AN725

Advanced Low Power Techniques for SiM3L1xx Devices

1. Introduction

The system power budget of a low power system has two components: active and low power mode. Active mode periods include an active core executing code or a larger number of active peripherals. In low power mode periods, the core enters a sleep state and fewer peripherals are active.

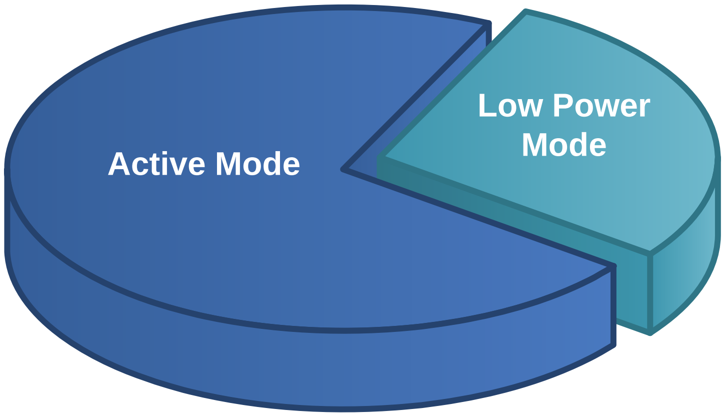


Figure 1. Defining the Power Budget—Active and Low Power Modes

SiM3L1xx devices have several features to address reducing power consumption in all operational modes to achieve a longer product lifetime in battery-operated systems. This document addresses each of these features and provides guidelines for achieving low power consumption in a variety of configurations and applications.

2. Key Points

This key topics of this document are as follows:

- How to reduce active mode time and power consumption
- How to reduce low power mode power consumption
- Measuring the low power modes on an SiM3L1xx MCU card
- General power-saving tips

3. Relevant Documentation

Precision32™ Application Notes are listed on the following website: www.silabs.com/32bit-appnotes.

- AN666: Usage Guide for e SiM3U1xx,SiM3C1xx, and SiM3L1xx DMA and DTM Modules
- AN720: Precision32™ Optimization Considerations for Code Size and Speed
- AN667: Getting Started with the Silicon Labs Precision32 IDE
- AN670: Getting Started with the Silicon Labs Precision32 AppBuilder

4. Reducing Active Mode Power Consumption

In active mode, the core is fetching instructions from memory and executing those instructions, and a large number of peripherals may be active at once. This section discusses ways to reduce the SiM3L1xx device power consumption in active mode (Normal, PM1, PM4, or PM5).

4.1. Dynamic AHB/APB Clock Scaling

One of the easiest ways to reduce overall system power consumption is to reduce the active mode time, which maximizes the amount of time spent in the low power mode.

If the longest path to the next low power mode is the execution of code (e.g., a math algorithm), then it is typically beneficial to run the AHB clock at the fastest speed possible to reduce the time spent in active mode.

If the longest path to the next low power mode is the transfer or collection of data through a peripheral, it is best to run the clock at the lowest speed required for the peripheral. For example, if using the UART at 115200 baud and the core is waiting for data to finish transferring, then running the AHB and APB at the slowest clock to achieve 115200 baud may be the lowest power configuration. However, if data is being transferred memory to memory by the DMA, running the clocks at the fastest speed possible yields the lowest power consumption.

Due to the clock system of the SiM3L1xx devices, the AHB and APB clocks can be dynamically changed quickly and easily using the CLKCTRL module based on the needs of the application.

4.2. Using the DMA and DTM Modules

The Direct Memory Access (DMA) and Data Transfer Manager (DTM) modules help move data without core intervention. This reduces overall power consumption by removing the power consumed during flash accesses. Additionally, since the Cortex-M3 is a load-store architecture where data is loaded into and out of registers only, multiple instructions are required to move data from one area of memory to another, so the DMA and DTM may be faster than data moves by the core, depending on the AHB load.

Instead of performing the data moves, the core can either sleep using wait-for-interrupt (WFI) or wait-for-event (WFE) instructions, or the core can perform other tasks in parallel, reducing the active mode time.

For more information on how to use the DMA and DTM modules, see “AN666:Usage Guide for e SiM3U1xx,SiM3C1xx, and SiM3L1xx DMA and DTM Modules” on the Silicon Labs 32-bit application notes website: www.silabs.com/32bit-appnotes.

4.3. Code Optimization

There are several different coding techniques, compiler, and library options available for the Precision32 devices. These options will change both code size and execution speed, which may result in power consumption savings in systems which aim to execute code in active mode as quickly as possible before entering a low power state. In addition to getting to the low power mode more quickly, changing the project settings may lead to smaller code footprints, compacting the code into a smaller area with fewer memory accesses. The power consumption benefits of code optimization for speed or size will vary depending on the project requirements and code.

For more information on how to write efficient code and the various optimization settings, see”AN720: Precision32™ Optimization Considerations for Code Size and Speed” on the Silicon Labs 32-bit application notes website: www.silabs.com/32bit-appnotes.

