Application Note

Application Note AN005

Active Interrupt Proven to Lower Power Consumption

This publication contains proprietary information which is subject to change without notice and is supplied “as is”, without any warranty of any kind.

Revision History

Revison Number	Date	Tasks
A	February, 2020	Reformat to new Application Note template and renumbered from ADAPP005

Introduction

This note describes Adesto's "Active Interrupt" feature, and illustrates how to implement this feature. The Active Interrupt works in conjunction with the host MCU to reduce the number of MCU cycles used for overhead, and therefore reduces the energy consumption of the entire system.

It is common for the peripheral devices surrounding an embedded MCU to have the ability to interrupt the MCU to request attention, or to indicate when data is available for the host to collect and process. However, today's industry-standard serial flash devices require constant polling by the MCU to maintain throughput. The Adesto Active Interrupt upgrades the Serial Flash memory device to an intelligent peripheral. Subsequently, when the serial flash memory device completes an internal program or erase operation, this intelligent memory device will issue an interrupt to the MCU to indicate that the selected operation is complete. This capability optimizes both power use and execution time.

The use of this feature yields a system that has a longer operating life for a given energy source, or requires fewer (or smaller) batteries. To demonstrate the benefit of Active Interrupt in lowering system-level energy consumption during Program and Erase execution, this note will present test data obtained from a third party power measurement system.

1. The Problem

For systems that use industry-standard serial flash today, the only way the MCU (host) can determine the status of the device while a program (or erase) operation is underway, is to "poll" the memory device's status register. To do this the MCU must read the register, extract the Ready/Busy bit, and then repeat until the pertinent bit is set. This "polling" process is time consuming and MCU-resource heavy.

The only alternative to "polling" is to put the MCU into a sleep mode for the predetermined MAXIMUM program, or MAXIMUM erase time. Upon waking up, the MCU can start "polling" the memory device. This process results in system inefficiency, as the actual time for these operations is usually much shorter than the MAXIMUM time specified.

While the flash device is performing an Internal Programming Cycle, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register [Byte 1, bit 0; the "Ready/Busy Status"] be polled rather than waiting the 2.5ms Max tPP time to determine if the data bytes have finished programming*.

The MCU can read Status Register Byte 1, with the Command: 05h, and then branch based on the value of bit 0. The MCU continues in this loop until the state of the RDY/BSY bit changes from a logical "1" to a logical "0."

Some design engineers refer to the "polling" method as, "The kids in the car repeatedly asking, 'Are we there yet?'"

2. Overcoming Power Consumption and Time Challenges

Adesto's Active Interrupt feature, which is available on all Adesto serial flash devices starting at 2 Mbit (AT25DF, AT25XE and AT25XV)., solves both the power-consumption and the time efficiency problem simultaneously. This is accomplished by adding a simple command at the end of the Program or Erase request from the MCU.

The Adesto Active Interrupt upgrades the Serial Flash memory device to an intelligent peripheral. When a Serial Flash memory device completes an internal program or erase operation, this intelligent memory device will issue an interrupt to the MCU to indicate that the selected operation is complete, optimizing both power use and execution time.

