Power Analysis

Background:

Our light monitoring system is composed of the STM32L432KC (STM) microcontroller and the TSL237 light sensor. The system is designed to take light samples every 15 minutes for 12 hours a day, with each sample taking up to 15 seconds. When the system is taking a light sample, it will be in one of three running modes, and while it is idle, the system will be in one of seven low-power modes. The following graphs show the power consumption of different combinations of run and low-power modes, generated by CubeMX. We are trying to determine the pair of run and low-power modes that will yield the best battery life with a CR2032 battery.

When testing each of the running modes, we made sure to use the highest CPU frequency that each mode can handle. We enabled the ADC, RTC, UART, a timer, and several GPIO peripherals. Additionally, for each run mode, we gave an additional 2 mA of consumption to account for the light sensor. For the low power modes, we enabled only the RTC and the UART if it is supported. In each plot, we let the system run for 15 seconds and then enter low-power mode for 900 seconds (15 minutes).

Analysis

RUN (Range 1)

The following figures 1-6 show the power consumption of the RUN power mode at Range 1 with multiple low-power modes. In each case, the CPU is running at a max of 80 MHz. However, the faster clock speed means that the STM draws more power. The code is running from FLASH, so that also takes extra current to maintain. Figure 6 shows the longest battery life approximation with STANDBY as the low-power mode at 1 month, 16 days and 15 hours. The other run modes last much longer across the board. Performance-wise, we receive little benefit from the extra clock speed because our system does not compute much, it just stores data. Thus, it does not seem worth sacrificing battery life for the extra speed of RUN at Range 1.

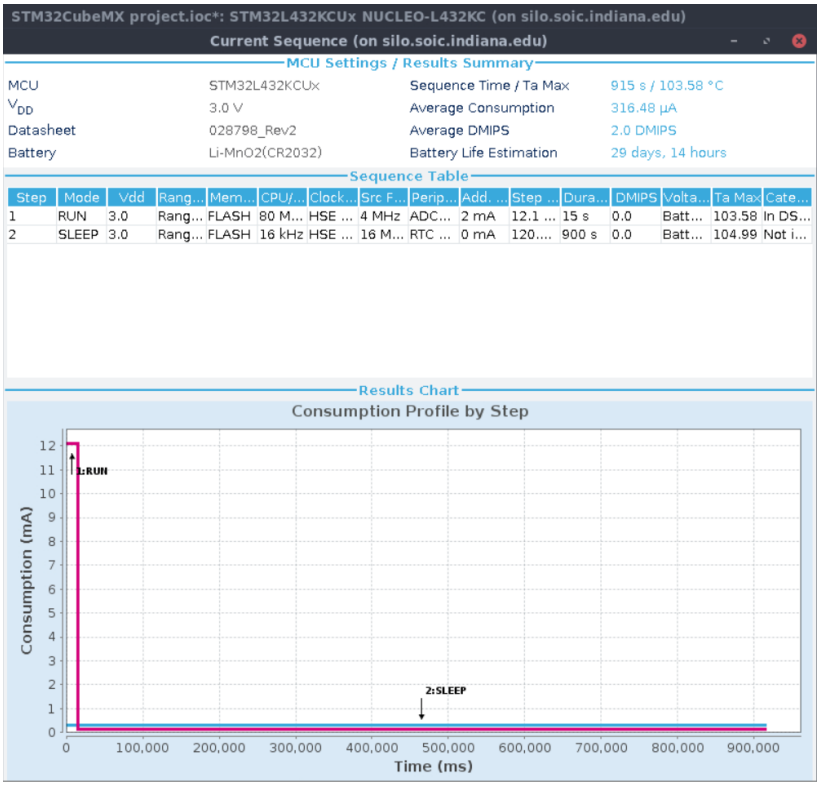

Figure 1: RUN (Range 1) with SLEEP

SLEEP consumes the most current out of all the low-power modes at 120 uA, but as a result the wake-up time is the fastest at 6 clock cycles. The LPUART and RTC can still be enabled, and SRAM is retained.

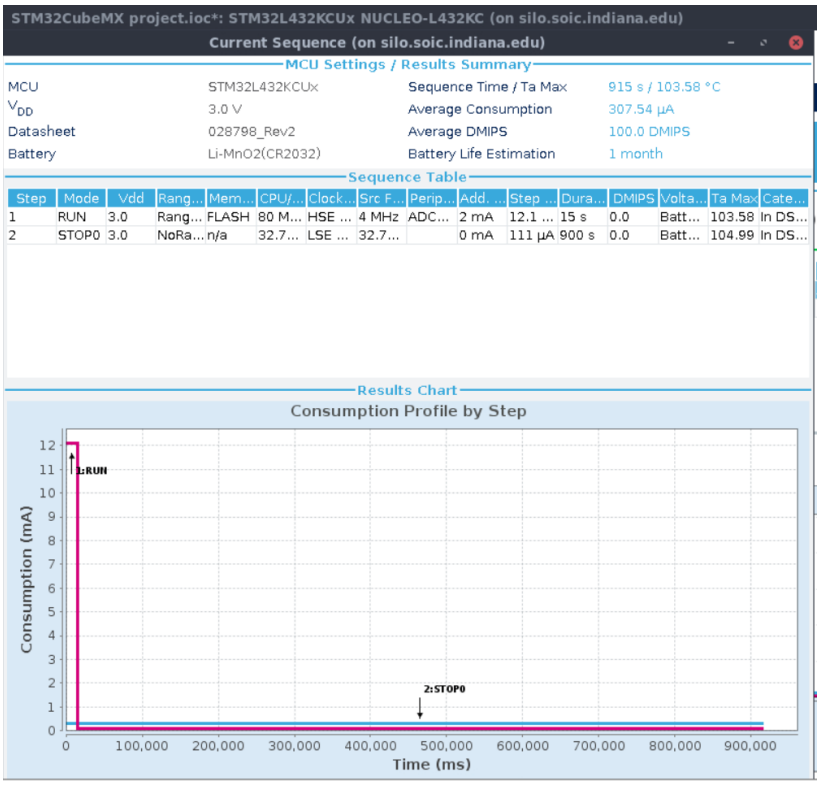

Figure 2: RUN (Range 1) with STOP 0

STOP 0 consumes slightly less current than SLEEP at 111 uA, and its wakeup time is slower at 0.7 us. The LPUART and RTC can still be enabled, and SRAM is retained. Battery life is improved by about 10 hours compared to using SLEEP.

