

Considering the technology available and the problems posed by the silver biocide dosing system motivated the choice of FRAM for the low power redesign. Performance testing of this controller lead naturally into the exploration of the characteristics of the FRAM implementation on the controller used as well as general design techniques for minimizing power consumption when using FRAM.

Research Objectives

The research in this thesis is focused on three areas:

1. Proving the efficacy of a closed loop feedback controller in a silver biocide system.
2. Developing and testing an extreme low power solution to implement the feedback system.
3. Exploration of FRAM performance and its applicability to other systems.

A scaled down version of a spacecraft water bus was constructed and used to prove that the feedback controller would maintain control of the system. The system was operated for approximately a two-month period and samples were drawn every working day for external analysis via an Inductively Coupled Plasma Mass Spectrometer (ICP-MS). Concentrations between 200 and 400 ppb have been shown to be effective in a biocidal role, so the goal was to ultimately prove that control is working within these bounds.

No substantial electrical testing was performed on the first controller prototype, and very little effort was put into optimizing it for energy efficiency. The second revision of the controller used the now proven technique of using a feedback controller to maintain the concentration and optimize the solution for low power consumption. Due to limitation of resources, this controller was never able to be tested against the mocked-up water bus. The measurement and dosing action of the controller was instead simulated, and energy consumption was tested in detail.

Finally, the low power version of the control system was used to perform some additional testing on FRAM and explore design concepts relevant to low power designs. In the FRAM-based microcontroller the FRAM storage is used for both programs and data. FRAM reads are slower and consume more energy than SRAM reads, which introduces new code concerns to achieve the best low power performance.

FRAM Technology

FRAM represents a leap forward in nonvolatile storage for embedded systems. FRAM combines a read/write speed on the same order of magnitude as SRAM, nearly unlimited write endurance, and significantly lower power consumption than flash when writing [8], [9].

FRAM technology can be leveraged alongside total shutdown operating modes in embedded systems to reduce idle current to previously unattainable levels. Retaining data in SRAM requires a small but constant current into the device to hold the SRAM transistors latched into their current state. Taking the MSP430F5438A as an example, 1.2 μA [10] are required for data retention. Placing the controller into total shutdown, where no SRAM retention occurs, will consume only 100 nA, an order of magnitude reduction in current. The primary challenge in utilizing shutdown is the loss of data retention. The controller either needs to be able to infer any necessary information after waking up, or to save the necessary information into nonvolatile memory. In addition to the lack of SRAM retention, waking from shutdown requires more wakeup time than regular low power modes. During this wakeup period, the controller will exhibit current consumption similar to active mode but will not perform any useful work.

Historically, most microcontrollers utilized flash or a similar technology for non-volatile data storage. In many cases, using flash to store data is not practical due to the limited read/write endurance. Texas Instruments (TI) estimates the cycle life of their flash devices at 100,000 cycles [11]. Flash devices intended for use in applications with frequent read/write cycles, such as a computer SSD, vary write locations within the flash memory such that all locations fail at approximately the same time. This extends the life of the device at the cost of additional complexity but wear leveling logic is impractical to implement in an embedded system.

Assuming a cycle on the flash occurs once per minute, a typical flash cell might last about 70 days based on TI's estimate. This is likely to be the limiting factor in the low power system, because a system with an idle current near 100 nA is only useful when targeting extended duration applications. Flash cycle life is a concern if it is necessary for the system to retain information. If the system can be designed such that no data needs to be passed forward through the shutdown period this limitation is irrelevant, however this is seldom the case.

While FRAM provides several benefits over flash, the write endurance is the key factor in enabling the easy use of shutdown mode. FRAM can be embedded in the microcontroller itself, provides nonvolatile storage, and can be accessed as easily as SRAM. Variables that need to be persisted through shutdown cycles can be located in the onboard FRAM and will be retained. Even in a situation where the limited endurance of flash can be tolerated, FRAM still provides a reduction in energy consumption when writing due to the ability to write a single byte and the lower voltages necessary to write.

System Design for the Lowest Possible Power

A practical microcontroller will include both volatile and nonvolatile memory. The program run by the controller is in the nonvolatile memory so that reprogramming is not required after a loss of power. The volatile memory is used for allocating variables in the program, but usually it is also possible to execute code from volatile memory. SRAM is the most common storage technology used for volatile storage in microcontrollers [12]. A small, but constant current is required at all times to prevent the loss of data contained in the SRAM [13].

For a low duty cycle application, the first steps taken to reduce power consumption involve deactivating modules in the controller. “Putting a controller to sleep” typically means disconnecting unused circuitry from the power supply and gating off clocks where possible. In an extreme case, the clock sources within the controller might be completely disabled and only an external event on one of the microcontroller pins can wake the device. A common thread between all sleep modes is that the SRAM is supplied with power so that data is retained through the sleep cycle. Current consumption on the order of 400 nA can be achieved by disabling all clocks and peripherals while still powering the SRAM.

Cutting power to the SRAM is the final step in reducing idle power consumption. When the SRAM power is cut the controller is said to be in “shutdown” mode. Shutdown introduces additional difficulties as it will cause the program state, held in the form of values stored in variables or registers, to be lost. Recovering from shutdown is an identical process to power up, because all required information will need to be loaded from nonvolatile memory. Any values stored in SRAM that need to be retained must be written into nonvolatile memory prior to entering shutdown. Shutdown currents around 50 nA are readily obtainable.

Another challenge to using a shutdown mode is that no clocks will be available on the controller itself [14]. If the application can allow for a pin to be asserted when the microcontroller needs to wake up, this is not an issue. In a periodic sampling application, one solution is to use an external real-time clock (RTC) to provide a wakeup signal to the controller.

RTCs in the 30 nA range are readily available. Microcrystal's RV-1805-C3 or TI's TPL5010 are two examples [15] [16].

The layout of the circuit board and connections to the microcontroller should also be considered. With a target current draw of 100 nA or less for the microcontroller and RTC, currents that could normally be neglected become significant contributors. Capacitor leakage should be considered. Where possible the use of decoupling capacitors should be eliminated and when necessary a low leakage style of capacitor should be used. MLCC capacitors with small footprints are ideal for this application [17]. If switches are used, a double throw configuration should be considered instead of a single pole with pull-up or pull-down resistor. The normally open and normally closed contacts of the switch can be weakly pulled to Gnd and Vcc rather than allowing a situation where one of the switch positions causes a constant current to flow.

Software Concerns

The design and implementation of the software can have a large effect on the current consumption during processing. From a logical standpoint, it is beneficial to reduce the number of cycles required during each processing cycle. Floating point arithmetic in particular should be avoided if at all possible. In many instruction set architectures (ISA), including the MSP430 ISA, a "branch if not zero" instruction is available. Constructing loops to count down instead of up allows for use of this instruction instead of performing a subtraction and checking the sign bit, saving a cycle for every iteration of the loop [18].

SRAM requires less energy than FRAM or flash to read. There is a tradeoff to consider when a section of code is to be executed many times. Copying sections of the code into SRAM requires an additional read for every instruction but will reduce energy consumption for every following read. If FRAM is used, there is an additional concern; because FRAM needs to be refreshed after every read there is a limit on how quickly instructions can be fetched. For the TI MSP430FR2311 the FRAM operation is limited to 8 MHz. Operating at core speeds faster than this requires processor wait states to allow the FRAM refresh process to occur.

Because of this limitation with FRAM the MSP430FR2311 includes an SRAM based instruction cache with the FRAM controller. Since the program is stored in the FRAM, the 8 MHz limit would also limit the maximum speed of the microcontroller as instructions could only be fetched at 8 MHz. Including the cache allows for instructions to be read from FRAM 4 at a time in parallel and stored in SRAM. Provided the instruction fetches result in cache hits, the microcontroller will not be limited by the limited FRAM speed.

The inclusion of the cache primarily impacts power consumption in two ways. Regardless of the clock frequency used, reads from the instruction cache will require less energy than reads from the FRAM itself. If the clock frequency is above 8 MHz, cache hits will also prevent energy from being wasted on processor stalls. It is beneficial to optimize the program to maximize cache hits at any speed, and critical when operating above 8 MHz.

In the MSP430FR2311 the FRAM cache is a two way set associative cache with a 64-bit line length [18]. Different FRAM implementations may vary, and the program should be designed with the actual cache size in mind. Some simple practices, such as aligning inner loops to maximize cache hits, can drastically reduce power consumption.

Work Performed and Contributions

The work for this thesis was performed both at NASA Johnson Space Center (JSC) and in the labs at UAF. I developed a concept to dose ionic silver into water for microbial control by using electrolysis with a feedback controller to maintain a set concentration. While at NASA, I designed and built two prototype embedded systems and used them to perform testing on the system concept as well as to measure current consumption for a potential low power system.

The first prototype consisted of an MSP430 Launchpad and a custom expansion board designed, constructed, and programmed by me. I formulated a test using this prototype to prove out the feasibility of the idea by building a subscale mockup and attempting to control the concentration. I took daily samples from the test stand and measured the concentration to ensure that it remained close to the set point. The results of this test were ultimately published in a paper submitted to the International Conference on Environmental Systems (ICES) [19].

The second prototype was also designed, constructed, and programmed by me and was developed as a low power iteration on the first prototype. This prototype was brought back to UAF for further current consumption testing. An implementation like the feedback controller benefits from the FRAM on the microcontroller, which allows for easy durable retention of values when very low power modes are used. Comparisons between FRAM and flash are made to provide some insight into when each technology is appropriate.

FRAM Energy Consumption Testing

This chapter is focused on the electrical performance of the second iteration feedback controller. Emphasis here is placed on the low power characteristics of this specific device and software implementation of the feedback controller.

Two general methods are used to measure the power consumed by the controller. The first is based on altering the controller software to force it into specific modes of operation to enable measurement. The second is based on devising a system that allows measuring the energy consumption of the system more directly.

Finally, an in depth look at the ramifications of the inclusion of FRAM onto the microcontroller is performed. Performance of the instruction cache on the controller is important to the overall energy performance of the device. Included as a mitigation to one of the primary drawbacks of FRAM, slow read speed, it serves a vital role in preventing processor stalls. While in any application a processor stall is detrimental to processor speed, in an embedded system there is an additional penalty in terms of energy consumed.

Low Power Controller

The design of the controller is intentionally made very simple. There are only two IC's on the circuit board, an MSP430FR2311 microcontroller and a TPL5010 timer. Only two bypass capacitors are used, both are multi-layer ceramic capacitors (MLCC) which have relatively low leakage current. For debugging and setup, there are 4 LEDs on the circuit board, and two momentary push buttons. The only external digital interface on the board is the JTAG connection used for programming. The schematic for the test circuit is shown in Figure 1.

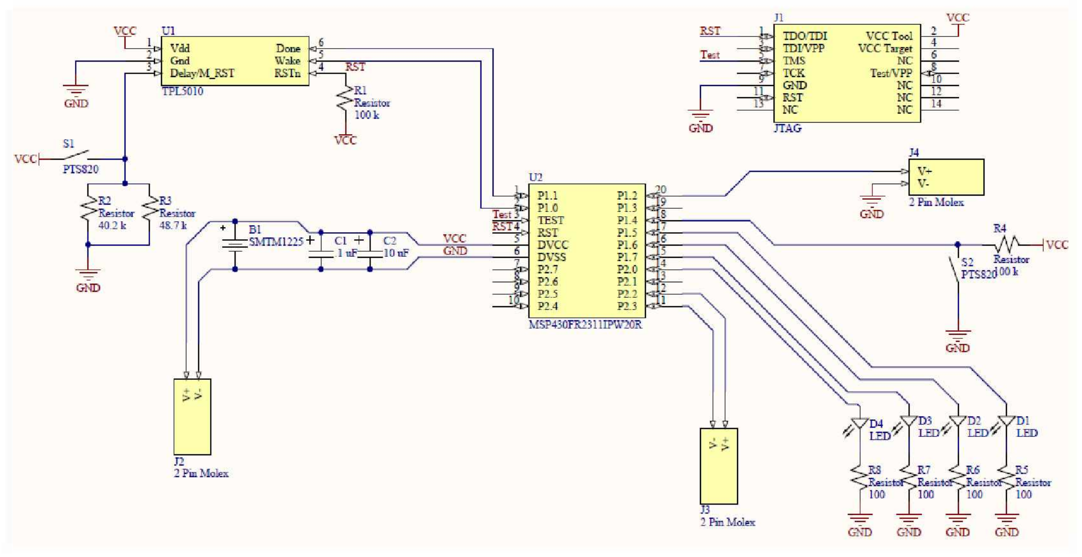

Figure 1: Test Device Schematic

The core of the controller are the two ICs, U1 and U2. U1 is a Texas Instruments TPL5010 low power watchdog/timer IC. It contains a low power, low frequency oscillator and can be programmed to provide a wake signal at a specific interval. This interval is selected by connecting the appropriate resistance to the M_RST pin; a one minute interval is selected here. When one minute is elapsed, the IC asserts the Wake pin, and waits for the controller to respond through the Done pin. The IC implements watchdog functionality in addition to the timed wakeups. If the controller fails to respond before the one minute interval elapses for the second time, the reset pin on the controller is asserted. A photograph of an assembled test device is shown in Figure 2.