4.4. Code Dependency

In addition to dynamic clock management, the power consumption of the SiM3L1xx device will vary with the type of code the core executes. For example, if the core executes a complex math routine with branches, the pipeline will miss every time a branch is taken and new instructions must be fetched. This stall and fetch period causes more flash accesses, which increases power consumption. In addition, the core executes a wide variety of instructions and activates the memory bus to fetch data from RAM or flash for use in these routines. In contrast, a string of NOP instructions will take less power because the core isn't executing complex instructions.

The data shown in Figure 2 does not use adaptive voltage scaling or any other techniques to change power consumption. All of these measurements were taken using the PLL as the clock source with a higher SPMD setting (i.e., reduced flash access frequency) at AHB frequencies above 40 MHz. The APB clock is equal to the AHB clock, when the APB is enabled.

As shown by the data, the flash access frequency (SPMD) has a direct effect on the code that includes branches, since the core stalls when waiting for the new instructions, resulting in reduced power consumption. This means that it may be more efficient to run at a faster frequency with the same current consumption to reduce overall time spent in active mode. For the code that is a long string of NOPs, the core never has a pipeline miss and never stalls, so there is no change in power consumption with a different flash speed mode.

For power sensitive applications, experimenting with various code styles and instruction mixes may result in reduced power consumption in active mode.

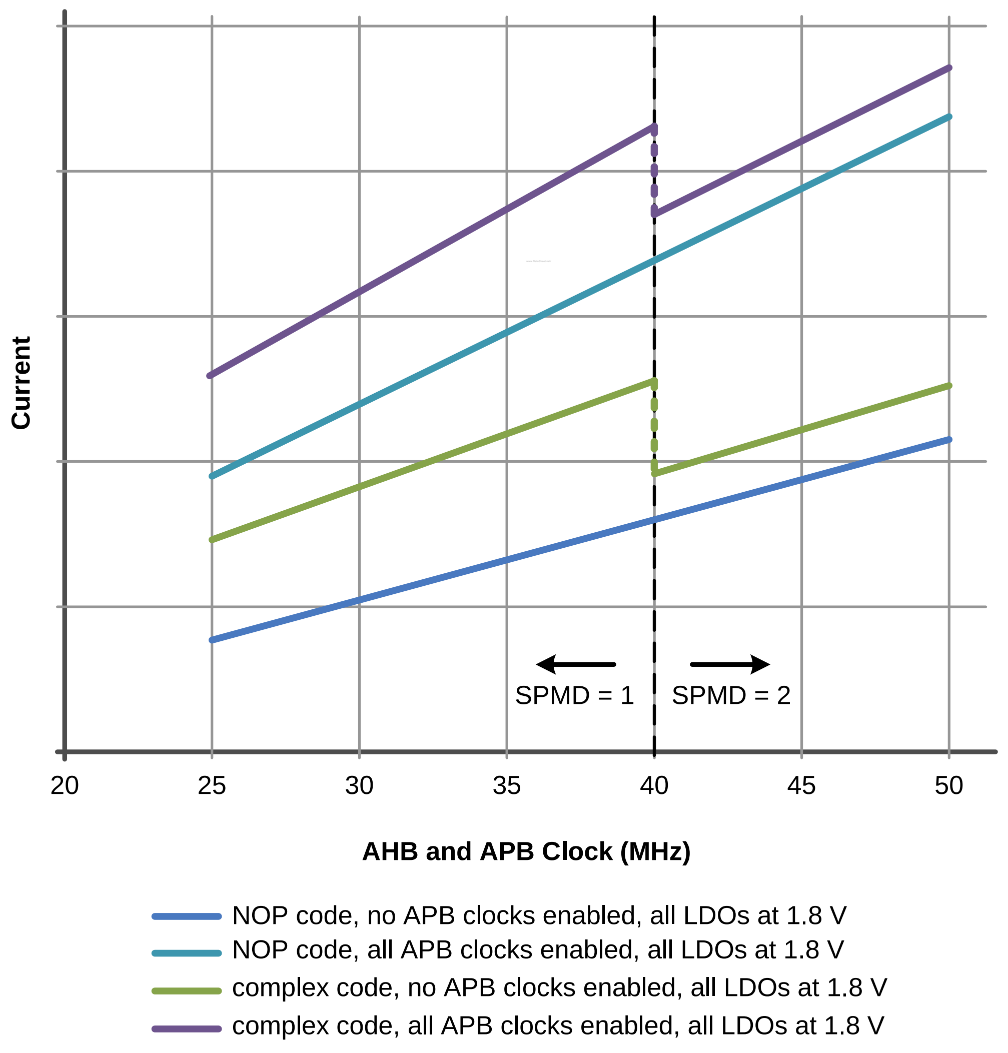


Figure 2. Power Consumption Code Dependency

4.5. Adaptive Voltage Scaling

SiM3L1xx devices have scalable LDOs powering the digital and memory modules on the device. These LDO outputs are factory calibrated to 1.8 V to handle all process and temperature variations, but this voltage is often much higher than the minimum voltage required by the digital and memory circuits. During normal operation, these LDO outputs can be set to a value less than 1.8 V to reduce the amount of excess power consumed by these circuits with some buffer to ensure correct operation.