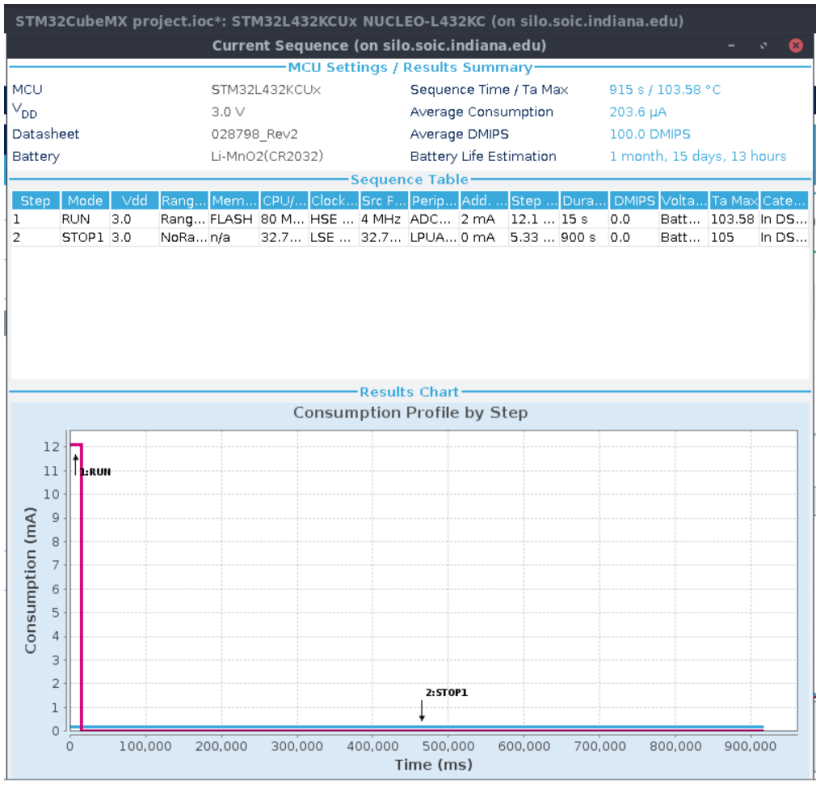

Figure 3: RUN (Range 1) with STOP 1

STOP 1 consumes even less current than STOP 0 at 5.3 uA, and its wakeup time is 4 us. The LPUART and RTC can still be enabled, and SRAM is retained. The difference between STOP 0 and STOP 1 in terms of power consumption is significant though, giving us an extra 15.5 days before the battery dies.

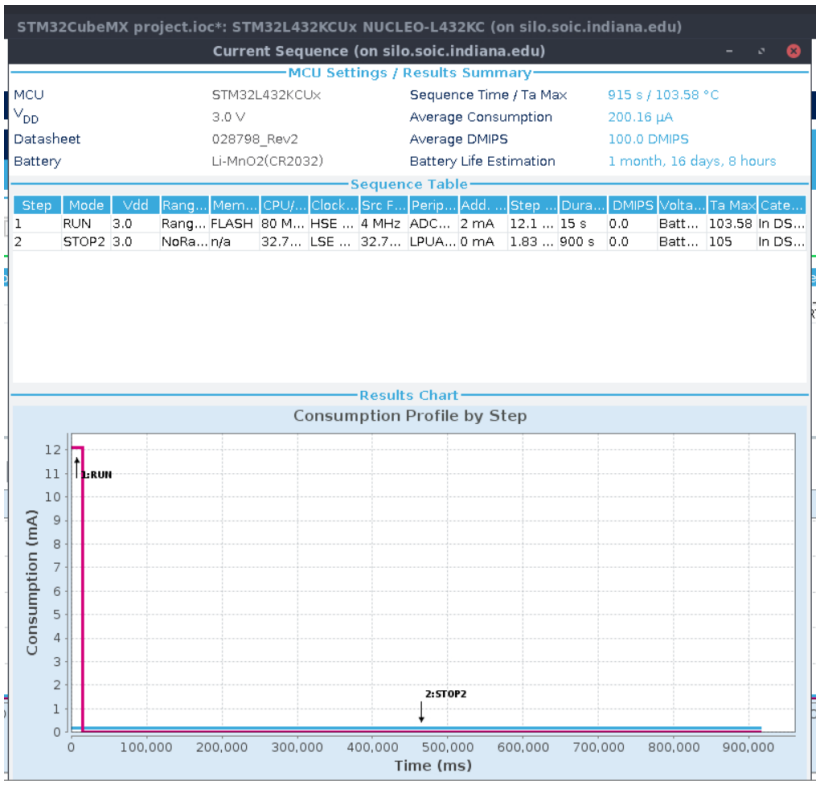

Figure 4: RUN (Range 1) with STOP 2

STOP 2 consumes less current than STOP 1 at 1.8 uA, and its wakeup time is 5 us. The LPUART and RTC can still be enabled, and SRAM is retained. STOP 2 gives us an additional 19 hours of battery life over STOP 1.