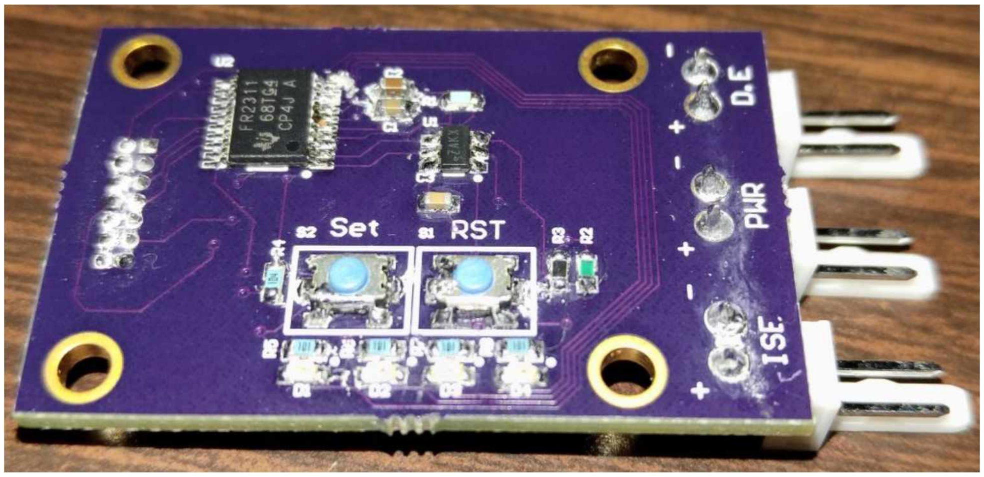

Figure 2: Photograph of Test Device

Using the external timer IC in conjunction with the MSP430 allows for placing the microcontroller into shutdown while retaining the ability to wake periodically. The timer and microcontroller are both selected to minimize current during shutdown. The TPL5010 consumes only 35 nA [16] continuously and is a readily available and inexpensive component. At the time of writing, the MSP430FR2311 possessed the lowest shutdown current of the available FRAM based MSP430 controllers and was selected for the feedback controller.

Most MSP430 microcontrollers are equipped with the capability to vary their clock speed. The FR2311 can operate at core speeds of up to 16 MHz and can either source a clock signal externally or generate one internally using a tuned RC circuit. The clock is fed through a digitally controlled oscillator, which allows the frequency of the clock source to be multiplied or divided to achieve the desired operating frequency.

Energy Consumption Test Methods

Shutdown Current

Measurement of the current was performed by using a series resistor in line with the supply of the test device. An Agilent 34420A Micro-Ohm meter was used to precisely measure the resistance of a 100 Ω 5% resistor. The 34420A was used again to measure the voltage over the resistor, and Ohm's law was used to compute the current. The 33420A allows for more accurate measurements of very small voltages and resistances. At 1 mV range accuracies of 0.007% can be expected [20]. The tradeoff with the 34420A is that it lacks the ability to take a current measurement directly, requiring the use of the series resistor.

By this method, the shutdown current was determined to be 72 nA. The datasheet value for the MSP430 in this configuration is 32 nA at room temperature, and the TPL5010 datasheet current is 35 nA. The measurement is slightly higher, but in line with the expected values from the datasheet.

Active Mode Current

The active mode currents are large enough to be easily measured directly using the 34401A, so the measurements are performed by placing the meter in series with a power supply. A second meter is put in parallel with the power supply to allow for more precise measurement of the supply voltage. Agilent 34401A multimeters are used for both measurements. The measurement circuit is shown in Figure 3.

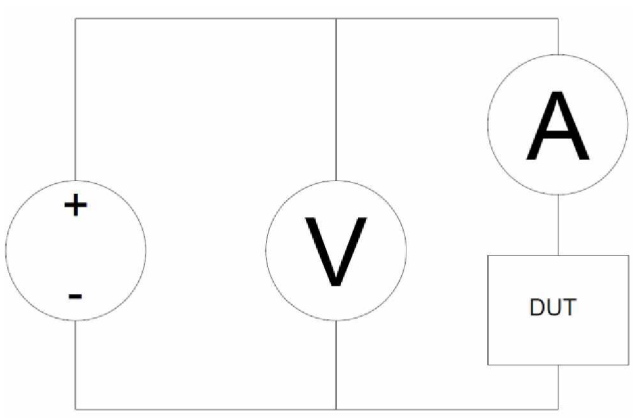

Figure 3: Active Mode Current Measurement Circuit

While taking measurements, the voltmeter is used to monitor the power supply output voltage and adjust it to as close to 3 V as possible. The test device is connected to the circuit with no battery and no other external connections. The 34401A is set to measure DC current with an integration period of 1 power-line cycle (PLC). The standard mains frequency in the United States is 60 Hz, which causes the meter to integrate the current for 1/60th of a second. Like most precision multimeters, the 34401A uses an integrating ADC to measure current [21]. The ADC integrates the voltage over the integration time and then discharges the integrator using an internal voltage source. The voltage of the internal source is known, so the time it takes to fully discharge the integrator can be measured to determine the current. The current draw of the microcontroller will change rapidly during each operating cycle, but the relatively long integration period will average out these changes.

Clock speed is a factor in current consumption, so tests are conducted at 4 different core frequencies: 1 MHz, 4 MHz, 8 MHz, and 16 MHz. The core clock signal is output to an external pin on the microcontroller so that it can be measured, and an oscilloscope is used to measure the core clock frequency. The core clock frequency is then adjusted until it is within 1% of the desired test value.

There are five benchmark programs used with the microcontroller for measurements, each one intended to demonstrate a specific scenario with regards to FRAM caching. The body of each program is NOP (no-operation) instructions, which perform no actual computing work. This choice is made to maintain a consistent baseline throughout each test, even if the tests are not consistent with a real-world scenario. Depending on the instructions being executed, the average current consumption can change and using NOPs avoids this. Ultimately, the difference in current consumption due to FRAM cache hits is the value of interest, rather than any individual current value.

When considering the impact of the cache it is important to understand the memory topology of the controller. FRAM is deployed in conjunction with SRAM in a similar fashion to how it is deployed in a traditional flash-based controller. The MSP430FR2311 is equipped with 4 KB of FRAM and 1 KB of SRAM. By default, the compiler will place stack and most global variables into SRAM; a compiler directive needs to be provided to place a variable into FRAM.

The FRAM is used for storage of both the program and variables that are meant to be persisted. Persistent data in this context is generally meant to denote data which will survive a power loss. Often for an embedded system persistent memory will consist exclusively of flash storage. There are however many special cases in specific applications.

Just as with a typical flash-based controller, the program code will reduce the space available for persistent storage of program data. The FRAM cache is a unique addition to FRAM controllers. While flash can be read at clock speed in most cases, the destructive readouts of FRAM limit potential clock speed. While occasional wait states for persistent variables will have a small impact on performance, the constant reads from FRAM to access program instructions would cause continual wait states. To maximize the effect of the FRAM cache it is only used to cache instructions. Instructions are very likely to be read sequentially and would cause the largest performance hit if read directly from FRAM.

The cache on this particular microcontroller is a two way set associative cache with a 64-bit cache line (or block) size and two blocks per set. MSP430 instructions are 16 bits long, so four instructions will fit into each cache line. On every FRAM cache miss the entire line is populated in the 125 ns FRAM operation time. With this cache design, a loop of up to 16 instructions can be executed with no cache misses after the first loop iteration. Figure 4 shows the layout of the FRAM cache. The numbers in each block are the offset of each instruction; the first instruction, which has offset zero, occupies byte zero and one in the cache block; the second instruction, occupying byte two and three has offset two, and so on.

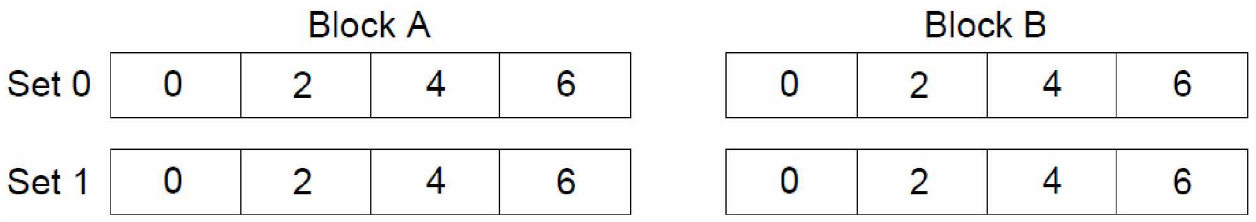

Figure 4: MSP430FR2311 FRAM Cache Layout

The cache is byte addressable, but for simplicity, the illustration is word based to match the size of an MSP430 instruction. Since there are two sets, every instruction address will be divided between the two. Since there are eight bytes in each block the value of the 4th bit can be used to determine which set a memory location will belong to. Figure 5 illustrates how instructions are mapped into the cache.

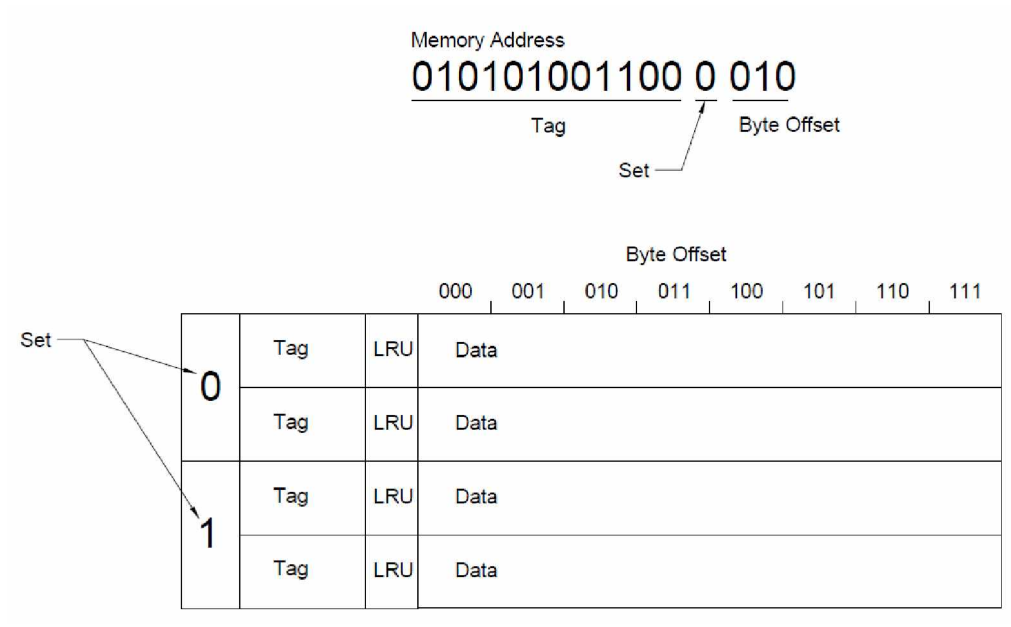

Figure 5: Address Mapping in Instruction Cache

The three least significant bits of the address will correspond to the byte offset within the cache block. The offset is not explicitly stored within the cache. During a cache read multiplexer circuitry will use the final three bits of the address to select the appropriate byte in the cache. Bit four determines which set in the cache the line of eight bytes will belong to [22]. The remainder of the address is stored in the cache as the tag. These tag bits are used to determine whether or not a specific value is present in the cache. Finally, a single bit is included in the cache to track the least recently used value. This bit is labelled as LRU on the diagram and is used to indicate which block should be replaced on a cache miss.

When executed in systems that have a cache, most programs will benefit from inherent spatial locality without special consideration by the user. Even for a small cache, hit rates of 70% are common [23]. Instruction accesses are typically sequential in nature so a relationship between cache size and average instructions per jump will emerge. This particular cache contains four instructions in each block. If the program consists mostly of linear segments that are at least four instructions long, the cache hit rate will approach 75%. Frequent execution of loops with less than four instructions will disrupt the spatial locality benefit and reduce the cache hit rate.

In a strictly linear program, the cache hit rate is static; all misses are compulsory misses and no optimization is possible. When loops are introduced into the program cache hit rates can begin to vary. A sufficiently small loop can provide a 100% cache hit rate after the first iteration of the loop by preventing capacity misses. If the loop is too large to fit completely in the cache capacity misses will begin to occur.

A balance must be struck between cache size, associativity, and energy efficiency. Increasing either cache size or associativity will improve performance. A larger cache size will allow more data to be stored. Storing more information in the cache increases the likelihood that information will already be present when needed. This can come in the form of larger blocks or more blocks. Larger blocks will leverage spatial locality to reduce misses by caching more data at once. More blocks will leverage temporal locality to reduce the probability that a block that is later needed will be replaced. More associativity prevents blocks from competing for the same cache set and being replaced [24].

Device Specific Caching Considerations

Tests directed at the FRAM performance are confounded by limitations specific to the MSP430 platform. In any program, most memory accesses will be to retrieve instructions, which is why TI implemented the instruction cache.

To derive meaningful results it is necessary to understand and compensate for effects specific to the MSP430 core, and not intrinsic to FRAM or caching systems in general. Jump instructions cause some difficulty when attempting to measure caching effects.

For this microprocessor architecture, a jump requires two cycles to complete [18]. While completing the jump, the instruction immediately following is requested from the cache even though it will not be executed. This is a consequence of the implementation of the jump command in the MSP430 architecture. Texas Instruments indicates that the CPU implements a pipeline which complicates the fetching and execution of instructions. Even though the CPU will determine a jump has occurred and appropriately resolve the pipeline hazards the first stage of the pipeline will still request the next instruction from the FRAM controller.

This effect can be demonstrated by constructing a loop which would normally cause no cache misses after the first iteration and varying the location of the jump instruction in the eight-instruction cache line. If the jump is the final instruction in the cache line, additional misses will occur when the following unexecuted instruction is inadvertently fetched by the pipelined CPU.

The same test circuit as shown in Figure 3 is used to measure the current. Two test programs are used, each aligned in memory to fit completely into the cache. Program A has the jump as the final instruction in the cache, and Program B has the jump as the second to last instruction in the cache. Both programs are shown below:

```
Program A    nop
             nop
             nop
             .
             .
             .
             nop
             jmp    Program A
```

Figure 6: Test Program A

```
Program B    nop
             nop
             nop
             .
             .
             .
             jmp    Program B
             nop
```

Figure 7: Test Program B

Each program is 16 instructions long, which is enough to fully populate the cache. In theory, neither program would exhibit any cache misses and would draw an identical current, but measured current for Program A is significantly higher than Program B. Because the jmp command is in the final position in the cache line the pipeline will fetch the following instruction, even though it will not be executed. As far as the FRAM cache is concerned the loop is now effectively 17 instructions long and too large to fit completely within the cache. This occurs despite the fact that only 16 of these instructions are ever executed. The measured currents are summarized in Table 1.

Table 1: Jump Test Program Currents

Program	Measured Current (μA)
A – Jump at end of cache	239.3
B – Jump set back 1 instruction	201.6

The secondary effect of the jump command is to create an effect that artificially decreases the current. Since two cycles are required to complete the jump, even if every jump causes a cache miss there will be an extra stall cycle between them. The addition of these stall cycles can give the illusion of a 50% miss rate, even in a situation where every executed instruction results in a miss.