When the digital or memory LDOs are sourced from the dc-dc converter, adaptive voltage scaling can also allow the LDOs to track the dc-dc output voltage to utilize the higher efficiency of the dc-dc converter and reduce the losses in the regulators.

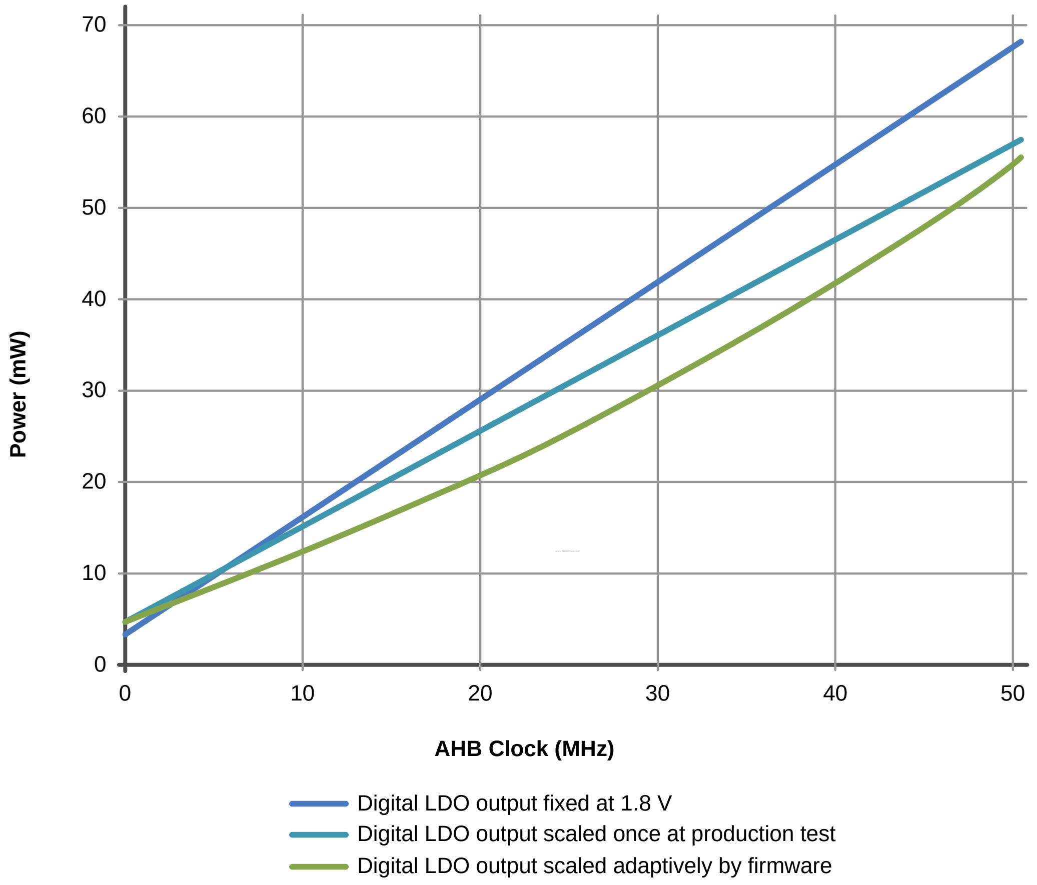


Figure 3. Reducing Power Consumption with Adaptive Voltage Scaling

4.6. DC-DC Load and Power Efficiency

The DCDC0 module on SiM3L1xx devices is a dc-dc buck converter with an input range of 1.8 to 3.8 V and an output range of 1.25 to 3.8 V. The efficiency of this regulator changes based on load and the converter configuration, as shown in Figure 4. The converter configuration can be changed dynamically or bypassed according to the anticipated demands of the application to maintain the highest power efficiency and reduce power consumption.