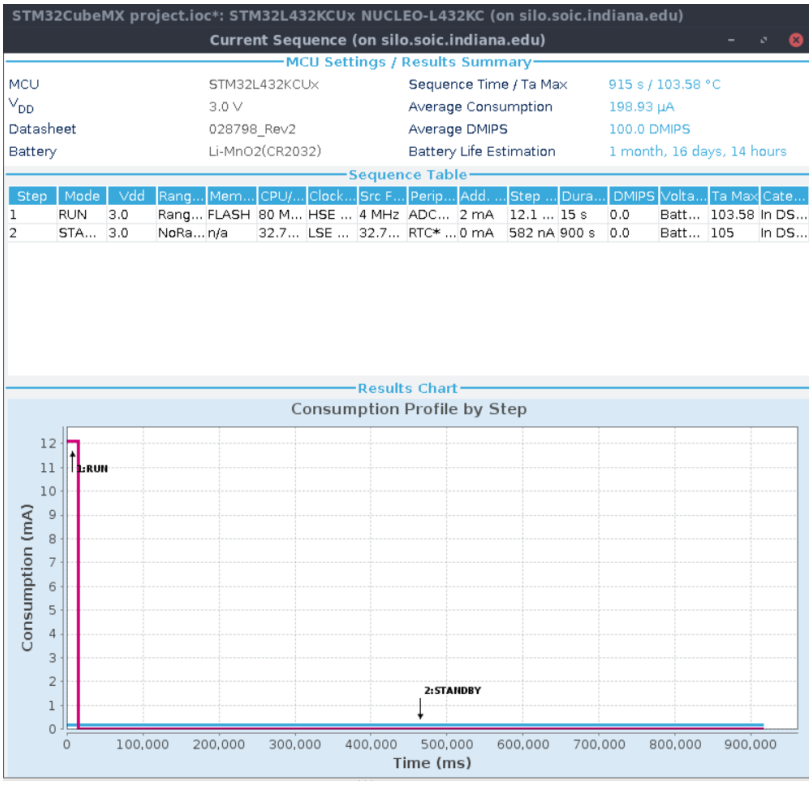

Figure 5: RUN (Range 1) with STANDBY + 32KB SRAM

STANDBY with SRAM consumes even less current than STOP 2 at 582 nA, but its wakeup time is more than doubled at 14 us. The LPUART can no longer stay enabled, but RTC is still active and SRAM is retained. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. For the effort, we only gain 6 hours of extra battery life compared to STOP 2.

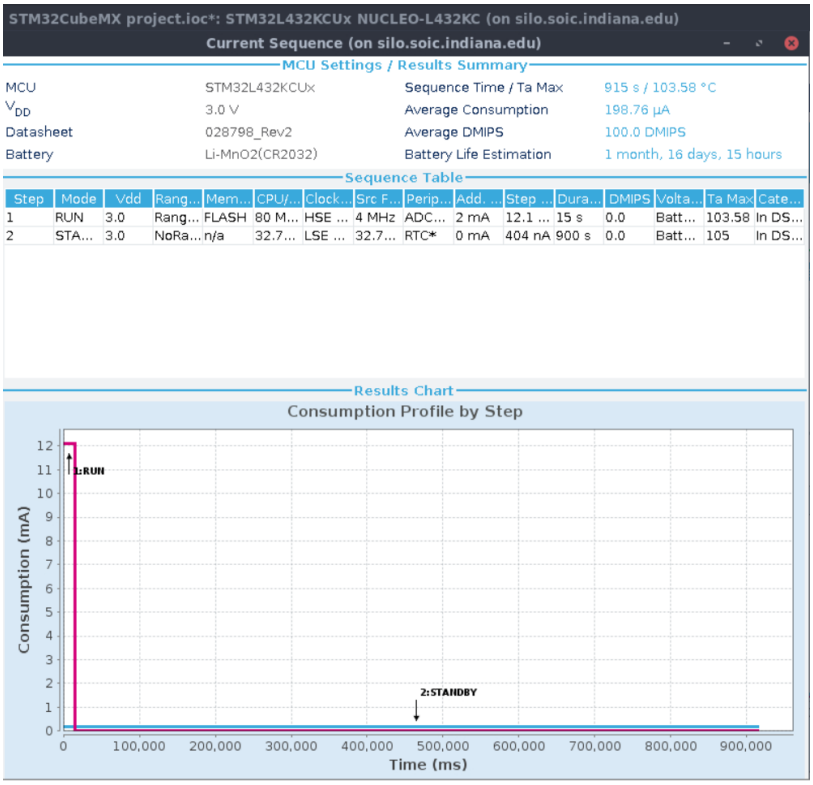

Figure 6: RUN (Range 1) with STANDBY

STANDBY without SRAM consumes less current than before at 404 nA, but its wakeup time is still at 14 us. The LPUART can no longer stay enabled and SRAM is not retained, but RTC is still active. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. This only gains us 2 hours of battery life.

RUN (Range 2)

The following figures 7-12 show the power consumption of the RUN power mode at Range 2 with different low-power modes. Range 2 is different from Range 1 because the CPU can only run at a max of 26 MHz, and the code is executing from SRAM instead of FLASH. If the low-power mode does not retain SRAM, we will need to move our code to FLASH before entering. However, the benefit to the battery life for these drawbacks is significant. Figure 12 shows the longest battery life approximation with STANDBY as the low-power mode at 3 months, 25 days and 18 hours.