Test Case 1

The first test case is a loop where the loop length is twice the cache size. This scenario is identical to the performance that would occur in an infinitely long linear program, where the caches provide no benefit from temporal locality. The caches will still yield a benefit through spatial locality however, as instructions are executed in order and read four at a time. Here every 4th instruction will cause a cache miss, every time the program counter enters a new cache block that is not already populated with the appropriate instructions. The first instruction in each group of four will cause a cache miss, and the next three instructions will be cache hits yielding an overall cache hit rate of 75%. Figure 8 shows an abbreviated version of the program. There are 26 NOP instructions in place of the ellipses in the actual program, for the total of 32 instructions, twice the cache capacity of 16 instructions.

```
TestCase1    nop
             nop
             nop
             .
             .
             .
             nop
             jmp    TestCase1
             nop
```

Figure 8: Test Case 1 Program

Note that the jump instruction is not the final instruction in the program. The MSP430 platform implements instruction prefetch, and a jump or branch instruction takes two cycles to complete. Placing the jump on the last line of the program will result in an additional cache miss on each iteration of the loop as the instruction following the jump is prefetched. The final NOP is never actually executed due to the jump, however the two cycles required to execute the jump causes the loop to take 32 cycles to complete.

The 32-instruction length of the program ensures that the cache is saturated and values are replaced during each iteration of the loop – there are 16 values from each set, and only eight can be stored. At the end of the loop, the cache will be filled with the instructions from the last 16 looped instructions, 8 in each of the two sets, which will force cache misses as the loop is restarted.

Test Case 2

The second test case is meant to verify that least recently used replacement is faithfully implemented. A loop that is one and a half times the cache length is created. The program is identical to the one used for test case one, except that there are only 24 instructions instead of 32.

If least recently used replacement is implemented this loop will result in identical current consumption to the loop in test case one. The first 16 instructions in the loop saturate the cache, and then the final eight instructions will replace the first eight instructions in the cache. As the loop restarts the first eight instructions will then replace the middle eight instructions, and the pattern continues with no cache line hits.

If least recently used replacement is not implemented, and random replacement or some other technique is used, then there will be a difference in current between test case one and test case two – the probability of an instruction line being present in the cache will be different from the first test case.

Test Case 3

This test case is a 16-instruction loop aligned to maximize cache hits. This means the first instruction of the loop is at an address ending in zero or eight (i.e., the last three binary digits of the address are zero) so that the loop breaks on an instruction line. The program is constructed similarly to test case one, but with a loop length of 16. Those 16 instructions will fit completely within the cache provided the instructions are aligned. Figure 9 illustrates an aligned loop.

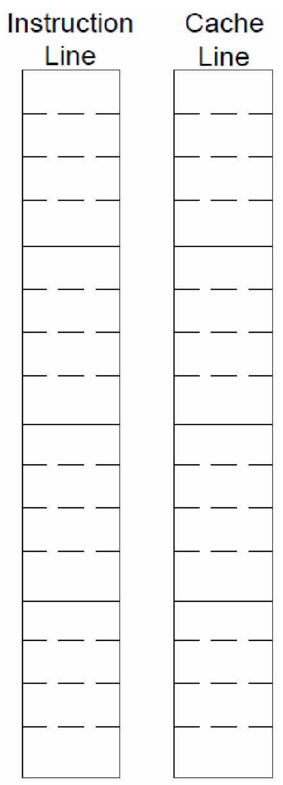

Figure 9: Aligned Loop Example

The blocks in each column represent a line of four instructions. Since each line of instructions in the loop is placed such that they will be placed into a single cache slot, after one iteration of the loop the entire loop is stored in the cache and will result in a 100% cache hit rate.

Test Case 4

This case is similar to test case three, but this time the loop is intentionally misaligned. Figure 10 shows the instruction lines against their respective cache slots. The numbers given to the instructions in the figure are sequence numbers, not addresses. When instruction zero is loaded, it will go into one of the slots allocated for set zero. Since the instruction is not aligned, it will go into word three of the cache slot and two unused instructions will be loaded into the first two words. There are five cache slots required to completely store the loop, and only four are available. In this example, assuming the top cache slot in the illustration is associated with set zero, there are three lines mapped to set zero and two lines mapped to set one.

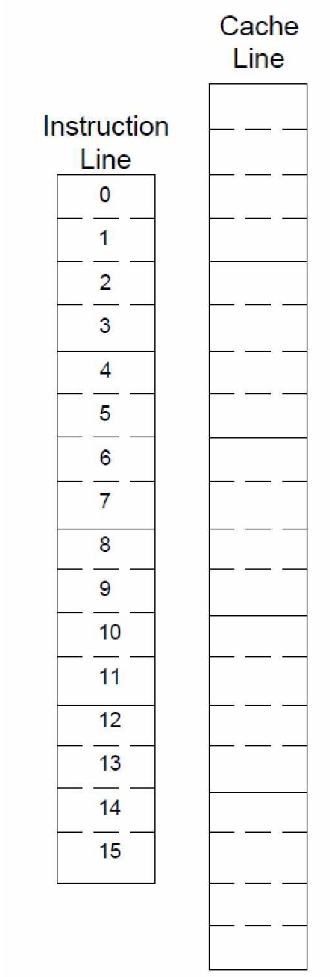

Figure 10: Misaligned Loop Example

Instructions zero, six, and 14 map to the same cache set. Because of the least recently used cache logic each of these instructions will result in a cache miss, because reading each instruction will cause the next one to be replaced in the cache. Here there will be three cache misses for 16 instructions, for an 81.25% cache hit rate.

Test Case 5

The final test case is intended to demonstrate the worst case scenario. The program consists of a series of jumps spaced out by NOP instructions so that the cache provides no benefit from temporal or spatial proximity. Each jump is followed by 7 NOPs, which means every executed instruction will cause a cache miss. In terms of instructions executed from the cache the hit rate is 0%, but each jump instruction requires two cycles to execute. The stall caused by each jump instruction will prevent the impending FRAM access from occurring immediately, resulting in a lower average current than would be expected from a 0% hit rate. FRAM accesses will occur with 50% of the CPU cycles, and the remaining 50% of the CPU cycles will be spent stalling while these jumps are processed.

This is a worst-case scenario for instruction caching, but it is unlikely to occur in a realistic program. Generally, there will be some instructions between jumps that will reintroduce some linearity into the program and reduce the cache miss rate.

Energy Consumption Test Results

Measured values are listed in Table 2.

Table 2: Energy Consumption Test Values

Operating Current (μA)

Clock Speed (MHz)	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
1	247	249	201	238	288
4	522	533	340	488	683
8	885	902	520	816	1200
16	1537	1565	820	1400	2190

The first item of note is that the difference in operating current between test case one and test case two is at most 2%. This suggests that least recently used replacement is implemented in the cache. The small difference can be explained by the increased number of jump instructions relative to the NOP instructions used to form the body of the program. Different instructions will result in different operating currents and it is likely that a NOP instruction, which does not change any registers or memory values, is relatively low in current draw.

Test case three exhibits the lowest operating current. This is not surprising as test case three yields a 100% cache hit rate; it is a construction of an ideal case for the caching system implemented in this MSP430. Due to this fact, it will be used as a baseline when computing energy expended on FRAM reads.

Test case four misaligns a loop in instruction memory to achieve an 81.25% cache hit rate. As expected, the operating current lies between the values for the 100% cache hit rate and 75% cache hit rate scenarios.

Test case five is essentially the opposite of test case 3; it is a program designed to completely nullify the benefit of the cache by consisting entirely of jumps. Its operating current is reflective of this fact – significantly higher than any of the other test cases.

Current measurement results are shown in Figure 11. As expected, the trend appears nearly linear. There are some caveats to consider when creating a set of lines such as shown in the plot. First is the effect of stalls caused by the jump instruction. Each jump will be followed by a stall cycle, which will yield a current consumption similar to a NOP cycle. This can cause the actual current consumption per task to be underrepresented.

A contrived example would be an attempt to create a program with a 0% cache hit rate. This can be done by spreading jumps throughout the program such as shown in test case five. Each jump causes a cache miss, but the stall following each jump prevents the FRAM from being accessed every clock cycle. In effect, the stalls will cause the instruction throughput of the program to be halved, because only 50% of the cycles are spent performing actual work. When this effect is taken into consideration the current consumption of the 100% miss case falls into line.

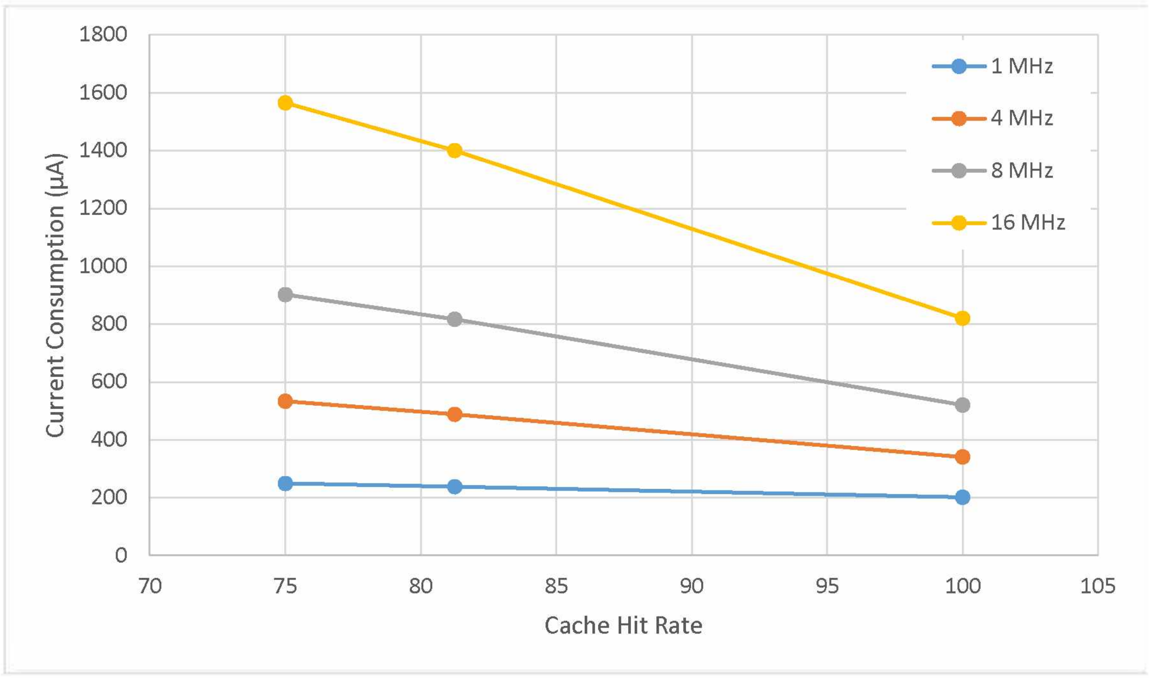

Figure 11: Raw Measurement of Operating Current vs Cache Hit Rate

When constructing a loop these stalls are unavoidable, they will happen whether the jump is taken or not [18]. This lowers the apparent current consumption from cache misses as often a cache miss follows a jump. The addition of a stall increases the number of cycles between cache misses, which accounts for the perceived reduction in energy consumption. Since the stalls are unavoidable and not a result of the cache miss itself, they can be neglected for the purpose of comparison.

A secondary check on the measurements can be made by subtracting the 100% cache hit rate measurement from each of the other test cases and then normalizing the measurements by the operating frequency. This yields the average charge per cycle spent on FRAM accesses for each case, which should be constant for each frequency. Because these values are being normalized against a 100% cache hit rate operating current, test case three is not present in the table – it provides the base operating current to normalize against.

Table 3: Average Charge Per Cycle Expended on FRAM Reads

Core Speed (MHz)	TC 1 75% (pC)	TC 2 75% (pC)	TC 4 82.5% (pC)	TC 5 0% (pC)
1	46	48	37	87
4	46	48	37	86
8	46	48	37	85
16	45	47	36	86

The frequency only affects the rate at which the cache misses occur, not the energy expended per each. The relative charge expenditure for each cache hit rate follows the expected trend as well, with the charge expenditure increasing for lower hit rates. Note that the charge consumption for the 0% case appears to be lower than one would expect. Approximately four times the charge expenditure of the 75% hit rate would be expected, because there are four times as many cache misses. The discrepancy is due to the stall cycles between jumps. Only 50% of the cycles are spent executing a jump instruction, and consequently, the corresponding cache miss is occurring in 50% of cycles.

Energy Consumption Test Conclusions

The results from these tests provide insight into best software design practices when using FRAM in the context of TI's implementation. Because the FRAM is used for both persistent data storage as well as storing the program uploaded to the microcontroller, the behavior of this cache has an outsized impact on the overall performance of the device.

The five test cases analyzed are indicators of expected performance in portions of a real program. Test case one is representative of the linear portion of a program. Any realistic program will have sections of procedural code where a set of instructions are executed in order without repetition. Setup or shutdown portions of code are good examples of this – a sequence of actions is undertaken to put the system into a defined state; such a sequence seldom requires loops.

Test case two is useful in that it proves out that LRU replacement is utilized in the caching logic. By structuring the loop such that there are 12 instructions that will map to one of the cache sets and always calling them in the same order, the test case represents a scenario similar to the completely linear program. Only 8 of the 12 instructions for the overloaded cache set can be cached at one time. If any method other than LRU is used it is expected that occasionally a pass of the loop will not cause misses because of the resultant cache conflicts. Because the operating current is nearly identical between test case two and test case one, it is safe to infer that LRU replacement is used.

Test cases three and four considered together illustrate the effect of cache mapping on the controller's energy performance. Both use the same sized loop in terms of instructions, saturating the cache with 16 instructions. The only difference between the two programs is how they are aligned in the program memory. By intentionally skewing test case four's position in memory to misalign it with the loop cache misses are induced. This is a profound result – simply considering where a block of code is located within the instruction memory can have an impact on energy consumption. The effect will occur regardless of the intent or functionality of the code in question, which is not an obvious phenomenon.