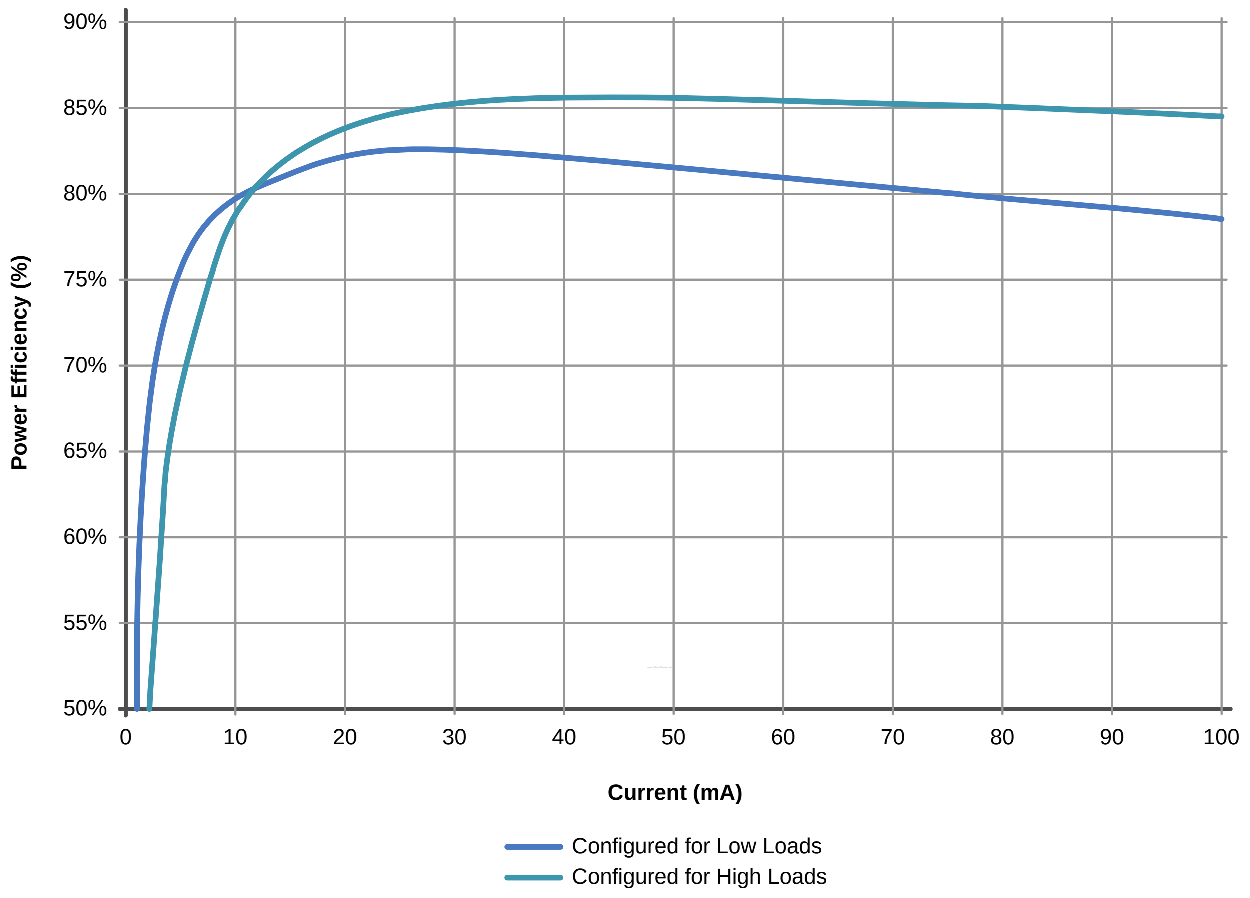


Figure 4. Reducing Power Consumption with the DC-DC Buck Converter

There are three ranges of operation for the dc-dc converter corresponding to three different load sizes:

1. Loads less than 5 mA
2. Loads between 5 and 15 mA
3. Loads greater than 15 mA

When operating with loads less than 5 mA, the dc-dc converter is most efficient when configured for light loads:

1. Power switch size set to 0 (PSMD = 0)
2. Asynchronous mode enabled (ASYNCEN = 1)
3. Minimum pulse width set to 40 ns (MINPWSEL = 3)

For loads between 5 and 15 mA, the converter should be set in a middle configuration:

1. Power switch size set to 0 (PSMD = 0)
2. Synchronous mode enabled (ASYNCEN = 0)
3. Minimum pulse width disabled (MINPWSEL = 0)

To configure the converter for loads greater than 15 mA:

1. Power switch size set to 3 (PSMD = 3)
2. Synchronous mode enabled (ASYNCEN = 0)
3. Minimum pulse width disabled (MINPWSEL = 0)

Table 1 shows a summary of these settings for each load configuration.

Table 1. DC-DC Load Configurations

Load	Power Switch Size (PSMD)	Synchronous or Asynchronous (ASYNCEN)	Minimum Pulse Width (MINPWSEL)
less than 5 mA	0	Asynchronous (1)	40 ns (3)
5 mA to 15 mA	0	Synchronous (0)	disabled (0)
greater than 15 mA	3	Synchronous (0)	disabled (0)

After configuring the dc-dc converter for the load size, switch the appropriate LDOs to the converter output to reduce system power. Each of the three LDOs (memory, digital, and analog) can be switched to the battery voltage (VBAT) or the dc-dc converter output independently.

The firmware can adjust the dc-dc converter configuration based on the anticipated load for the active time. For example, if only a few peripherals will be active and the core will halt for a period of time, then the load may be less than 5 mA. Firmware can adjust the dc-dc converter appropriately during this period. If the core is fetching instructions from flash at full speed and executing a math routine, then switching to the high load configuration will yield lower power consumption.