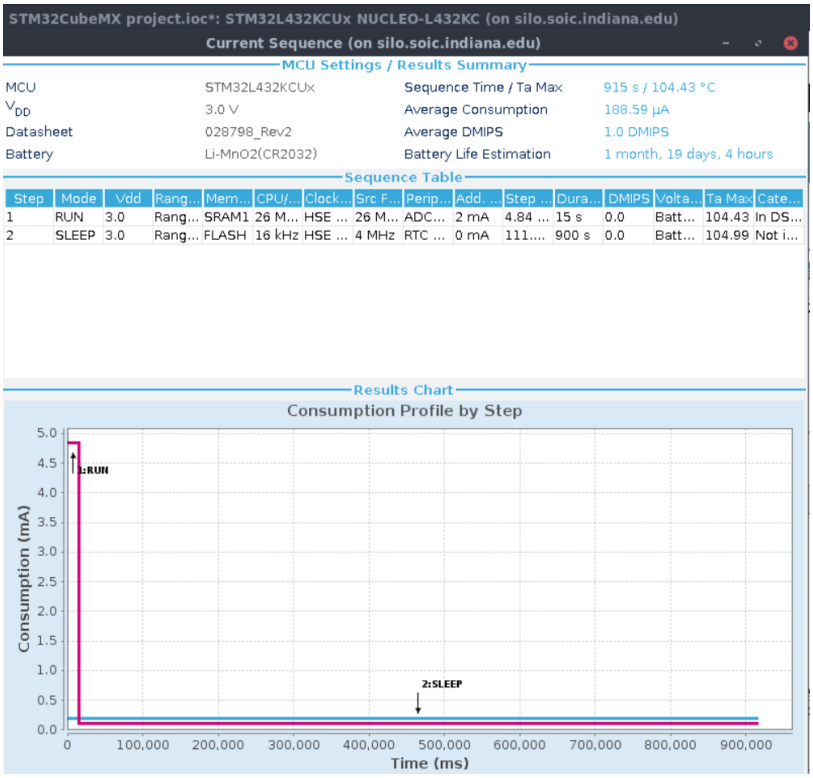

Figure 7: RUN (Range 2) with SLEEP

As mentioned before, SLEEP consumes the most current out of all the low-power modes at a little over 111 uA, but as a result the wake-up time is the fastest at 6 clock cycles. The LPUART and RTC can still be enabled, and SRAM is retained.

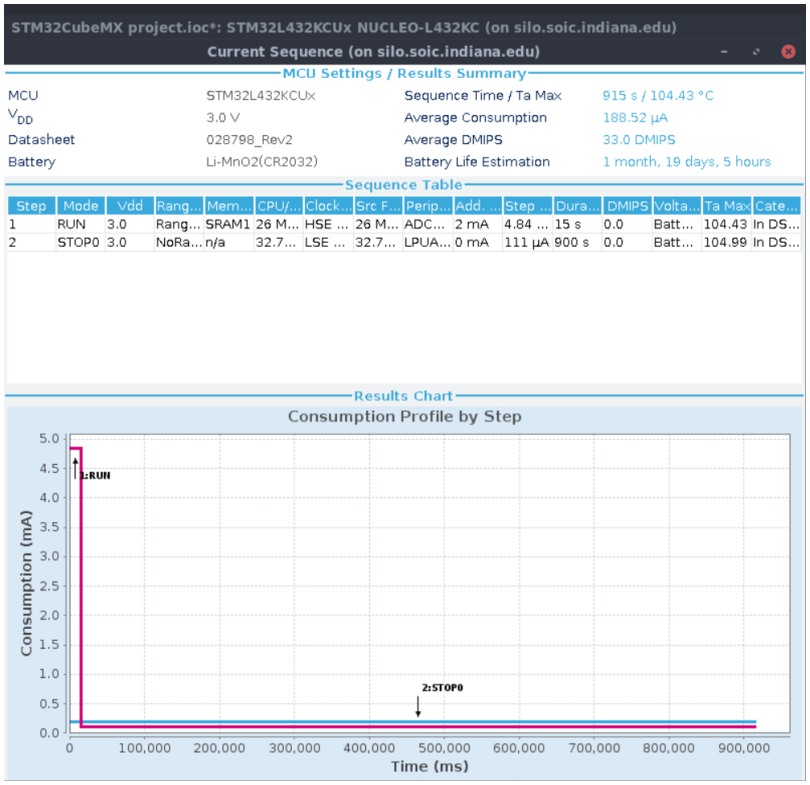

Figure 8: RUN (Range 2) with STOP 0

STOP 0 consumes roughly the same current as SLEEP at 111 uA, and its wakeup time is slower at 0.7 us. The LPUART and RTC can still be enabled, and SRAM is retained. Battery life is improved by about 1 hour compared to using SLEEP.