Finally. the validity of the tests is examined by computing the energy expended on FRAM reads for each test case. By using the expected cache hit rate to adjust the operating values and normalizing the frequency, an energy value for each FRAM read can be determined. This value should be the same across operating frequencies and cache utilizations, which is indeed the case here. This provides validation that the assumptions made are correct.

Silver Biocide Feedback Controller Case Study

Introduction and Background

Control of microbial growth in long duration manned space flight water systems is a complex and multifaceted challenge. A successful microbial control system must be mass efficient, reliable, and safe for long-term human consumption. Ionic silver is one such suitable anti-microbial [25], and is particularly attractive, as it need not be removed prior to consumption. This is a benefit over the current state of the art U.S. systems, which utilize Iodine as the antimicrobial and remove it prior to consumption [26].

Ionic silver in water will be lost onto any metal surfaces the water is in contact with [27] [28]. This is problematic for use in spaceflight as modern spacecraft water systems are constructed primarily of metallic materials. Existing silver biocide dosing systems, such as the current Russian silver biocide system, have been based on an open loop control system. The current strategy requires that dosing rates are calibrated based on a flow rate known a priori and water is only stored for a short duration. While suitable for the short missions these systems were intended for, this “dose and hold” strategy is not practical for long durations.

There are two primary difficulties with using one of these fluid mixing strategies for long term missions. The first is achieving a sufficiently high concentration in the dosing fluid to make storage feasible. Typically, for a silver system, Silver (I) Flouride, CAS No. 7775-41-9, is dissolved into water and used for mixing into untreated water. A solubility of 179.1g/100ml [29] is expected at 25° C, and approximately 85% of this will be free silver when dissolved. Ultimately, this means that an additional 65 g of water would need to be carried for every 100 g of silver to be dosed. In addition to the extra mass, this dosing fluid will be subject to the same deposition issues as the drinking water.

One possible solution is to employ a closed loop control system, utilizing measurements taken with an Ion-Selective Electrode (ISE) and proportional control logic implemented on a microcontroller. Electrolytic dosing of ionic silver requires very little power [30], making a low power, embedded system control scheme an ideal pairing. This type of simple low power system can be packaged very compactly, making it suitable for manned space exploration. A prototype control system was developed to verify the control scheme, demonstrating concentration control within 10% of the 300 ppb set point for two months.

Electrolytic Silver Dosing

Compared to alternatives requiring mixing of silver salts, electrolytic process have the advantage of being entirely electrically controlled. A dosing electrode can be constructed which will introduce silver into water at a rate proportional to the influx of current [31]. It is straightforward to design a device that will control a silver dosing electrode. Either variable voltage or Pulse Width Modulation (PWM) can be used to change the rate of dosing. This property of electrolytic processes make them particularly attractive compared to a mechanical system when implementing a control system.

The simplest way to introduce silver into water electrolytically is to suspend two plates of silver and induce a voltage across them [32]. The two silver plates form a pair of electrodes, and silver ions will flow from one plate to the other to complete the electrochemical circuit. Equation 1 shows the half reaction at the anode.

$$Ag_{(s)} \rightarrow Ag^{+}_{(aq)} + e^{-} \qquad (1)$$

The ionic silver is released from the anode into the water, and the electron departs into the wire towards the cathode. Chemical half reactions never occur in isolation, so at the cathode a reaction must occur where an electron is received instead of donated. The element being oxidized will depend on the impurities present in the water, but generally, it will form a layer on the surface of the electrode. Figure 1 illustrates the process.

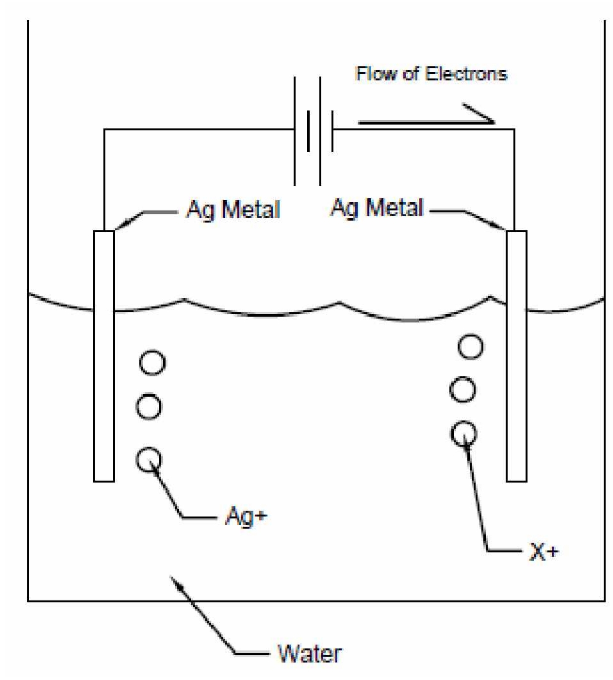

Figure 12: Anode Half Reaction

The unknown compound, shown as X on the figure, is the recipient of the electrons transport from the plate on the left. The nature of this reaction is uncertain, but it is speculated that the end result is the formation of oxides on the cathode [30]. To prevent excessive buildup on one of the silver surfaces the polarity of the plates is periodically reversed during operation. Alternating the polarity prevents one of the plates from constantly being anodic and thus depleted more quickly than the other, as well as preventing the oxide layer from forming on only one of the two plates.

The dosing electrode can be modeled electrically by a parallel RC circuit. The capacitive element of the dosing electrode represents the capacitance of the plates, which are suspended in close proximity in water. The resistive element represents the flow of current through the water. Current through flowing through the water is responsible for the actual silver generation.

The rate at which the polarity is reversed needs to be kept low enough that the majority of the current flows through the resistive element in the model. Excessive switching speed will cause most of the current to flow through the capacitive portion of the model [33], representing a situation where the plates are constantly being charged and discharged. Fast switching is not required to mitigate the oxidation of the plates, so very low switching speeds on the order of 1 Hz are selected.

Methods of Measuring Silver Content

Three ways to determine the silver content of water are considered here, inductively coupled plasma mass spectrometry (ICP-MS), Ion Selective Electrodes (ISE), or colorimetric methods. Each method has individual strengths and differing, but substantial drawbacks for long duration spaceflight use.

ICP-MS is the preferred method for taking laboratory measurements of dissolved elements. It allows for high measurement repeatability, measures total silver regardless of form, and will work regardless of the other ions present in the test solution [34]. However, it is a very impractical technology due to the volume, mass, and delicate nature of the machinery. ICP-MS also requires a steady supply of argon gas to operate, which would necessitate another consumable item.

An ISE measures concentration by inducing a galvanic action through a membrane [35]. The membrane is specially designed so that is permeable to the target ion, providing the selectivity aspect of the ISE. As ions diffuse through this membrane a potential develops, which can be related to the ionic activity of the selected ion through Equation 2, a simplified version of the Nernst equation.

$$E = E_0 + \frac{2.303RT}{nF}\log(A) \tag{2}$$

In (2), E is the junction potential between the membrane and test solution, E_0 is the sum of potentials across the extraneous junctions in the ISE, 2.303 is the conversion from natural log to common log, n is the charge number of the ion, F is the Faraday constant, and A is the ionic activity.

One advantage to using an ISE for measurement is that it eliminates the need for any additional consumables, because an ISE is a self-contained probe. The ISE outputs a voltage that is proportional to the concentration, which is simple to measure and analyze. Colloidal silver, which has no ionic activity, will not be detected by the ISE. This can be a significant drawback if high levels of colloidal silver are expected.

Colorimetric methods involve adding a reagent to a sample of water, which causes a color change to occur. The light absorption of the sample can be measured to determine the concentration; more highly concentrated samples will cause higher levels of light absorption [36]. At very low concentrations, long path lengths are required to achieve significant enough absorption to be measurable; this can be mitigated by using a technology such as liquid core waveguides [37].

The primary drawback with colorimetric methods is the unavoidable use of reagents to perform the measurements. A sample of water must be drawn off and mixed with the reagent for the measurement. Generally, these reagents are toxic and the sample water cannot be reintroduced into the water system, which introduces another consumable item as well as consuming otherwise potable water.

Based on the merits of the individual measurement technologies an ISE is used for the test run. An ISE is simple to integrate into a microcontroller based system, it outputs a voltage proportional to the silver concentration and requires no real interaction on the part of the microcontroller to operate.

Verification of the Dosing Concept

Prior to any in depth work into the refinement of the design of the device itself, functional testing of the general theory and mode of operation is performed using a feedback controller designed specifically for this purpose. The control system is used with an ISE and dosing electrode and attached to a mock water bus. A water bus is a similar concept to an electrical bus. It usually takes the form of a series of piping which distributes water within a vehicle. Water will be circulated through the system, actual silver is dosed into the water, and samples are taken for ICP-MS analysis daily. The test will be run for approximately two months to evaluate the system for drift in the setpoint.

Verification of the concept was performed prior to the design of the low power controller. An MSP430F5529 Launchpad was used to implement a rapid hardware solution. After proving out the concept development work on the low power controller was started. The low power controller was designed to be operationally equivalent to the concept hardware but introduce incremental improvements.

Water Bus

To facilitate functional testing a mock water bus was designed to test the control system against. Metallic materials were used in the construction of the water bus to approximate the construction of a practical spacecraft water bus [38]. Additionally, using metal rather than a plastic system and keeping the water in contact with a metallic surface provides loading for the control system. A schematic of the water bus is shown below in Figure 13.

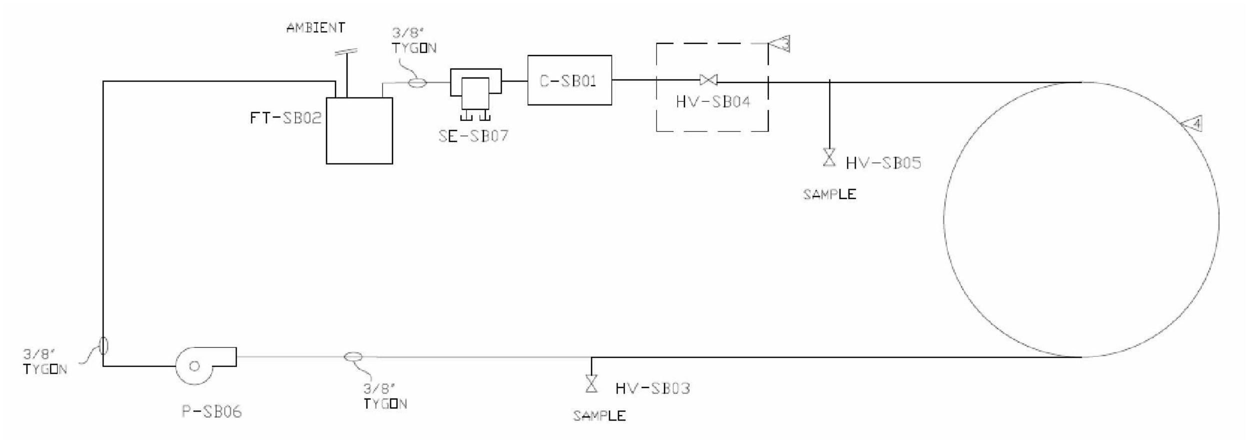

Figure 13: Water Bus Diagram

The general construction is a pump circulating water from a holding tank through a coil of stainless steel tubing, ISE housing, and dosing electrode. MasterFlex Tygon tubing is used to make flexible connections between certain features of the water bus, easing the assembly. A picture of the constructed water bus is shown in Figure 14. A housing for the ISE and connections for the dosing electrode are included in the mock water bus.

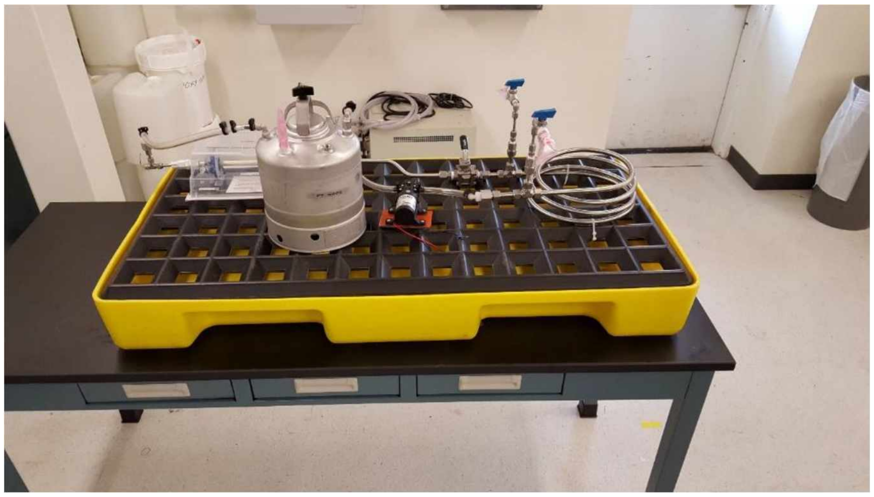
Figure 14: Constructed Mock Waterbus

Feedback Controller

For the functional testing, a feedback controller is implemented on an MSP430F5529 launchpad with a custom designed expansion board. The expansion board contains circuitry for driving the dosing electrode, connectors for the dosing electrode and ISE, and a trim potentiometer for adjusting the dosing electrode drive voltage.

The feedback controller is designed to run off of either external battery power or USB. For this testing the USB power was used to supply the board. The power regulation circuitry is shown in Figure 15.