For extremely light loads (less than 2-3 mA), the dc-dc converter will be less efficient than the LDOs at ~50% efficiency, depending on the VBAT voltage and LDO bias settings. When this occurs, the dc-dc converter should be bypassed (BEN = 1), which connects VBATDC to VDC, or disabled (DCDCEN = 0).

5. Reducing Power Consumption in Low Power Modes

In low power mode, the device core halts and the number of active peripherals typically reduces to the minimum required by the application (i.e., real time clock). This section discusses techniques to reduce the SiM3L1xx device power consumption in low power modes (PM2, PM3, PM6, and PM8).

5.1. SiM3L1xx Low Power Mode Overview

The SiM3L1xx devices feature seven low power modes in addition to normal operating mode. Several peripherals provide wake up sources for these low power modes, including the Low Power Timer (LPTIMER0), RTC0 (alarms and oscillator failure notification), Comparator 0 (CMP0), Advanced Capture Counter (ACCTR0), LCD VBAT monitor (LCD0), UART0, low power mode charge pump failure, and PMU Pin Wake.

In addition, all peripherals can have their clocks disabled to reduce power consumption with the clock control (CLKCTRL) registers whenever a peripheral is not being used.

The SiM3L1xx devices have the power modes defined in Table 2.

5.1.1. Normal Mode (Power Mode 0)

Normal mode encompasses the typical full-speed operation. The power consumption of the device in this mode will vary depending on AHB/APB clock speeds, the settings of CLKCTRL and the peripherals, and the dc-dc converter and LDO settings.

5.1.2. Power Mode 1

Power Mode 1 occurs when the core executes code from RAM instead of flash. The power consumption of the device is less than normal mode when in PM1.

5.1.3. Power Mode 2

In Power Mode 2, the core halts and the peripherals run at full speed. To place the device in this mode, the clock settings in CLKCTRL should remain the same as Normal or Power Mode 1 and the core should execute a WFI or WFE instruction. If the WFI instruction is called from an interrupt service routine, the interrupt that wakes the device from PM2 must be of a sufficient priority to be recognized by the core. It is recommended to perform both a DSB (Data Synchronization Barrier) and an ISB (Instruction Synchronization Barrier) operation prior to the WFI to ensure all bus access is complete.

5.1.4. Power Mode 3 Fast Wake

Power Mode 3 Fast Wake occurs when all the clocks are stopped except for the LFOSC0 or RTC0TCLK. The core and the peripherals are halted in this mode. The available wake up sources to wake from PM3 are controlled by the Power Management Unit (PMU). The available wake up sources are: Low Power Timer (LPTIMER0), RTC0 (alarms and oscillator failure notification), Comparator 0 (CMP0), advanced capture counter (ACCTR0), LCD VBAT monitor (LCD0), UART0, and PMU Pin Wake. Any reset event will also wake the device from PM3.

If the WFI instruction that wakes the device from PM3 Fast Wake is called from an interrupt service routine, the interrupt that wakes the device from PM3FW must be of a sufficient priority to be recognized by the core.

By keeping the core clock running at a slow frequency in PM3 and changing the AHB and APB clocks to the Low Power Oscillator, the device can wake up faster than in standard Power Mode 3 at the expense of higher power consumption.

5.1.5. Power Mode 3

Power Mode 3 occurs when all the clocks are stopped, and the core and the peripherals are halted. Waking from PM3 requires one of the PMU wake sources described in “5.1.4. Power Mode 3 Fast Wake” to be properly configured.

If the WFI instruction is called from an interrupt service routine, the interrupt that wakes the device from PM3 must be of a sufficient priority to be recognized by the core.

5.1.6. Power Mode 4

Power Mode 4 is the same as normal operation except the AHB clock operates at a slower speed. The power consumption of the device in this mode will vary depending on the AHB/APB clock speeds, the settings of CLKCTRL and the peripherals, and the dc-dc converter and LDO settings.

5.1.7. Power Mode 5

Power Mode 5 is the same as PM1 with a slower AHB clock source selected. The core executes code from RAM instead of flash. The power consumption of the device in PM5 is slightly less than PM4.

5.1.8. Power Mode 6

In Power Mode 6, the core halts and the peripherals run at a slower speed than PM2. To place the device in this mode, the clock settings in CLKCTRL should remain the same as PM4 or PM5, and the core should execute a WFI or WFE instruction. If the WFI instruction is called from an interrupt service routine, the interrupt that wakes the device from PM6 must be of a sufficient priority to be recognized by the core.

5.1.9. Power Mode 8

In Power Mode 8, the core and most peripherals are halted, most clocks are stopped, and registers retain their state. In addition, the LDO regulators are disabled, so all active circuitry operates directly from VBAT. Alternatively, the PMU has a specialized VBAT-divided-by-2 low power mode charge pump that can power some internal modules while in PM8 to save power. The fully operational functions in this mode are: LPTIMER0, RTC0, UART0 running from RTC0TCLK, port match, advanced capture counter, and the LCD controller.