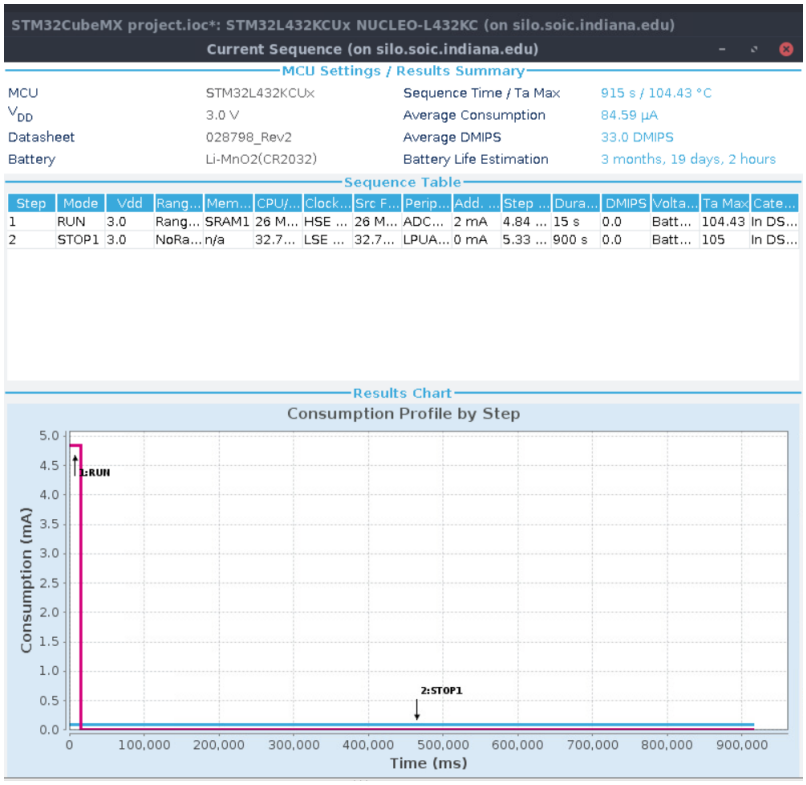

Figure 9: RUN (Range 2) with STOP 1

As mentioned before, STOP 1 consumes much less current than STOP 0 at 5.3 uA, and its wakeup time is 4 us. The LPUART and RTC can still be enabled, and SRAM is retained. The difference between STOP 0 and STOP 1 in terms of power consumption is significant though, giving us an extra 2 months before the battery dies.

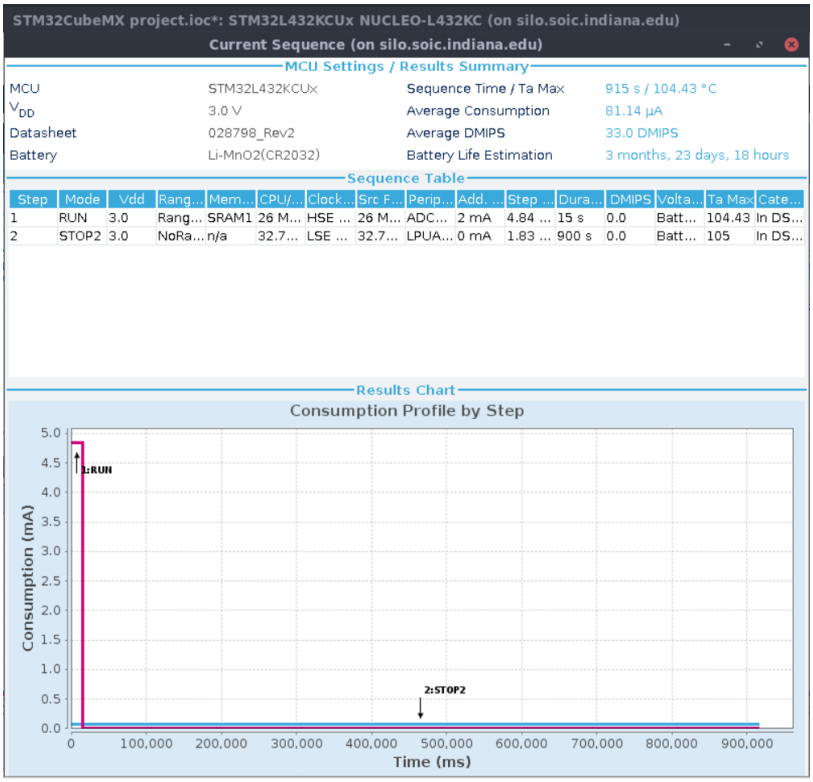

Figure 10: RUN (Range 2) with STOP 2

As mentioned before, STOP 2 consumes less current than STOP 1 at 1.8 uA, and its wakeup time is 5 us. The LPUART and RTC can still be enabled, and SRAM is retained. STOP 2 gives us an additional 4 days and 16 hours of battery life over STOP 1.