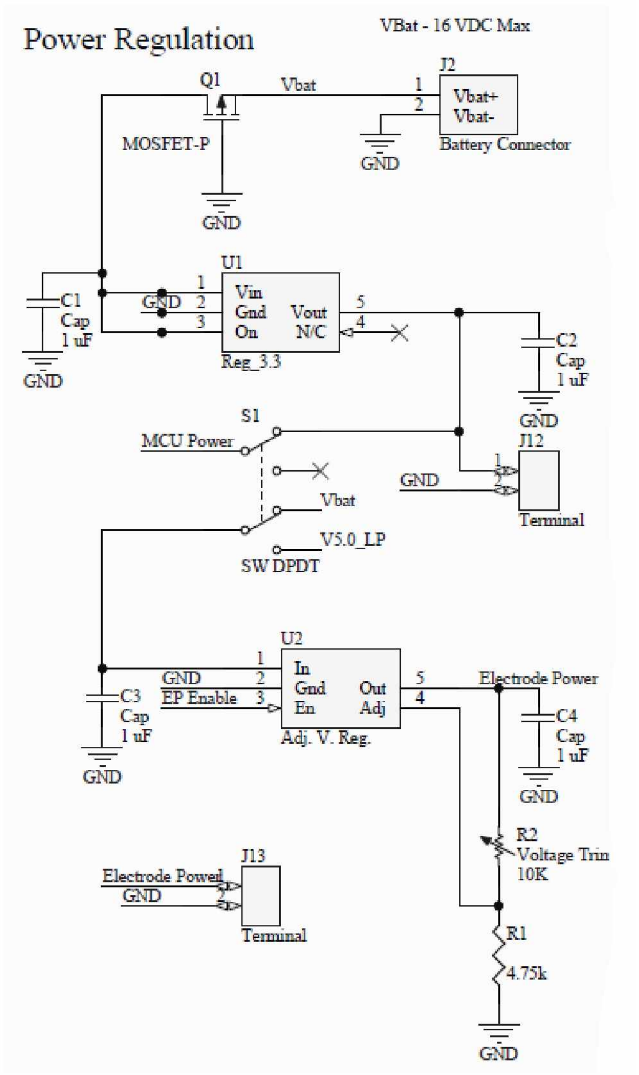

Figure 15: Feedback Controller Power Regulation Circuit

The power supply to the board can be changed from USB to battery via switch S1. When the expansion board is configured to use battery power it provides a 3.3 V output back to the Launchpad to power the microcontroller (MCU). Whether the power is supplied by a battery or through USB the general operation will be the same. The expansion board has an adjustable regulator that generates the supply voltage, which will be used to drive the dosing electrode. Dosing electrodes are generally run using an alternating PWM signal, where the duty cycle is varied to control the amount of dosing and the polarity of the PWM is reversed every other period. The Voltage Trim potentiometer, R2, can be used to change the output voltage of the adjustable voltage regulator U2. This output voltage is provided to a motor H-bridge, which is connected to the dosing electrode as shown in Figure 16.

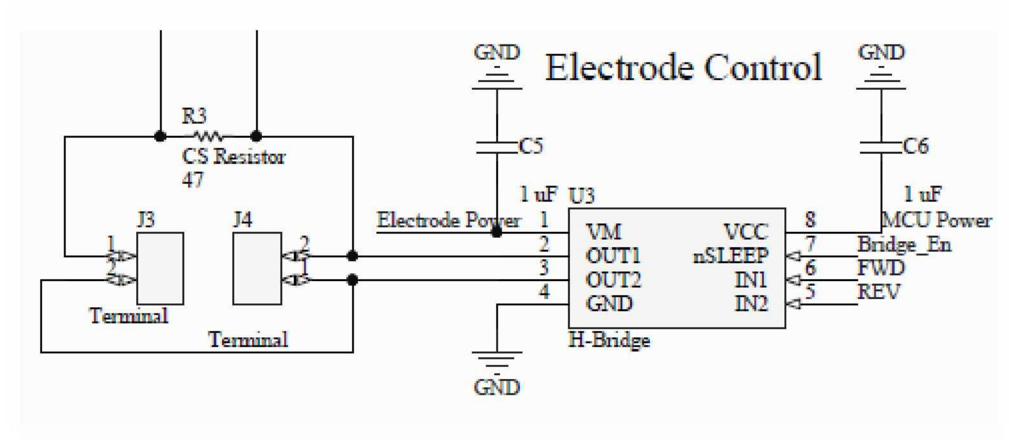

Figure 16: Feedback Controller H-Bridge Circuit

The H-bridge, U3, is used to drive the dosing electrode from the adjustable voltage provided from the regulation circuit. The H-bridge provides an easy way to reverse the polarity of the dosing electrode, or to put it into a high-Z state. Terminal J4 is an open testpoint, which can be used to observe the output voltage from the H-bridge, and terminal J3 connects to the dosing electrode. All of the above circuitry is implemented on an expansion board, which mates with the Launchpad. A picture of the expansion board is shown in Figure 17.

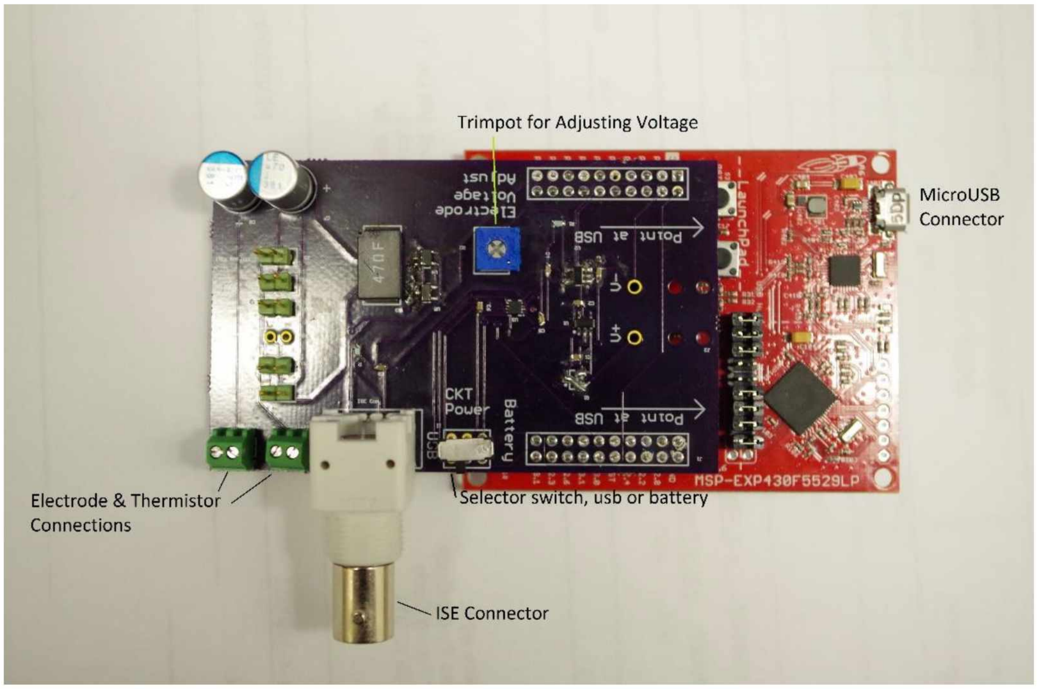
Trimpot for Adjusting Voltage
MicroUSB Connector
Electrode & Thermistor Connections
Selector switch, usb or battery
ISE Connector

Figure 17: Photograph of Electrode Controller

The backchannel UART provided by the MSP430 is used to offload data from the controller back to a data collection computer. A LabView VI is used to capture the data from the controller, but any software capable of capturing data from a COM port could be used. Numeric values are in binary, rather than ASCII form.

Included on the feedback controller is the ability to connect to a thermistor for measuring the temperature of the process water. This functionality was never utilized or incorporated into the software. Temperature has a minor effect on the ionic strength of the solution, so it should be possible to increase the accuracy of the control system by compensating for temperature. Adequate control was achieved without compensating for temperature but the option is available in the hardware.

The feedback controller software implements a simple proportional-integral (PI) feedback loop. The Timer A module of the MSP430 is configured to run directly from an external 32768 Hz crystal in UPDOWN mode. The timer counts to a peak value specified by writing a value to the CCR0 register and then counts back down to zero. Interrupts can be attached to CCR0 and CCR1 modules, and are used to configure the output waveform. Figure 18 is provided to help explain the progression of events triggered by the timer.

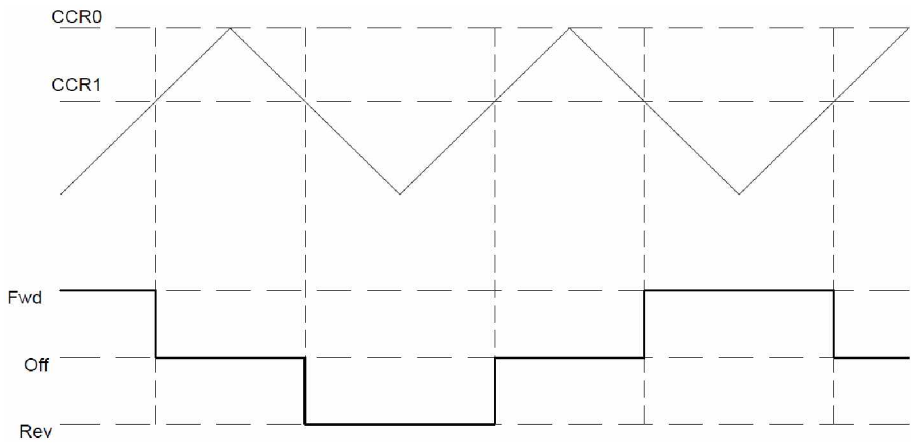

Figure 18: MSP430 Counter Pattern

When the timer count reaches the programmed value of 32768, the following set of actions is performed:

1. The ADC is activated to perform four measurements of the ISE voltage
2. A variable containing the plate direction is changed to the opposite value
3. A data string is offloaded through the backchannel UART
4. CCR1 is updated to represent the new duty cycle
5. Data is sent through UART interface, including the current proportional and integral gain values, the most recent ADC average measurement, and the error signal in the timer register.

The ADC operation is carried out in the background, and upon completion of these measurements a separate interrupt for the ADC is triggered. The interrupt starts a routine that averages the four measurements just completed, and then computes a rolling average of the previous 10 measurement ensembles. Finally, this value is subtracted from the setpoint, and the result is multiplied by a gain factor to determine the new error signal.

Sampling time for the ADC is based on the resistance of the source being measured. An ISE has an internal resistance of approximately 10 MΩ [39], and according to the MSP430 datasheet, the internal resistance of the ADC is 2 kΩ [10]. The ADC can essentially be considered as an RC circuit where the resistances are in series with the ADC capacitance, which is given as 2.5 pF [10]. A time constant can be determined from the RC circuit, which in this case works out to $25 * 10^{-6}$ s, or 25 μs. For a 10 bit ADC, a sampling time of eight time constants is required for 1 least significant bit of precision, so a sample time of 200 μs is used.

This value is stored in a global variable, which will be used to update CCR1 the next time the timer hits the CCR0 value. The end result is a PWM with alternating direction, an example output of which is shown below in Figure 19.

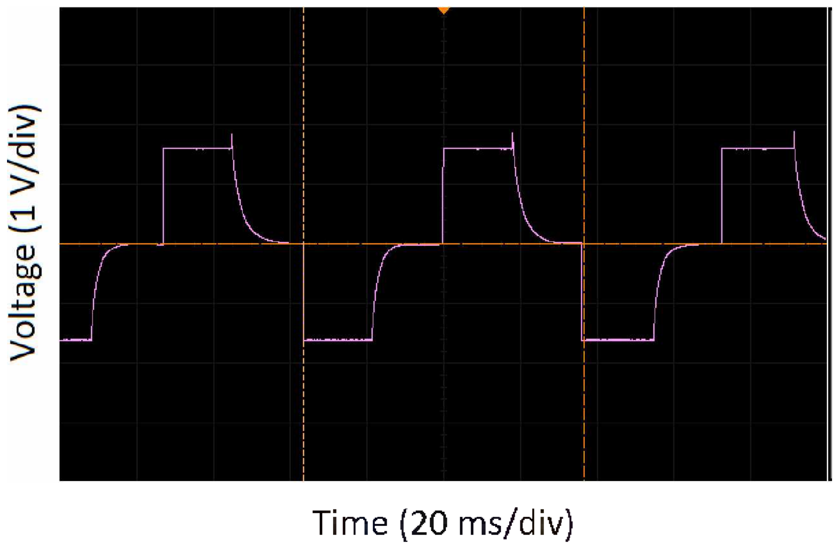

Figure 19: Feedback Controller Output to Dosing Electrode

Dosing Electrode

The dosing electrode used is provided by a Small Business Innovation Research grant, and consists of a silver rod suspended within a silver tube with an abrasive scrubber positioned to sweep between the two. A picture of the dosing electrode is shown in Figure 20.

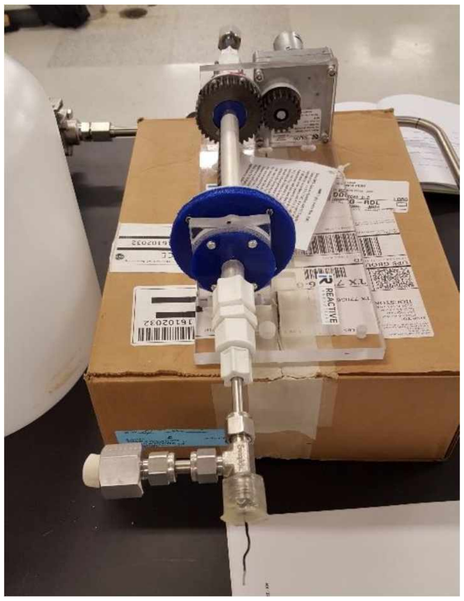

Figure 20: SBIR Dosing Electrode

The silver tube in the image is positioned between the blue idler wheel and gear. The silver rod, which forms the opposing surface of the electrode, is suspended in the center of the tube. Between the two silver surfaces there is a magnetic wire treated with an abrasive coating. This wire is pulled along the inner surfaces by a series of permanent magnets fixed onto an arm. The arm is rotated along the outside of the silver tube and via magnetic action pulls the scrubbing wire along the two silver inner surfaces.

Verification Test

Test Procedure

Functionality of the device is confirmed by connecting it to the mock water bus and running the system for a seven week period. Prior to attachment to the water bus, a setpoint for the controller is determined by attaching the ISE and immersing it into an AgF solution with a known silver content. The measurement recorded by the feedback controller is recorded and used as the setpoint.

With the setpoint determined, the feedback controller is connected to the ISE in the mock water bus, and the SBIR dosing electrode is connected to the output. The system is filled with deionized water and the pump and feedback controller are energized. During the operating period samples are taken each workday and measured via ICP-MS. Water is added to the system each time samples are taken to replenish the water used for taking the measurement as well as water lost to evaporation.