This mode provides the lowest power consumption for the device, but requires an appropriate wake up source or reset to exit. The available wake up or reset sources to wake from PM8 are controlled by the Power Management Unit (PMU). The available wake up sources are: Low Power Timer (LPTIMER0), RTC0 (alarms and oscillator failure notification), Comparator 0 (CMP0), advanced capture counter (ACCTR0), LCD VBAT monitor (LCD0), UART0, low power mode charge pump failure, and PMU Pin Wake. The available reset sources are: $\overline{\text{RESET}}$ pin, VBAT supply monitor, Comparator 0, Comparator 1, low power mode charge pump failure, RTC0 oscillator failure, or PMU wake event.

To enter this mode, firmware must write the SLEEPDEEP bit in the ARM System Control Register. Firmware must then execute a WFI or WFE instruction. The core will remain in PM8 until an enabled wake up or reset source occurs.

Table 2. SiM3L1xx Power Modes

Mode	Description	Mode Entrance	Mode Exit
Normal	■ Core operating at full speed ■ Code executing from flash		
Power Mode 1 (PM1)	■ Core operating at full speed ■ Code executing from RAM	Execute code from RAM	Jump to code in flash
Power Mode 2 (PM2)	■ Core halted ■ AHB and APB operate at full speed for peripherals	WFI or WFE instruction	NVIC or WIC wakeup
Power Mode 3 Fast Wake (PM3FW)	■ All clocks stopped except LFOSC0 or RTC0TCLK ■ AHB and APB set to Low Power Oscillator ■ Core clock set to LFOSC0 or RTC0TCLK	■ DMACTRL0 disabled ■ Fast wake mode enabled in PM3CN ■ AHB switched to Low Power Oscillator ■ WFI or WFE instruction	Requires a wake up source or reset defined by the PMU
Power Mode 3 (PM3)	All clocks stopped	■ DMACTRL0 disabled ■ Clocks disabled in PM3CN ■ WFI or WFE instruction	Requires a wake up source or reset defined by the PMU
Power Mode 4 (PM4)	■ Core operating at slower speed ■ Code executing from flash	■ Set the AHB clock to a slower source	Set the AHB clock to a faster source
Power Mode 5 (PM5)	■ Core operating at slower speed ■ Code executing from RAM	■ Set the AHB clock to a slower source ■ Execute code from RAM	Set the AHB clock to a faster source or jump to code in flash
Power Mode 6 (PM6)	■ Core halted ■ AHB and APB operate at a slower speed for peripherals	■ Set the AHB clock to a slower source ■ WFI or WFE instruction	NVIC or WIC wakeup
Power Mode 8 (PM8)	■ Low power sleep ■ The LDO regulators are disabled and all active circuitry operates directly from VBAT ■ The following functions are available: ACCTR0, RTC0, UART0 running from RTC0TCLK, LPTIMER0, port match, and the LCD controller ■ Register state retention	■ SLEEPDEEP set in the ARM System Control Register ■ WFI or WFE instruction	Requires a wake up source or reset defined by the PMU

5.2. Measuring Power

This section discusses the hardware setup and code required to reproduce the power specifications reported by the SiM3L1xx data sheet.

5.2.1. Hardware Setup

To measure the power on the SiM3L1xx MCU Card with a fixed 3.3 V VBAT:

1. Connect a USB Debug Adapter to the 10-pin CoreSight connector (J14).
2. Remove the three shorting blocks from J7.
3. Remove the JP2 **Imeasure** shorting block and put a multimeter across (positive side on the bottom pin).
4. Move the **VBAT Sel** switch (SW2) to the middle +3.3 V_VREG position.
5. Move the **VIO Sel** (SW8) and **VIORF Sel** (SW9) switches to the bottom VBAT position.
6. Connect the 9 V power adapter to POWER (J6).
7. Download the code to the board.
8. Remove the debug adapter connection.
9. Measure the power of the device.