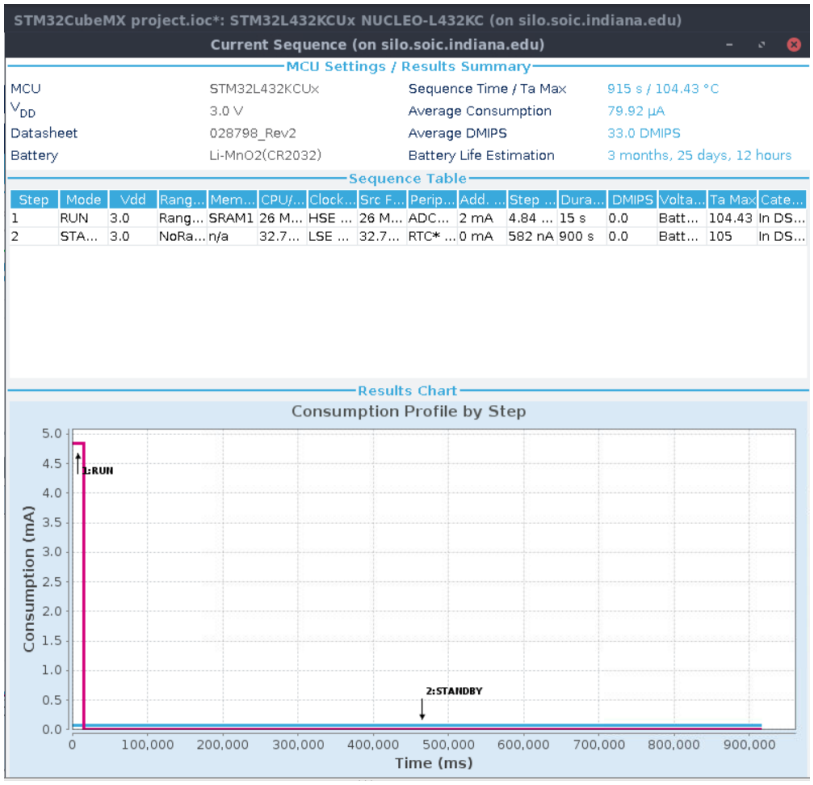

Figure 11: RUN (Range 2) with STANDBY + 32KB SRAM

As mentioned before, STANDBY with SRAM consumes even less current than STOP 2 at 582 nA, but its wakeup time is more than doubled at 14 us. The LPUART can no longer stay enabled, but RTC is still active and SRAM is retained. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. For the effort, we gain 1 day and 18 hours of extra battery life compared to STOP 2.

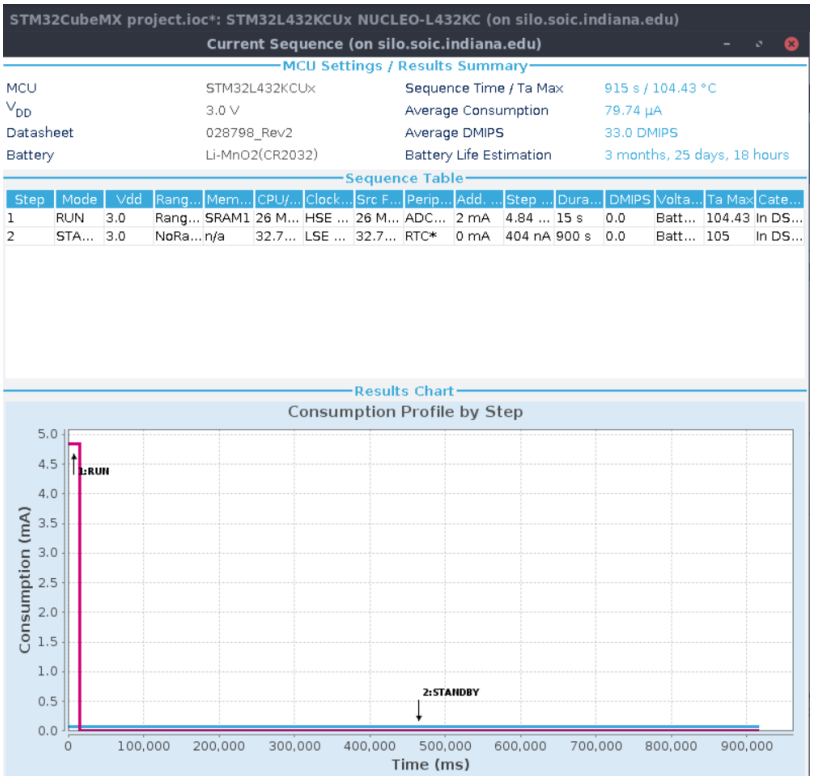

Figure 12: RUN (Range 2) with STANDBY

As mentioned before, STANDBY without SRAM consumes less current than before at 404 nA, but its wakeup time is still at 14 us. The LPUART can no longer stay enabled and SRAM is not retained, but RTC is still active. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. Additionally, upon entering the state we must store the code in FLASH, and upon exiting we must load the code from FLASH into SRAM. This only gains us 6 hours of battery life.

LPRUN

The following figures 13-17 show the power consumption of the LPRUN power mode with different low-power modes. LPRUN has unique access to LPSLEEP, but it does not support STOP 2. The CPU can only run at a max of 2 MHz. The slower clock speed means that the STM conserves a lot of power. The code is running from FLASH, so that also takes extra current to maintain. However, the benefit to the battery life for this drawback is significant. Figure 17 shows the longest battery life approximation with STANDBY as the low-power mode at 7 months, 24 days and 15 hours. The issue will be whether 2 MHz is enough to perform all our operations in a reasonable amount of time.

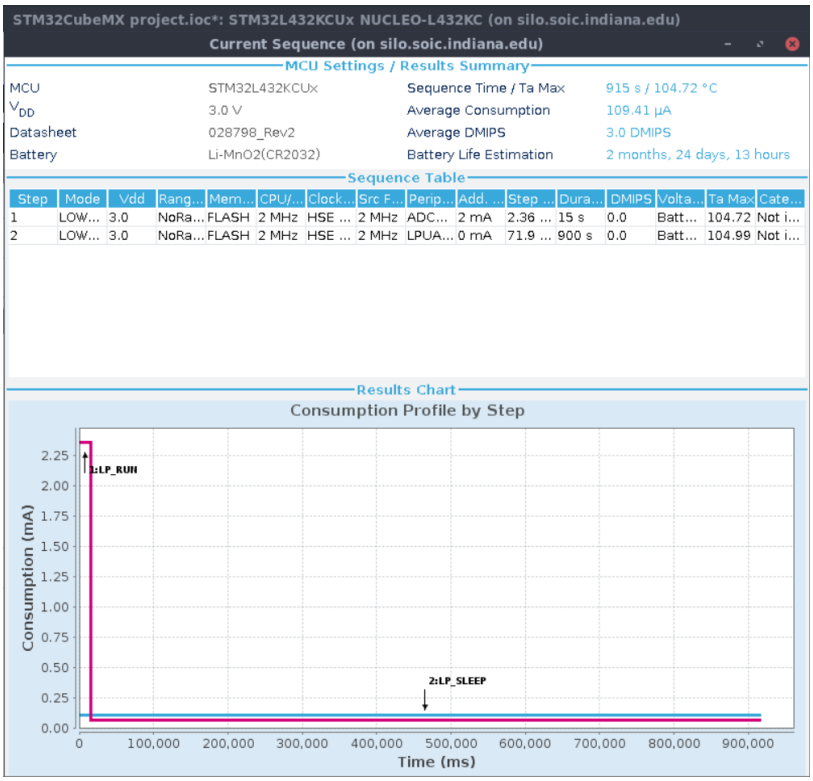

Figure 13: LPRUN with LPSLEEP

LPSLEEP can only be used with LPRUN and consumes very little current at just 72 uA. It also wakes up from sleep very quickly at just 6 clock cycles. The LPUART and RTC can still be enabled, and SRAM is retained.