Results

Figure 21 shows the values recorded by the LabView VI with the measurements taken by ICP-MS for comparison. The blue line is the concentration of the system as measured by the controller, and the orange triangles are the ICP-MS measurements. The system takes approximately a day to reach equilibrium according to the controller and the ISE measured concentration is stable afterwards.

To evaluate the performance of the controller the standard deviation of the ICP-MS measurements after stabilization is computed. The standard deviation exhibited by the system is 20 ppb. Using two standard deviations as the upper and lower thresholds yields a control range of 300 ± 40 ppb. An acceptable range of concentrations could fall between 100 and 400 ppb, based on the lower limit on efficacy of silver biocide and the upper limit of safe exposure to humans.

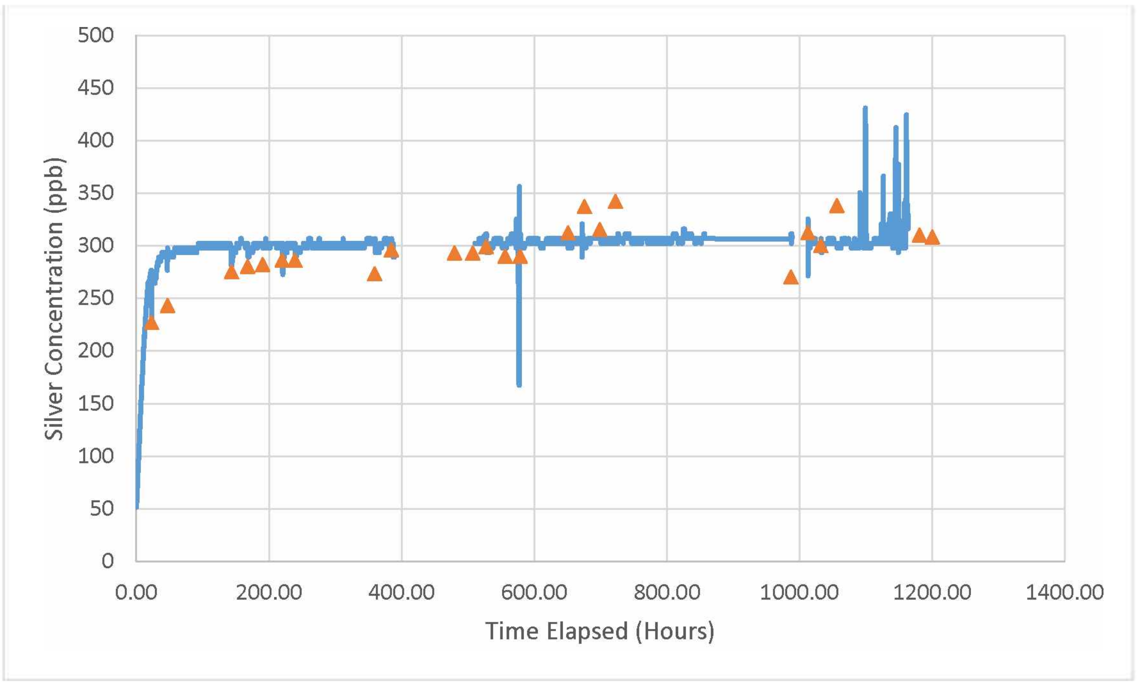

Figure 21: System Concentration vs Time

Low Power Redesign

The technologies and methods used to implement the feedback controller make them an excellent candidate for a low power embedded system. In a practical application there are likely to be long quiescent periods where no silver dosing is required. Despite this, slow but gradual losses will ensure that periodic checks still need to be made.

Theoretical current consumption for a dosing electrode is very low, dosing one liter of water from 0 to 400 ppb of silver would require 0.36 C of current. It follows that a treatment rate of one liter an hour would require approximately 100 µA of continuous current. A practical application for the device is likely to also include long quiescent periods where no dosing occurs. Because of these two facts, it is beneficial to keep the current overhead of the control system as low as possible.

The performance of the low power controller described in the Low Power Controller section on page 9 was considered in this application. A control program similar to the one used in the Launchpad based controller was created for the low power controller and current values are measured.

It is difficult to obtain accurate current measurements due to the variation in current between active and low power modes. While sleeping, the controller might consume as little as 70 nA, and in active mode current consumption on the order of 200 µA can be expected. Active mode durations are very short which makes accurately capturing the current consumption difficult.

To make an accurate measurement of the time averaged current used by the system two methods are used. The first method involves breaking the operation of the controller into a series of segments based on their likely current consumption. These segments can be measured for duration and power consumption on an individual basis and then time averaged to determine the overall average current.

The second method is to measure current directly with an integrating ADC type instrument. By setting the measurement aperture of the instrument to a comparatively long duration compared to the controller active period. A voltage reference and filter capacitor are used to keep the input voltage to the device under test (DUT) stable and smooth out current spikes.

Piecewise Current Measurement

There are five distinct regions that should be considered when constructing a piecewise measurement of the current profile:

- Start-up: The time required to wake from the external general purpose input/output (GPIO) signal and prepare the controller to begin taking the first measurement.
- ADC Sample Time: The time the controller is asleep, but with the ADC active taking the measurement.
- ADC Sample processing time: Processing time that occurs between each ADC sample, where the ADC result is added to a cumulative value for averaging. This, and the previous step, occur once for each ADC sample being taken.
- Processing Time: The time required to compute the average ADC value, set the state of the dosing electrode, and prepare to sleep again.
- Sleep: Time the controller is in LPM4.5 between wake events.

A current profile needs to be constructed for the controller. The formulation of that profile can be simplified by separating the determination of time and current consumption into two separate processes.

Times can be measured by setting the controller to set a pin in software and recording with the oscilloscope. For the Start-up segment, the Wake pin of the TPL5010 is considered alongside a GPIO to capture the entire wake from LPM4.5 process. This is necessary as it is not possible to set a GPIO during the wake process, where the MCU is in a transient state and powering up. The sleep segment also requires special consideration, as it is too long to practically measure with an oscilloscope. Each other segment can be measured by pulling a pin high at the beginning of it, and then pulling it low again at the end.

There are three operating modes associated with the stages, active mode, LPM0 with the ADC on, and LPM4.5. A current measurement for each operating mode is taken by modifying the controller code to "lock" it into that operating mode. For active mode a GOTO instruction is used to force the controller to repeat a section of code without entering a low power mode to take measurements or go to sleep. For LPM3 with the ADC the controller is configured to sample and convert indefinitely without firing an interrupt to read out the value.

The LPM4.5 operating mode is long enough in duration that no modifications to the code are required to measure it. The measurement technique detailed in the Shutdown Current subsection on page 11 is used to determine the LPM4.5 current.

Results

Time and current measurements for each stage are given in Table 4 and Table 5 respectively.

Table 4: Time Measurements

Stage	Mode	Duration (µs)
Startup	Active	554
Sample and Convert	LPM0 + ADC	306
Process Samples	Active	22
Sleep Prep	Active	30
Shutdown	LPM4.5	~ 60 s

Table 5: Current Measurements

Mode	Current (µA)
Active	331
LPM0 + ADC	356
LPM4.5	.072

The samples processing step occurs between each ADC sampling when the newly measured value is added to a running total. These piecewise measurements can be combined to form the current profile shown below in Figure 22. The figure excludes the relatively long shutdown time for clarity. Taking the time average of the currents yields an overall average current of 83 nA.

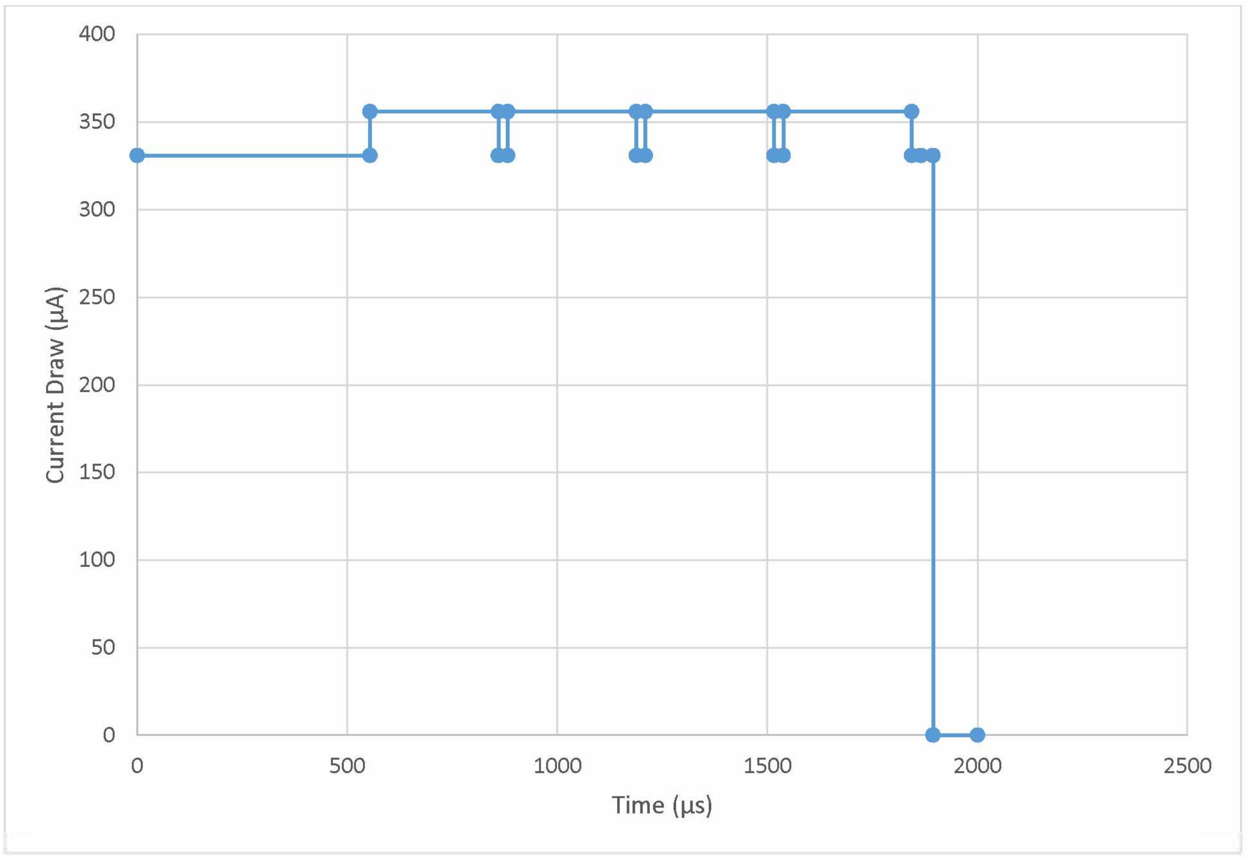

Figure 22: Current Profile for Low Power Feedback Controller

Direct Measurement

The average current of the system can be measured directly by taking advantage of the nature of the integrating ADC used in most commercial digital multimeters. As explained in the Active Mode Current section on page 12, an integrating ADC will time average the test current during the measurement aperture window. Even with the long integration time, it is necessary to ensure that the test current does not exceed the maximum for the range setting on the multimeter. If the current range is exceeded even momentarily, the meter will attempt to switch ranges or yield an over limit reading.

A second issue with measuring the current in this manner is voltage regulation in the test circuit. If the circuit is not carefully designed the input voltage to the DUT will drop as the DUT draws more current. Changes in the input voltage will likewise cause the measured current of the device to change. To facilitate testing a support circuit is designed with the following goals in mind:

- Prevent current spikes during the active period from overloading the input stage of the multimeter.
- Keep the input voltage as close to constant as practical.
- Minimize unpredictable parasitic current during testing.

A bandgap reference in conjunction with a set of filter capacitors is used to implement a support circuit for the DUT. A linear LT1460-2.5 bandgap reference is used to provide a stable input voltage to the DUT. This device was chosen for its very low load regulation at the expected operating currents and high accuracy. At 100 µA, less than 1 mV of load regulation is expected, and the output voltage is expected to be within 0.075% of the specified value.

The output of the voltage reference is fed into a Keysight 34461A instrument configured to measure current. The output of the 34461A is decoupled to ground by a 2.2 µF electrolytic capacitor and fed into the DUT as the supply. An additional 34461A is connected in series with the electrolytic capacitor for voltage monitoring during initial checking but is removed for testing. Figure 23 shows the layout used for measuring the current consumption of the device.

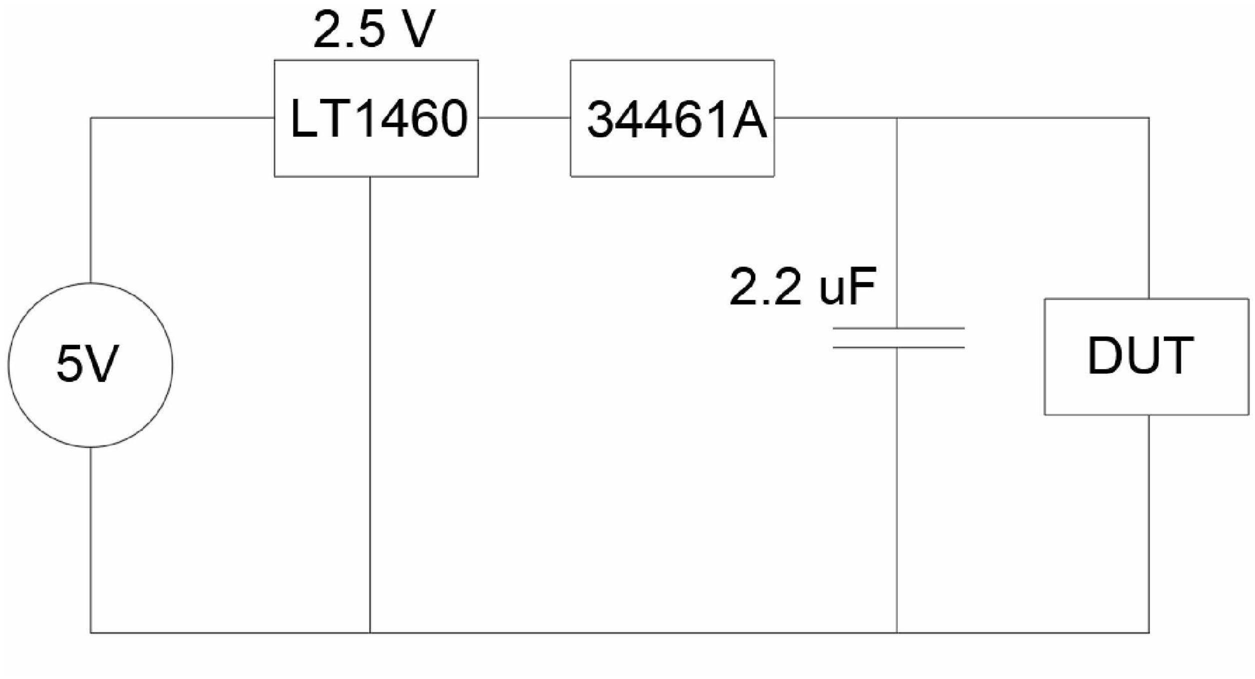

Figure 23: Direct Measurement Schematic

Prior to using the circuit for testing, second 34461A is placed to measure voltage in parallel with the decoupling capacitor. This is done primarily to ensure that the LT1460 is working appropriately and providing the correct output voltage.