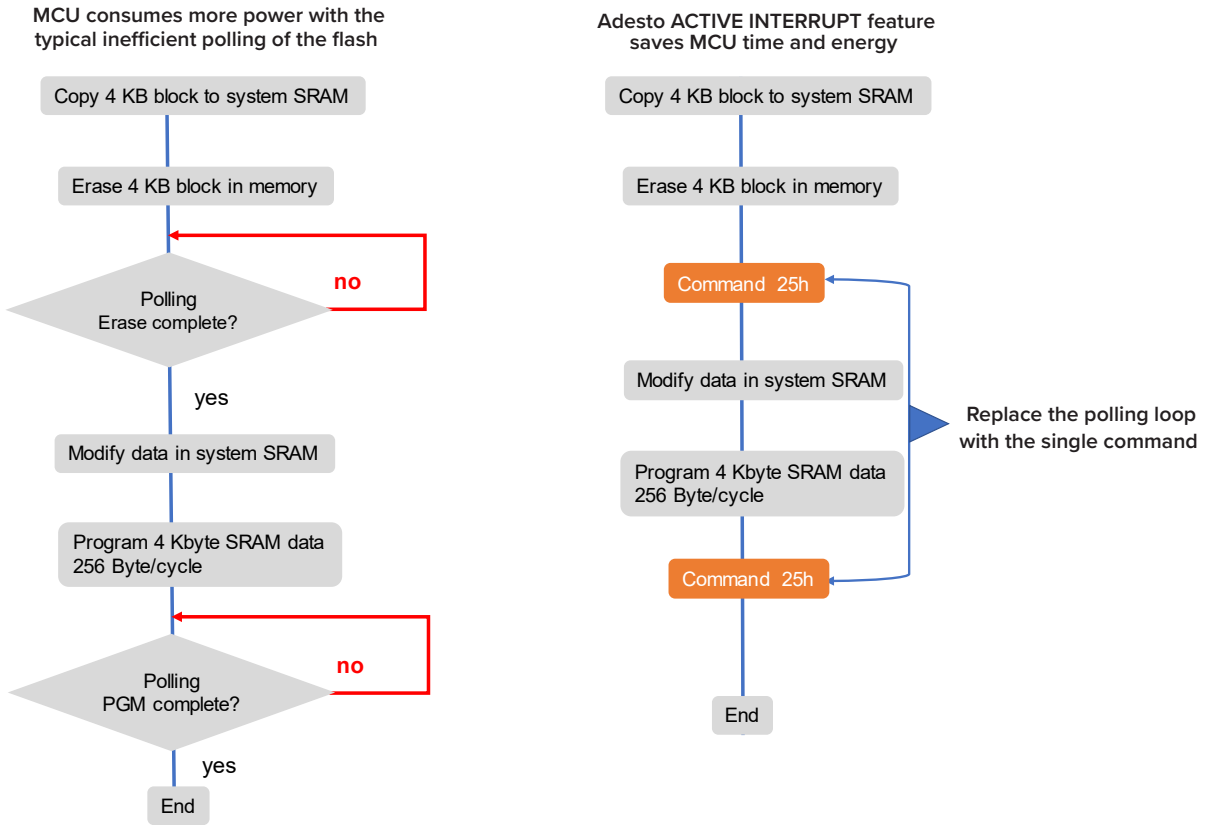

figure 1.

Adesto's Active Interrupt capability is implemented in the Fusion Family (AT25DF, AT25XE, and AT25XV) of serial flash memory devices, in the 2Mbit and 4Mbit densities. Active Interrupt is a command-enabled option that is used when the host initiates a program or erase operation to the serial flash. By issuing the Active Interrupt command, the MCU (the SPI bus master) can then enter a low-power sleep state during the time the flash is performing the program/erase operation.

The Adesto memory device with Active Interrupt will drive the flash memory's SO line (which is the MCU's SI line) from high to low at the completion of the program/erase cycle. The MCU SI line can be configured as an interrupt input to wake up the MCU from the sleep state and continue normal operation. The Active Interrupt feature brings the memory device into the realm of an intelligent peripheral device, interactively communicating with the MCU as required to signal the end of a program/erase cycle.

The Active Interrupt capability works for all internal programming operations and all internal erase operations on the Adesto serial flash memory device. The MCU needs to support edge-triggered interrupts on its SI input line from the memory device; or alternatively, the SO output signal from the Serial Flash can be tied to an alternative MCU GPIO pin that supports edge-triggered interrupt capability.

This one feature will significantly reduce system energy consumption when performing program and erase operations by allowing the MCU (host) to sleep during these periods. It will reduce inefficiencies in time-out based solutions and reduce software complexity and size.