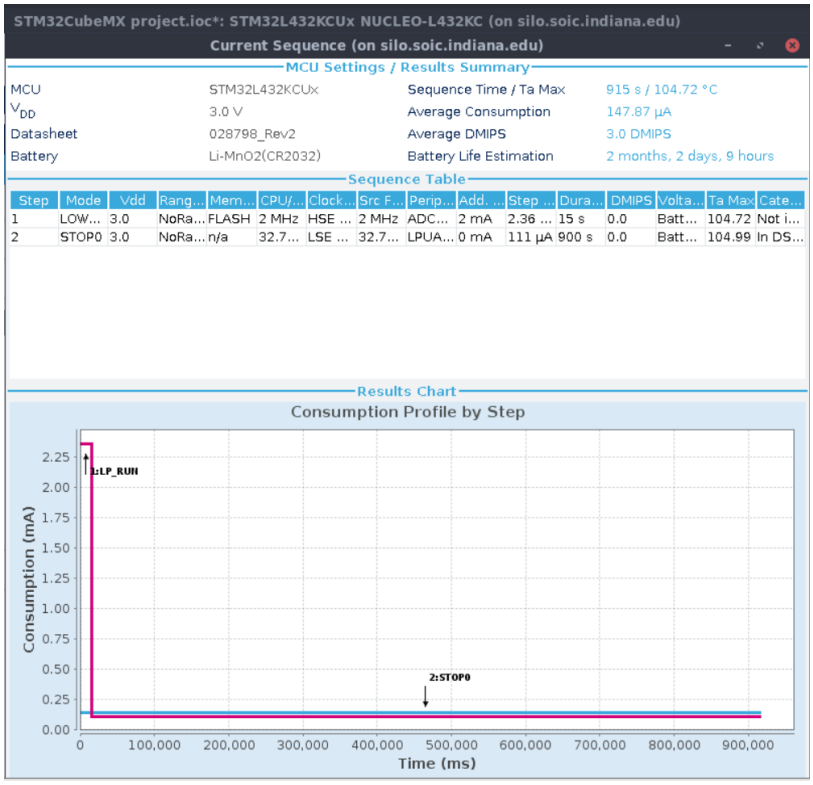

Figure 14: LPRUN with STOP 0

STOP 0 is surprisingly worse than LPSLEEP when combined with LPRUN, consuming 111 uA of current and with a slower wakeup time of 0.7 us. The LPUART and RTC can still be enabled, and SRAM is retained. Battery life is worse than LPSLEEP by 22 days; it is unlikely we would choose STOP 0 over LPSLEEP when using LPRUN.

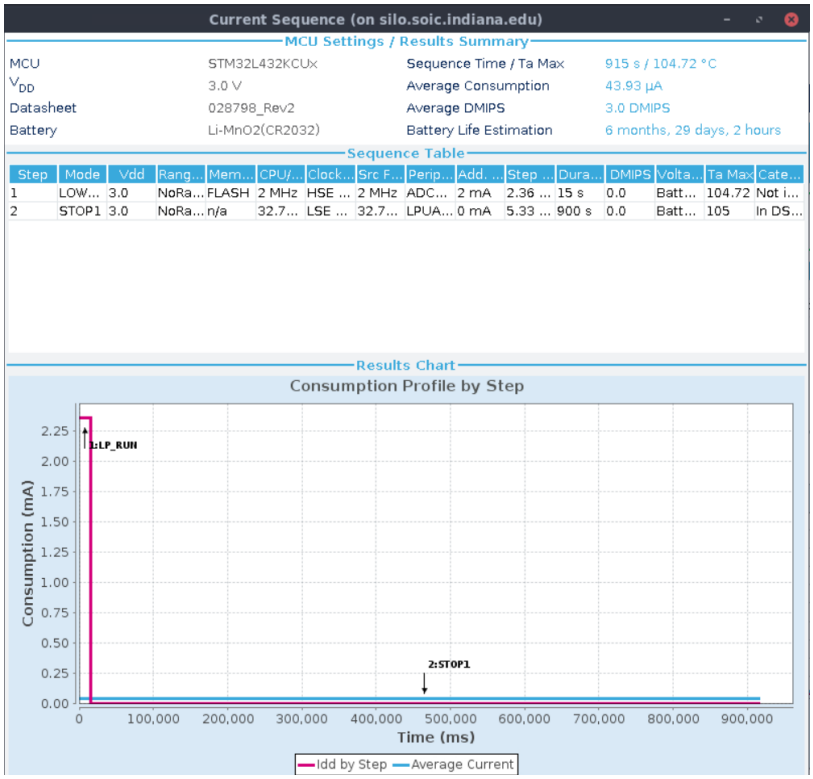

Figure 15: LPRUN with STOP 1

As mentioned before, STOP 1 consumes much less current than STOP 0 at 5.3 uA, and its wakeup time is 4 us. The LPUART and RTC can still be enabled, and SRAM is retained. The difference between STOP 0 and STOP 1 in terms of power consumption is significant though, giving us almost 5 extra months before the battery dies.