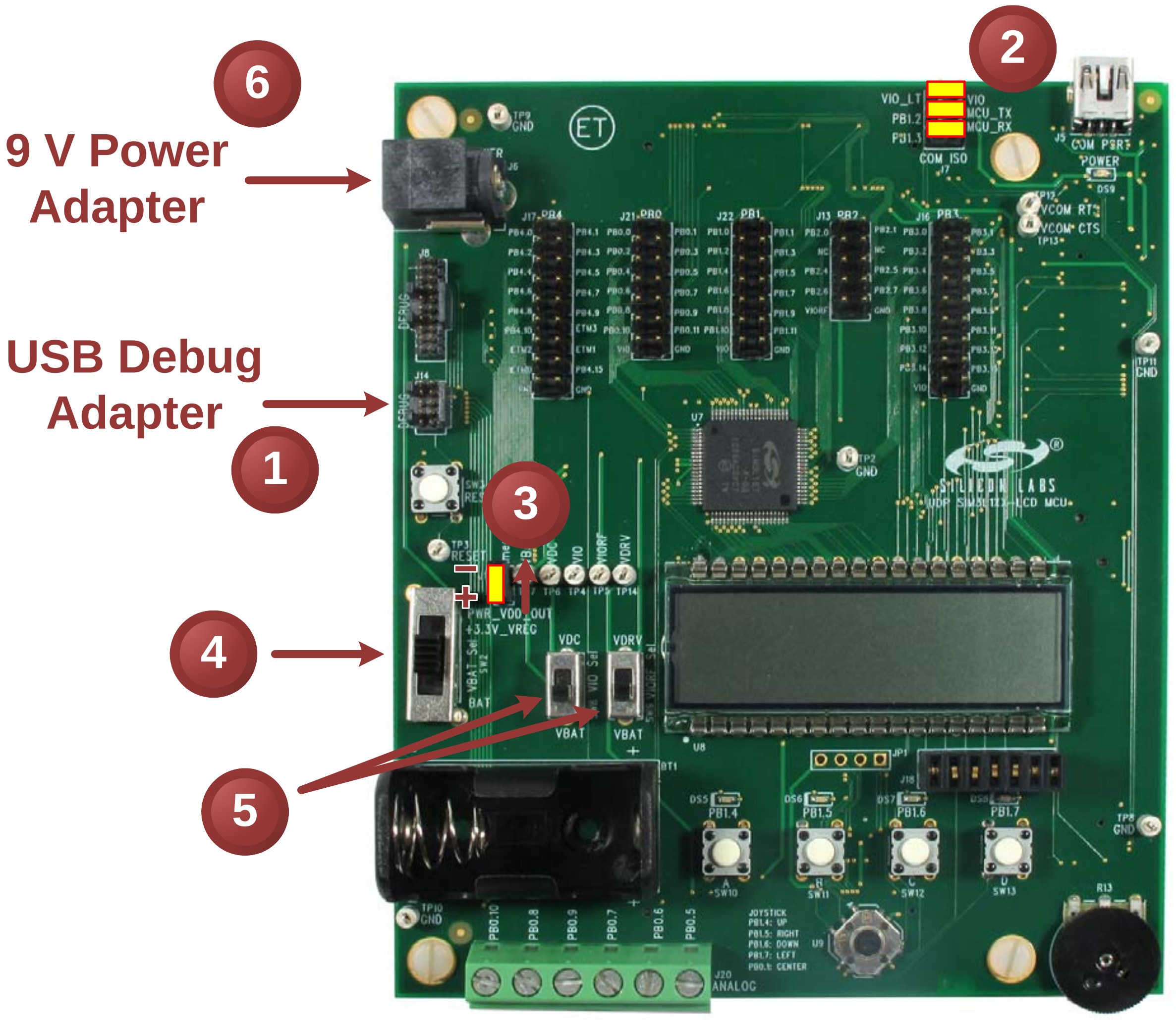


Figure 5. SiM3L1xx MCU Card Power Measurement Configuration—Fixed VBAT

To measure the power on an SiM3L1xx MCU Card with a varying VBAT:

1. Connect a USB Debug Adapter to the 10-pin CoreSight connector (J14).
2. Remove the three shorting blocks from J7.
3. Remove the JP2 **Imeasure** shorting block.
4. Move the **VBAT Sel** switch (SW2) to the bottom BAT position.
5. Move the **VIO Sel** (SW8) and **VIORF Sel** (SW9) switches to the bottom VBAT position.
6. (Optional) Place a battery in the BT1 holder.
7. (Optional) Instead of a battery, connect the positive terminal of a bench power supply to the positive terminal of the multimeter, the negative terminal of the power supply to the MCU card ground, and the negative terminal of the multimeter to the top pin of the Imeasure jumper (JP2). This configuration will prevent any circuitry on the board from interfering with the power measurement.
8. Download the code to the board.
9. Remove the debug adapter connection.
10. Measure the power of the device.