3. Demonstration of Power Savings

The use of Adesto's Active Interrupt feature yields a system that has a longer operating life for a given energy source, or requires fewer (or smaller) batteries. Test data obtained from a third party power measurement system demonstrates the benefit of Active Interrupt in lowering system-level energy consumption during Program and Erase execution.

To demonstrate the power savings effect of Active Interrupt on a system, we have used the Silicon Labs' EFM32LG-STK3600 Leopard Gecko starter kit, shown in figure 2.

The kit contains an Energy Monitoring system that provides the system's current consumption vs. time in a graphical display format, as shown in figures 6 and 7.

figure 2. EFM32LG-STK3600

The starter kit is connected to an EMSENSR board by its expansion header. A small breakout board, with the SPI flash mounted on it, is installed on the EMSENSR board.

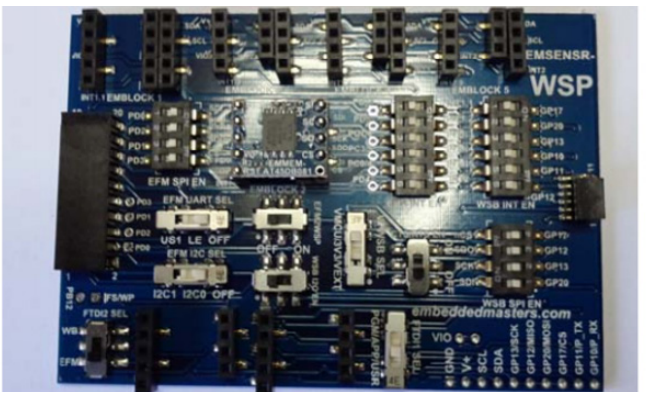

figure 3. EMSENSR Board

The EFM32 Starter Kit runs the Simplicity Studio software package, which contains the energyAware Profiler. This is the PC-side interface to the Advanced Energy Monitor. It allows energy-debugging and power profiling of application code running on the attached hardware.

With this hardware and software system, we can run various scenarios, and see the power usage effects resulting from each scenario. We can then compare the power profiler output and determine the system-level power usage effect directly. For this article, we looked at two scenarios: first the system without the Active Interrupt feature, and second, the system using the Active Interrupt feature.

Conceptually, the two scenarios can be represented by the following two drawings. First we have the traditional “polling” method, and the power usage associated with that method. Note the MCU power remains constant during the Flash Erase/Program cycle, due to its need to poll the flash for completion.

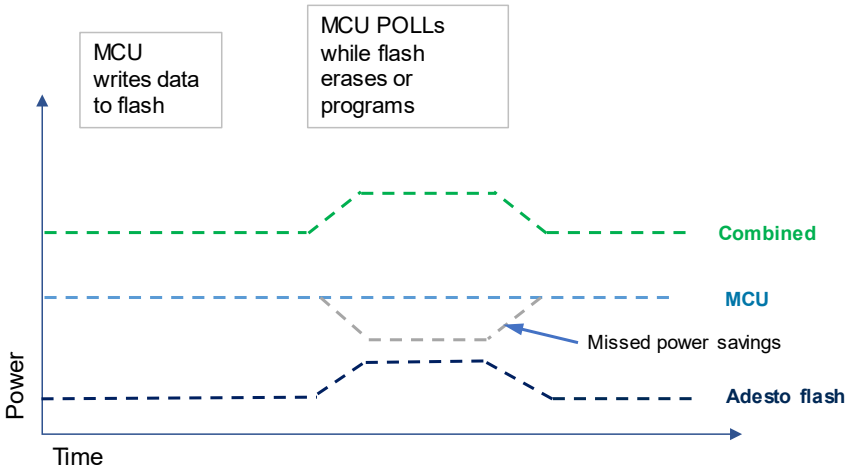

figure 4. Traditional polling method

Next we have the system employ the “Active Interrupt” feature, and the power reduction associated with that method. Note the MCU goes into low-power sleep mode during the flash Erase/Program cycle, because the Flash will wake it up upon completion of the cycle.