After verifying the functionality of the circuit, a current measurement is taken with the DUT detached. This allows for calibrating the current flowing through the capacitor and parallel voltage digital multi-meter (DMM). After the current DMM is set to null the stray currents and the DUT is reattached, the current DMM measurements will now reflect the current flowing through the DUT alone.

Both the current and voltage DMMs are set to use a 100 power line cycle integration period which corresponds to 1.66 seconds on 60 Hz mains frequency. Both meters are set to capture statistics and the test rig is operated for 15 minutes. At the end of the test period the average output voltage of the LT1460 and current consumed by the DUT are recorded.

After achieving a satisfactory result with the test run, the voltage DMM is disconnected. The null on the current DMM is reset to reflect the reduced load current with the voltage DMM removed, and the test rig is again operated for 15 minutes. At the end of the period, the average value captured by the current DMM is recorded.

Before using the circuit with the controller, a test resistor is used as a DUT in the final measurement configuration. The resistance of the test resistor is measured using the DMM prior to performing the test. This measured value can be used in conjunction with the voltage assumed from the bandgap reference to estimate the cumulative error from imprecision in the voltage reference, parasitic currents from the breadboard, and error caused by zeroing out the capacitor leakage current.

Table 6: Direct Measurement Test Current Values

Test Scenario	Measured Current
No DUT	98 pA
98.84833 kΩ Resistor	25.3582 µA
Parallel DMM w/ no DUT	248.59 nA
Controller with parallel DMM	326.59 nA
Controller without parallel DMM	78.31 nA

During the test runs with the parallel DMM the output voltage of the LT1460 is monitored for consistency. Throughout all the runs, it yielded only a 30 µV standard deviation, suggesting that the load regulation of the voltage reference is very low.

Measuring the current through the test resistor allows for an estimate of the error resulting from the apparatus. Given 25.3582 µA through a 98.84833 kΩ the voltage of the bandgap reference would be 2.5066 V. These values suggest an experimental error of 0.264%.

A current consumption of 78.31 nA is measured via this method, as opposed to the 83 nA computed using the piecewise method. It is expected that the piecewise estimate will be high due to the way the individual regions are derived. For each region in the piecewise estimate, a measurement is taken which will represent the high mark for that time period. An approximation of the current consumption during that region is made by simply multiplying this current by a duration. In actuality, the change in current is continuous and will ramp up rather than immediately jump. This is particularly pronounced during the wakeup of the controller. These errors are not present in the direct measurements, thus resulting in a lower measured current.

Comparison to Other Technologies and Techniques

The most straightforward comparison that can be made is to compare the FRAM based system using shutdown to a similar MSP430 which is not FRAM equipped. There are two approaches that could be taken in such a case: only use a low power mode, which allows for RAM retention, or write the necessary values to flash. To make a comparison, the MSP430FR2311 will be used for both cases, because it can use a ram retention mode instead of shutdown. This allows for an even footing between both approaches since active current values will be identical. To provide for the flash approach the requirements from the similar MSP430F23x0 series will be used alongside the specs from the MSP430FR2311. Energy consumption by the flash writes will be taken from the F23x0 datasheet and all other energy consumption values will be taken from the FR2311 data.

If a strategy of simply keeping the controller in retention mode is used, there are two major changes to the current profile. The sleep current will increase to 450 nA instead of the 25 nA for shutdown. The startup time, however, will decrease to 10 µs.

For a scenario where flash is used to store values, the shutdown current and wakeup time from the FRAM case will be used. An additional step in the profile will be added for the flash programming time, using values from the MSP430F23x0 family datasheet. Typical current for a flash program or erase is 1 mA. The controller can make single byte or word programs of the flash memory in 120 us. Before a program operation the entire segment must be erased, which takes approximately 14.5 ms. In this application there is only one byte which needs to be stored, the last drive direction of the plates which simplifies the use of flash. If there are other values stored in the same flash segment they will also need to be reprogrammed, incurring an additional 54 us delay per word. The current profile for the three devices is summarized in Table 7.

Table 7: Case Current Profiles

Stage	FRAM	Ram Retention	Flash
Startup	554 µs at 331 µA	10 µs at 331 µA	554 µs at 331 µA
ADC	1312 µs at 356 µA	1312 µs at 356 µA	1312 µs at 356 µA
Sleep Prep	30 µs at 331 µA	30 µs at 331 µA	30 µs at 331 µA
Flash Program	-	-	14.6 ms at 1 mA
Shutdown	72 nA	497 nA	72 nA

The current profiles were used to compute average current for a span of different sampling periods.

Table 8 summarizes the average currents at several selected periods, and Figure 24 shows a plot of the current consumption values.

Table 8: Summary of Case Average Currents

	Average Current (µA)		
Sampling Period	FRAM	Ram Retention	Flash
0.1 s	6.7	5.3	152.8
1	0.73	0.97	15.4
10	0.14	0.55	1.6
60	0.083	0.51	0.33
600	0.073	0.497	0.097
3600	0.072	0.497	0.076

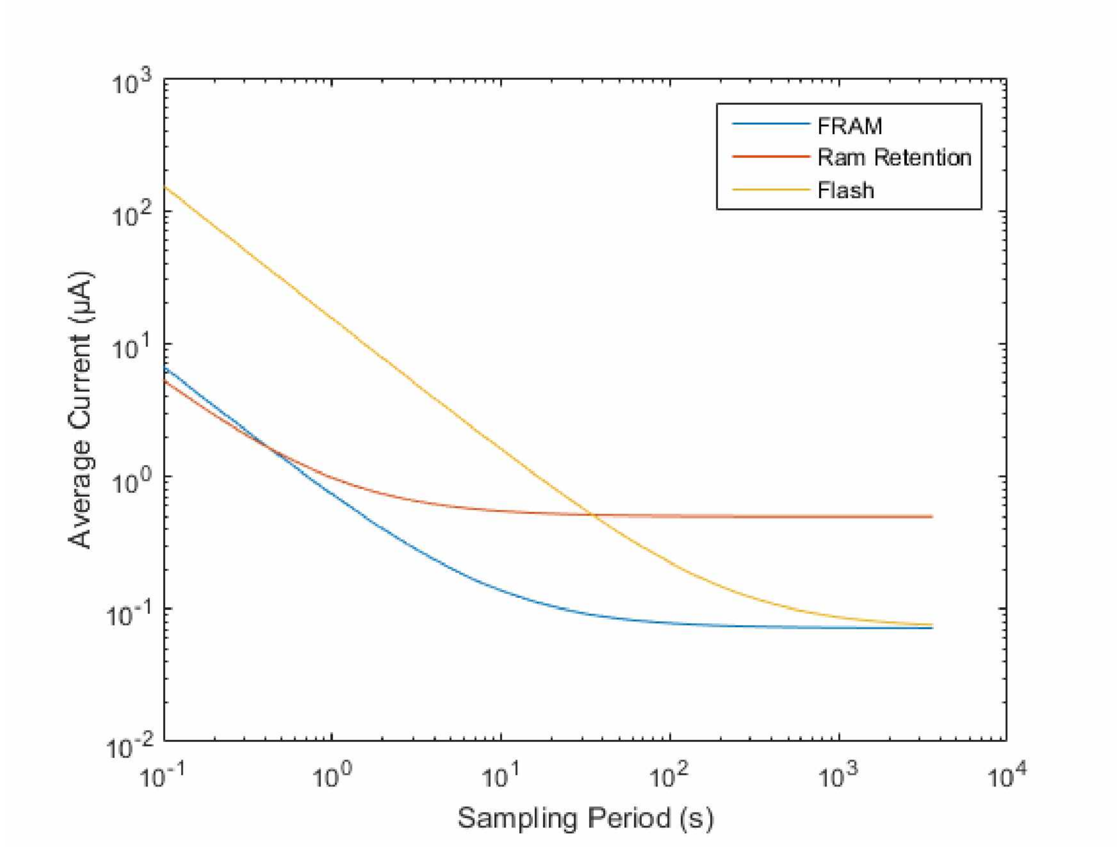

Figure 24: Case Average Current vs Sampling Period

As expected, the average current trends toward the sleep current value as the sampling period increases in length. If the sampling period is sufficiently long, on the order of 1000 s, the energy benefits of using FRAM begin to disappear. It is important to recall that flash has a limited cycle life, and that only a single byte of data is needed to be retained for this controller application. As more data needs to be persisted, the sampling period will need to be longer to make flash competitive with FRAM. Additionally, in a worst case scenario where the flash only survives for 10,000 cycles, even hourly polling will result in a lifetime of 416 days. At the 72 nA current draw a 38 mAh lithium coin cell battery would last for 56 years assuming no self-discharge. For most practical power sources the flash cycle life will be the limiting endurance factor.

For very low sampling periods, RAM retention becomes the most favorable configuration. This is driven largely by the long wakeup time required to bring the controller out of shutdown, rather than the FRAM itself. Provided the sampling period is roughly 0.5 s or longer using FRAM will provide the lowest average current.

Comparison to NOR Flash Memory

The low power required to perform FRAM writes suggests that it might be an attractive choice for data logging applications. In reality, FRAM provides less benefit when it is used for a static solution such as this. The true benefit of FRAM is that it allows for frequent, random writes to memory without a large energy penalty [40]. The tradeoff between FRAM and flash in this sort of application is defined by the nature of the data being written to long term storage.

FRAM outperforms NOR flash on a byte per byte basis during writes, but this advantage is minor. The major contributor to FRAM's energy performance is that multiple writes can be performed on the same bytes in memory without an intervening erase step. FRAM readouts are by nature destructive and the FRAM controller will need to rewrite each byte following a read, but the chemistry of FRAM does not have an analog to the NOR flash erase.

If a byte is to be rewritten in NOR flash, an erasure will have to be performed on some block of memory that contains the desired memory address. The size of this block can vary with the implementation, but is usually in excess of 100 bytes. Rewriting one byte will require that the entire block is read out and stored in volatile memory, erased, and finally rewritten in its entirety. This sequence amplifies greatly the benefits of FRAM.

When memory writes are performed only in sequence however, it is possible to eliminate the penalty from the block erase step. As long as the same byte is not written twice, erasure of the block will not be necessary. In the trivial case of a static data logger which does not need to change and store variables between sleep cycles writes to flash will be strictly sequential. The overhead of the erase step can be eliminated by simply erasing the flash memory prior to deploying the device.

Two cases are examined to illustrate the difference in energy performance based on the storage technology and the type of application. Both FRAM and NOR flash energy consumption will be compared in a simple data logger application and a controller situation where memory needs to be overwritten. To focus the comparison on the memory technologies, the energy consumption of the controller itself can be neglected. It is assumed that best practices will be employed to limit the current consumed by the controller during the memory accesses and that any remaining effect will be negligible.

For the following analysis, a Macronix MX25R4035F 4 Mb Flash memory IC will be used for the flash based solution and a Cypress Semiconductor CY15B104Q 4 Mb FRAM will represent the FRAM solution. Both IC's utilize a similar serial peripheral interface (SPI) scheme to transmit data to the device. An opcode and address are sent first and then a series of bytes to store follow [41] [42].

Case 1 – Datalogger

A datalogger doesn't require any random memory access so the flash module will not require any erasing. It is trivial to compute the energy consumed by the module by simply determining how long each byte will take to write; no erase steps are needed. The relevant characteristics of the flash module are shown in Table 9 and the FRAM module in Table 10. For the datalogger case, the controller will make 32 byte writes to the memory, in sequence, and never read them back or change them. Vcc will be held at a constant 2.5 V, a middle value for both memory modules.

Table 9: MX25R4035F Performance Characteristics

Page size	256 Bytes
Sector size	4 KB
Byte Read Time (ULP Mode)	192 ns
Byte program time	40 µs
Sector erase time	58 ms
Read current	2.2 mA
Program current	3.5 mA
Erase current	3.1 mA

Table 10: CY15B104Q Performance Characteristics

Byte read time	200 ns
Byte write time	200 ns
Active mode current	1.4 mA

To write the data, the flash module will spend 1.28 ms programming the bytes, drawing 3.5 mA throughout. At 2.5 V supply voltage this will result in 11.2 µJ of energy consumed. For the FRAM to perform the same task it will require 22.4 nJ of energy. FRAM provides orders of magnitude benefit in energy consumption, but is significantly more expensive. In 2019, the MX25R4035F is available for $0.39 for a single unit, while the CY15B104Q costs $26.91.

It is also important to note that while the relative difference in energy consumption in this scenario is large, the absolute difference is still quite small. For perspective, at 300 µA operating current, 50 ms of computation time, and 2.5 V supply voltage a controller will consume 37.5 µJ of energy. Despite the large relative difference between the currents consumed, the overall energy per measurement would only decrease from 48.7 µJ to 37.5224 µJ, a 23% improvement. In many cases, simply increasing the energy storage of the device would be a more economical avenue.