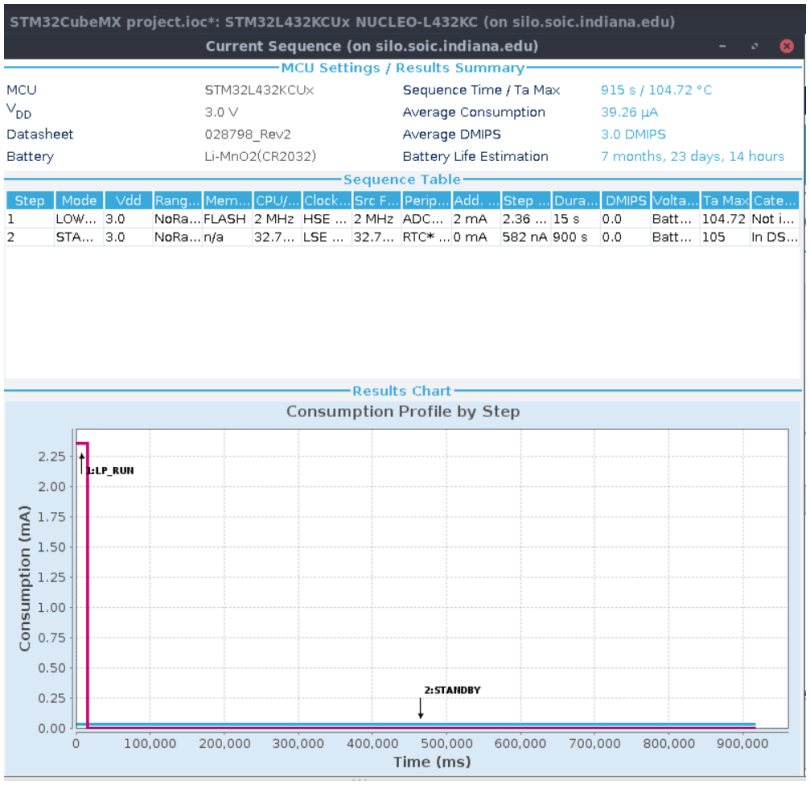

Figure 16: LPRUN with STANDBY + 32KB SRAM

STANDBY with SRAM consumes even less current than STOP 1 at 582 nA, but its wakeup time is more than tripled at 14 us. The LPUART can no longer stay enabled, but RTC is still active and SRAM is retained. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. For the effort, we gain about 24 days of extra battery life compared to STOP 1.

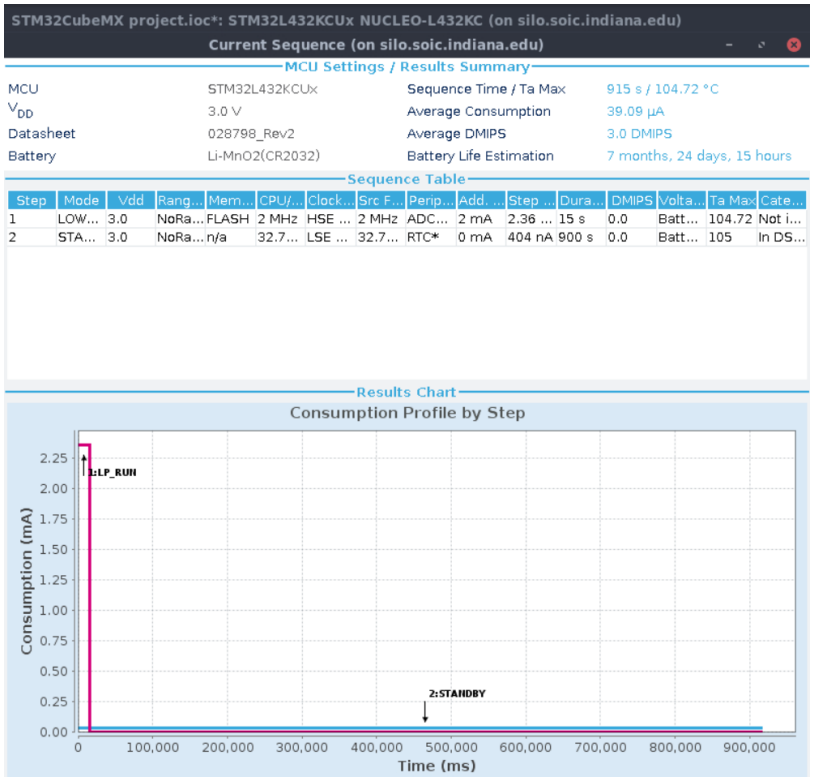

Figure 17: LPRUN with STANDBY

As mentioned before, STANDBY without SRAM consumes less current than before at 404 nA, but its wakeup time is still at 14 us. The LPUART can no longer stay enabled and SRAM is not retained, but RTC is still active. This means if we want to communicate with the device over UART, we must wait 15 minutes before the system is back in the run mode. This only gains us 1 day and 1 hour of battery life.

Conclusion

For the run mode, RUN at Range 2 provides a reasonable clock speed for decent battery performance. LPRUN would provide much better battery performance, but 2 MHz may be too slow for calculations and will make the response over UART slow. For the low-power mode, STOP 2 seems ideal because it provides the best battery performance while still allowing us to communicate over UART and retain the state of SRAM. STANDBY options would give us better battery life, but it disables UART, meaning that we only have a small 15 second window to connect with the system in the RUN state. As figure 10 states, the RUN (Range 2) coupled with STOP 2 gives 3 months, 23 days and 18 hours of battery life. Moreover, if we do not need the full 26 MHz for our computations, we could lower the clock rate to achieve even better battery life.