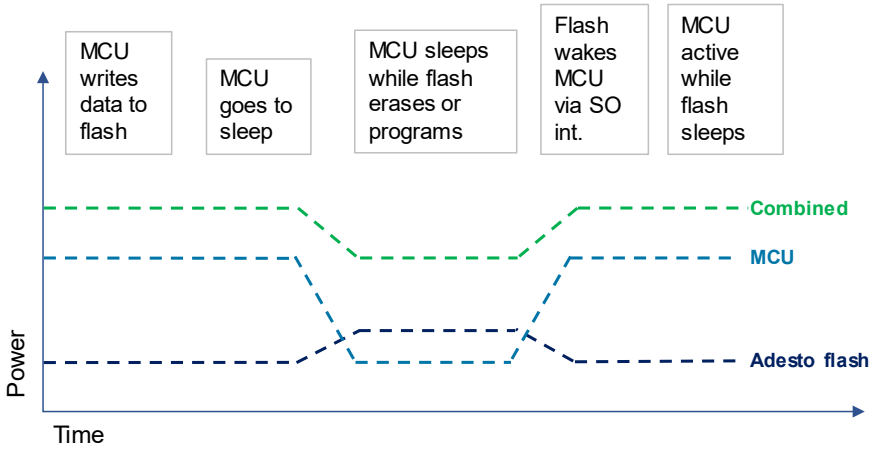

figure 5. Use of “Active Interrupt” Lowers System Power Use

Now let's use the Silicon Labs' EFM32LG-STK3600 Leopard Gecko starter kit and its Simplicity Studio software package, and run these two scenarios using the energyAware Profiler.

For our first scenario, we have the system program the SPI Flash with 256Bytes of data, while using the "polling" method to determine the completion of the Internal Programming Cycle. The Power Profile for this "polling" method is shown in Figure 6.

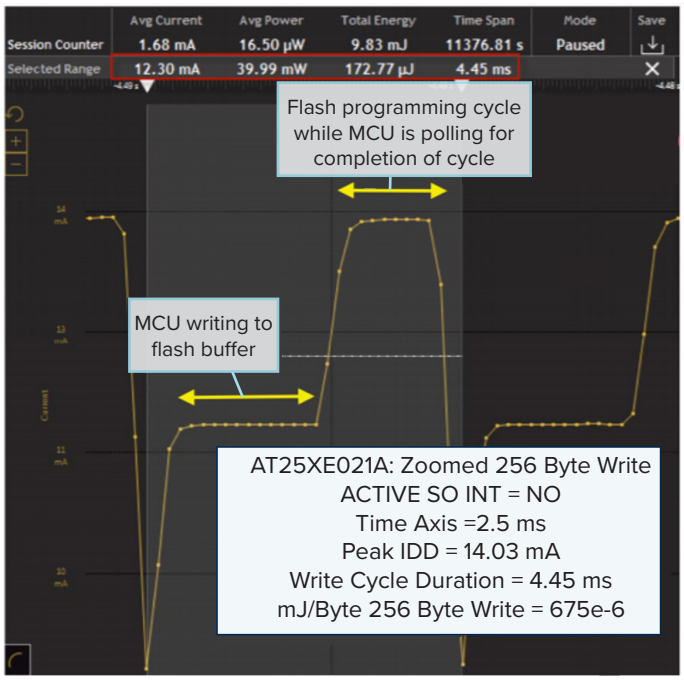

figure 6. Traditional "Polling" Method Power Consumption

After applying the settings according to one of the options described above, check "Verify download" as shown in Figure3 above. Now you can use the EcoXiP Flash loader with verification.To load the program image, click the Download and Debug buttons.

We clearly see the power usage rise as the MCU is loading data into the SPI Flash Write Buffer, then the power rises even higher, while the SPI flash undergoes its Internal Program Cycle and the MCU is "polling" the flash for completion of the cycle.