Case 2 – Feedback Controller

Here a scenario where a value needs to be read back and rewritten is considered. For every measurement, 32 bytes need to be read from memory, altered, and written back.

The flash module will require 6.144 µs of read time at 2.2 mA to read the memory, 58 ms at 3.1 mA to erase the sector, and then the same 11.2 µJ of energy as Case 1 to write the values back to memory. The total energy consumed by the flash module in this instance is 460.7 µJ. If the FRAM module is used instead, there is no need for an erase stage and the read and write stages both require 6.4 µs at 1.4 mA current consumption, for a total energy use of 44.8 nJ.

This is a scenario where the full benefit of FRAM is demonstrated. Because of the need to erase the sector in the flash module, a large amount of additional energy is consumed without benefit. When considered alongside the same 37.5 µJ processing consumption as Case 1, the flash module now requires 498.2 µJ per measurement vs the 37.5448 µJ required for FRAM. This is a 92% improvement, as opposed to the 23% improvement from before.

Conclusions

FRAM can be a transformative technology, but it is not a clear upgrade in every case. The primary situation where FRAM can provide maximum benefit is a use case where values need to be changed frequently but the idle period of the system is sufficiently long that using a shutdown mode is justified. Although it will provide a benefit any time writes are performed, the destructive readout of FRAM will cause higher energy consumption in read-only scenarios.

The ease of use with FRAM when putting controllers into deep sleep make it an attractive choice for solutions which will be operated for short times at some regular interval, especially if there is some need to retain and change values between sleep cycles. There are many examples of this when considering applications for embedded systems. An Internet of Things device could leverage FRAM to retain a TCP state between wake periods, eliminating some overhead establishing a TCP socket every period. Some datalogging applications might benefit from the ability to change the types or nature of measurements taken depending on previous measurements. The silver biocide feedback controller outlined here could function with a very long measurement cycle and requires tracking of previous states. There are many feedback control systems that could be implemented in a similar fashion.

When to use FRAM

FRAM is the ideal choice when nonvolatile storage is required for values that will be frequently overwritten. In a case where no overwrites are required, like the data logger, the erase step of flash programming can be omitted. Without the need to perform costly erase operations, which operate on entire segments of the flash memory and are extremely slow, bytewise storage to flash is practical and effective. Although FRAM will still provide some benefit when performing sequential writes with no overwriting, the benefit is less pronounced when the application would not require flash erasure.

The silver biocide feedback controller scenario presents an overwrite case which is generally favorable to FRAM. The controller requires only a single word to be persisted through shutdown. Using flash introduces a need to reprogram an entire segment to change one value. After performing the erase command, any other values in the same segment will need to be reprogrammed, even if left unchanged. This method of operation further aggravates the flash implementation, particularly when values require updates at different intervals. It is possible to mitigate this effect in flash somewhat by carefully planning the layout of variables in the flash memory. Potentially only a single variable can be stored in each segment to avoid unwanted erasure of data. This obviously leads to extremely inefficient use of memory and still does not eliminate the penalty of the erase step.

In contrast, FRAM allows for fast and low energy changes to single values without the need to reprogram an entire segment. In this domain, the primary competition for FRAM is SRAM. When nonvolatile storage is not a requirement SRAM will allow for the fastest, lowest energy storage available. The decision to use FRAM over SRAM is based on the need for nonvolatility. In the feedback controller case, nonvolatility was needed to allow for the use of shutdown mode to save power. The long wakeup time from shutdown limits the usefulness of this technique at low sample periods. If the idle times are sufficiently short, savings are realized by using avoiding the use of shutdown due to the long wake up time.

The application considered with the MSP430FR2311 suggests that a sample period of roughly one half second is enough to justify shutting down the controller. Waking the controller from deep sleep introduces overhead. In the MSP430 architecture each wake from deep sleep is treated as a power-on event and causes execution to begin again from the main entry point. This means that all of the setup code is executed on every wakeup, so effort should be taken to minimize initialization code impact.

The use of shutdown to save current is only one justification for nonvolatile storage. It is also possible that nonvolatile storage is desired for robustness and reliability reasons. If a critical device is being designed that must be tolerant of interruptions in its power source, nonvolatile storage would be desired even when SRAM would otherwise be the best choice. On the MSP430 platform an FRAM write takes 125 ns, compared to an almost 15 ms write time for flash. Although it will introduce some wait states, the onboard FRAM is fast enough that critical variables could be placed onto it. Even in the case of an unexpected reset, this will allow for the state of the program to be maintained by simply adding some initialization code to allow recovery.

With flash memory, this type of persistence is not feasible. If a value is changed tens of times per second FRAM will provide a tremendous energy savings over flash. If energy is not a limiting factor, the amount of time required to write to flash and the additional complexity required to track when erases are necessary would severely discourage the use of flash in this type of application. Even if the energy cost and performance penalty could be tolerated, a flash implementation would certainly fail in such an application due to its low cycle life. Wear leveling could be considered but this increases hardware and/or program complexity, making an FRAM based solution a clear choice in such a situation.

When FRAM benefit is limited

Figure 24 shows that using FRAM and shutdown mode begins to outperform simply keeping values stored in RAM once the idle period exceeds 0.5 s. The use of FRAM becomes increasingly beneficial until the idle period reaches approximately 1000 s or roughly 15 minutes. At this point, the shutdown current is the dominant factor in the average current and the energy saved by using FRAM becomes irrelevant. If the wear life of flash is sufficient for the application, the additional cost of FRAM will provide little benefit to offset its additional costs. Applications with extremely infrequent writes, such as a datalogger that records a value once per day, will see almost no benefit from using FRAM for storage. In addition to the diminished energy savings, the longer the sleep period the less likely the finite cycle life of the flash memory will be exceeded.

Read heavy applications will also favor flash over FRAM. The per byte energy cost for reading FRAM is in general greater than that of flash due to the destructive readout of FRAM. Specific numbers will vary depending on the implementation of the individual flash unit and how it is used. The Macronix MX25R4035F is capable of outputting data on multiple lines to increase read speed and efficiency. The additional complexity of the instructions and decoding the data make it impractical for applications where only 32 bytes are being read but they can have a large impact for bulk reads. Simply utilizing the second I/O line while keeping the MX25R4035F in low power mode would make it perform reads almost twice as quickly as the FRAM module give it a significant lead in energy performance. If nothing is being written no erases are necessary, alleviating the most substantial performance penalty of flash memory.

Future Work

Research into stable energy sources would be a logical follow up to the conclusions presented. In a battery powered application a 70 nA average current is low enough that battery self-discharge becomes the dominant factor in battery life. If the self-discharge can be mitigated it would be possible to create devices with battery lifetimes in the order of 10's of years. Alternatively, a very low average current could make it possible to power devices by temporarily storing scavenged energy. Such devices could be useful for low frequency control loops, such as the one given in the feedback controller case study, in extremely remote locations. Space applications, where energy constraints are at a maximum could benefit greatly from this type of embedded system.

Works Cited

[1] W. Li, L. M. Calle, A. J. Hanford, I. C. Stambaugh and M. R. Callahan, "Investigation of Silver Biocide as a Disinfection Technology for Spacecraft-An Early Literature Review," in *International Conference on Environmental Systems*, Albuquerque, 2018.

[2] M. R. Callahan and M. J. Sargusingh, "Design Status of the Capillary Brine Residual in Containment Water Recovery System," in *International Conference on Environmental Systems*, Vienna, 2016.

[3] D. L. Pierson, "Microbial contamination of spacecraft," *Gravitational and space biology bulletin : publication of the American Society for Gravitational and Space Biology,* vol. 14, pp. 1-6, 2001.

[4] B. M. Slote, E. Salley, D. Carr and M. C. Kimble, "Silver Ion Biocide Delivery System for Water Disinfection," in *International Conference on Environmental Systems*, Vienna, 2016.

[5] S. Aritome, R. Shirota, G. Hemink, T. Endoh and F. Masuoka, "Reliability Issues of Flash Memory Cells," in *Proceedings of the IEEE*, 1992.

[6] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M. R. Stan and S. Swanson, "Modeling Power Consumption of NAND Flash Memories Using FlashPower," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 32, no. 7, pp. 1031-1044, 2013.

[7] R. Bailey, G. Fox, J. Eliason, M. Depner, D. Kim, E. Jabillo, J. Groat, J. Walbert, T. Moise, S. Summerfelt, K. Udayakumar, J. Rodriquiz, K. Remack, K. Boku and J. Gertas, "FRAM memory technology - advantages for low power, fast write, high endurance applications," in *International Conference on Computer Design*, San Jose, 2005.

[8] J. F. Scott and C. A. Paz De Araujo, "Ferroelectric Memories," *Science,* vol. 246, no. 4936, pp. 1400-5, 1989.

[9] Texas Instruments, "Application Report SLAA628 - MSP430 FRAM Technology," 2014.

[10] Texas Instruments, "MSP430F5438 Datasheet," 2018.

[11] Texas Instruments, "Application Report SLAA334A - MSP430 Flash Memory Characteristics," 2008.

[12] M. Jimenez, R. Palomera and I. Couvertier, ntroduction to Embedded Systems: Using Microcontrollers and the MSP430, Springer Publishing Company, 2014.

[13] A. Azizi-Mazreah, M. Manzuri, H. Barati and A. Baradi, "Delay and Energy Consumption Analysis of," *International Journal of Electronics, Circuits and Systems,* vol. 2, no. 1, 2008.

[14] J. Davies, MSP430 Microcontroller Basics, Elsevier, 2008.

[15] Micro Crystal Switzerland, "RV-1805-C3 Datasheet," 2016.

[16] Texas Instruments, "SNAS651A - TPL5010 Nano-Power system Timer," 2018.

[17] S. S. Park, "Electrical Properties of Thin Film Type MLCC with Nickel Electrodes," *Ferroelectrics,* vol. 406, no. 1, pp. 75-82, 2010.

[18] Texas Instruments, *SLAU445G - MSP430FR4xx and MSP430FR2xx Family User's Guide,* Oct. 2014 [Revised Aug. 2016].

[19] C. A. Gossel, M. R. Callahan and D. Raskovic, "Development of an Electrolytic Silver Biocide Dosing System for Use in a Spacecraft Potable Water Bus," in *International Conference on Environmental Systems*, Charleston, 2017.

[20] Texas Instruments, "34420A Datasheet".

[21] Keysight, 34401A 6 1/2 Digit Multimeter Manual, May 2012.

[22] D. Patterson and J. Hennessy, Computer Organization and Design MIPS Edition, San Francisco: Morgan Kaufmann Publishers Inc., 2013.

[23] C.-L. Su and A. M. Despain, "Cache design trade-offs for power and performance optimization: a case study," in *International symposium on Low power design*, Dana Point, 1995.

[24] B. Jacob, S. Ng and D. Wang, Memory systems; Cache, DRAM, Disk, Burlington: Morgan Kaufmann Publishers, 2008.

[25] H. Cole, D. Dees, R. Daugherty, N. E. Weir, S. Manuel, J. R. Keller and S. S. Woodward, "Development and Verification of an Electrode System for Electrolytic Generation of

Silver Ion Biocide for the Space Station Internal Thermal Control System Fluid," in *31st International Conference On Environmental Systems* , Orlando, 2001.

[26] B. Shkedi, "International Space Station (ISS) Water Transfer Hardware Logistics," in *International Conference on Environmental Systems*, Norfolk, 2006.

[27] M. Callahan, N. Adam, M. Roberts and J. Garland, "Assessment of Silver Based Disinfection Technology for CEV and Future US Spacecraft," *SAE Transactions,* vol. 116, pp. 481-491, 2007.

[28] N. Adam, "Compatibility Study of Silver Biocide in Drinking Water with Candidate Metals for the Crew Exploration Vehicle Potable Water System," in *39th International Conference on Environmental Systems* , Savannah, 2009.

[29] "SciFinder; Chemical Abstracts Service," American Chemical Society, Columbus.

[30] B. Slote, E. Salley, D. Carr and M. Kimble, "Silver Ion Biocide Delivery System for Water Disinfection," in *46th International Conference on Environmental Systems*, Vienna, 2016.

[31] Garret AiResearch, "Development of an Electrolytic Silver-Ion Generator for Water Sterilization in Apollo Spacecraft Water Systems," Los Angeles, 1967.

[32] R. Chang and K. Goldsby, Chemistry, New York: McGraw-Hill Education, 2014.

[33] D. Neamen, Microelectronics: Circuit Analysis and Design, New York: McGraw-Hill, 2010.

[34] R. Thomas, Practical Guide to ICP-MS, Boca Raton: CRC Press, 2013.

[35] S. Kannankote and G. Rechnitz, "Selectivity Studies on Liquid Membrane, Ion-Selective Electrodes," *Analytical Chemistry,* vol. 41, no. 10, pp. 1203-1208, 1969.

[36] C. Hiskey, "Principles of Precision Colorimetry," *Analytical Chemistry,* vol. 21, no. 12, pp. 1440-1446, 1949.

[37] R. Waterbury, W. Yao and R. Byrn, "Long Pathlength Absorbance Spectroscopy: Trace Analysis of Fe(II) using a 4.5 m Liquid Core Waveguide," *Analytica Chemica Acta,* vol. 357, no. 1-2, pp. 99-102, 1997.

[38] M. Petala, V. Tsiridis, E. Darakas, I. Mintsouli, S. Sotiropoulos, M. Kostoglou, T. Karapantsios and P. Rebeyre, "Silver deposition on wetted materials used in the potable

water systems of the International Space Station," in *46th International Conference on environmental Systems*, Vienna, 2016.

[39] A. Demoz, E. Verpoorte and D. Harrison, "An equivalent circuit model of ion-selective membrane l insulator I semiconductor interfaces used for chemical sensors," *Journal of Electroanalytical Chemistry,* vol. 389, pp. 71-78, 1994.

[40] M. Kim, J. Lee, Y. Kim and Y. Ho Song, "An analysis of energy consumption under various memory mappings for FRAM-based IoT devices," in *IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018.

[41] Macronix International Co., Ltd., "MX25R4035F Datasheet," 2017.

[42] Cypress Semiconductor, "CY25B104Q," 2017.