Note the peak IDD current draw for the system is 14.03mA during this phase. When the flash completes the Internal Program Cycle, the power drops briefly, until the MCU starts to load the next 256Bytes of data into the SPI Flash Write Buffer for the next cycle.

In figure 6, the Power Profiler Screen Shot shows the power usage effect of the MCU "polling" the SPI flash during an Internal Programming cycle. Both the MCU and the flash are drawing high power, because the MCU is running cycles to read the Ready/ Busy Status Register (polling).

4. The Power Savings Afforded by Active Interrupt

For our second scenario we have the system program the SPI Flash with the same 256Bytes of data, but instead of using the "polling" method to determine the completion of the Internal Programming Cycle, we will use the "Active Interrupt" command and place the MCU into low-power sleep mode during the Internal Programming Cycle. The Power Profile for the "Active Interrupt" method is shown in figure 7.

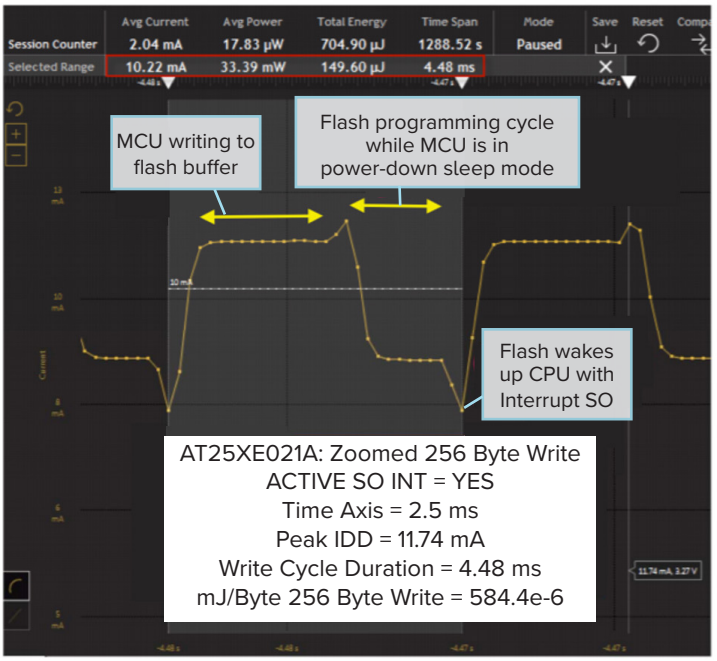

figure 7. Power Savings Achieved by "Active Interrupt"

Note that the beginning of the cycle in this scenario is identical to the first scenario, the power used by the MCU to load the 256Bytes of data into the SPI Flash write buffer is the same. We then see the total system power start to rise as the SPI Flash starts its Internal Programming Cycle, but then, the MCU goes into low-power sleep mode, and the total system power DROPS. Note the peak IDD current draw is 11.74mA with Active Interrupt, vs. 14.03mA during this same phase, when not using Active Interrupt. The difference is the power saved by letting the MCU sleep during the SPI Flash Internal Programming Cycle.

By letting the MCU go to low-power sleep mode during the SPI Flash Internal Programming Cycle, we dramatically lower the total system-level power and we avoid a high current spike that is detrimental to the battery life of many battery-operated systems.

Conclusion

The data in this article conclusively demonstrates the significant power-savings achieved by use of the simple Active Interrupt feature. The command is easy to implement in the user's code, and many MCUs have power-down modes and edge-triggered interrupts available to take advantage of the Active Interrupt feature.

In many applications that are powered from AA batteries, coin cells, or energy harvesters; minimizing system-energy usage is critical. The Active Interrupt works in conjunction with the host MCU to reduce the number of MCU cycles used for overhead, and therefore reduces the energy consumption of the entire system. Adesto's Active Interrupt feature is only one of many features that Adesto's application-specific memory products bring to a system to lower Bill of Materials costs, decrease power consumption, and increase system performance, all at the same time.

*Timing specifications from the AT25DF021A data sheet