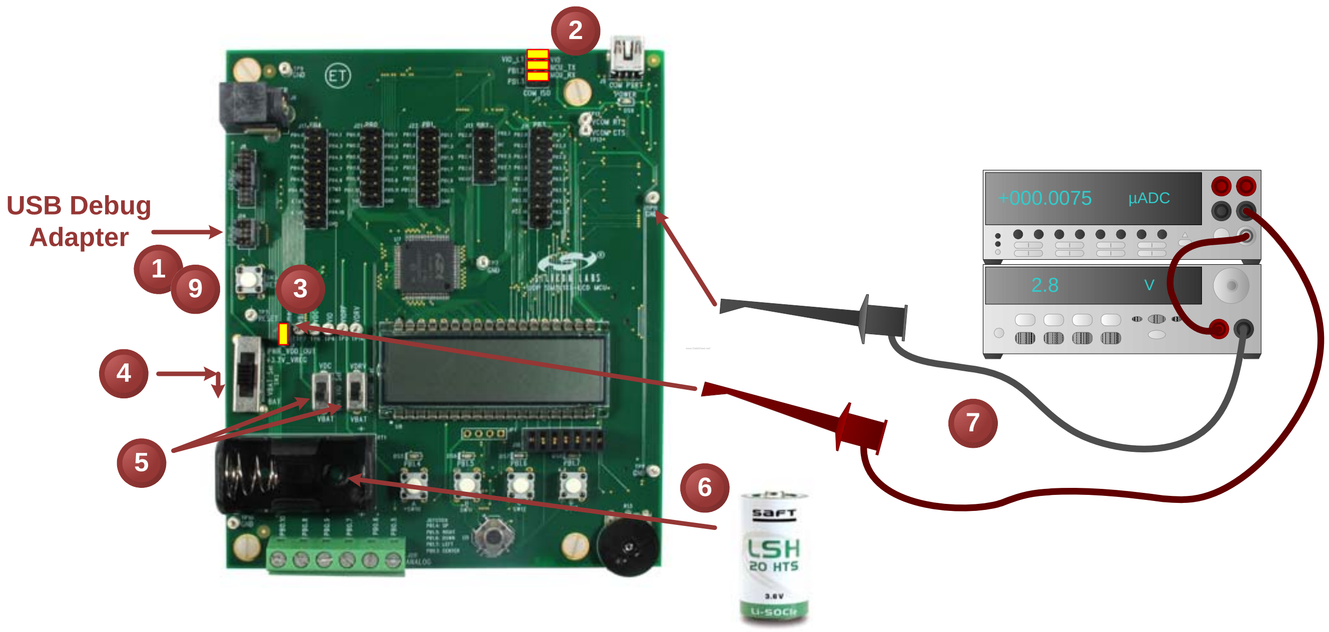


Figure 6. SiM3L1xx MCU Card Power Measurement Configuration—Varying VBAT

5.2.2. Configuring a Device for Normal (PM0) and Power Mode 4

In Normal (PM0) and Power Mode 4, the device executes code from flash. The only difference between PM0 and PM4 is the core clock speed (fast and slow, respectively).

To configure a device to run in PM0 or PM4:

1. Enable the clocks to peripherals that will be configured by firmware.
2. Select the desired clock source (LPOSC0, PLL0, etc.) and speed for both AHB and APB clocks.
3. Select the desired adaptive voltage scaling settings using the LDO module.
4. (Optional) Disable the retention mode of any enabled RAM banks.
5. Set the pins in the lowest power configuration for this mode.
6. Disable all unused peripherals.
7. Disable the clocks to all unused peripherals.
8. Jump to code in flash.

To measure the data sheet numbers using an SiM3L1xx MCU Card and the AN725_PowerModes_0_and_4 example:

1. Configure the SiM3L1xx MCU Card according to the instructions in “5.2.1. Hardware Setup”.
2. Open the AN725_PowerModes_0_and_4 example in either Keil µVision or the Precision32 IDE.
3. Select the desired settings using the #defines at the top of the file. There is a set of defines for each data sheet specification.
4. Compile and download the code to the device.
5. Disconnect the USB Debug Adapter.
6. Reset the device.
7. Measure the power consumption of the device.

5.2.3. Configuring a Device for Power Modes 1 and 5

In Power Modes 1 and 5, the device executes code from RAM rather than flash. The only difference between PM1 and PM5 is the core clock speed (fast and slow, respectively).

Keil µVision uses a scatterfile with a retention RAM tag defined as MCU_RAM_CODE:

```
MCU_IRAM +0  {
   .ANY (+RW +ZI)
   .ANY (MCU_RAM_CODE)
}
```

The project source file can then define a tag in code to place functions in this section of memory:

```
#define __SI32_RRAM __attribute__ ((section("MCU_RAM_CODE")))
```

Any functions that use the __SI32_RRAM tag in the definition will be placed in RAM starting at 0x2000_0000:

```
__SI32_RRAM void run_PM1(void)
{
...
}
```

For Precision32 projects, a function can be located in RAM by treating it like normal data and applying a data section attribute tag to it.

```
__attribute__ ((__section__(".data.name")))
```

This tag will result in a warning indicating that code is being placed in the data section, but this warning can be ignored. As an example, placing the PowerModes_2_or_6() function in RAM would look like:

```
__attribute__ ((__section__(".data.pm2_or_6")))

void PowerModes_2_or_6(void)

{

...

}
```

There are two ways to access the function in RAM: direct call or function pointer. Direct calls can cause some issues with debugging the code in RAM, and the debugger may just step over the function instead of stepping into it. To work around this, single step at the instruction level, set a breakpoint in the RAM code, or use a function pointer.

To configure a device to run in PM1 or PM5:

1. Enable the clocks to peripherals that will be configured by firmware.
2. Select the desired clock source (LPOSC0, PLL0, etc.) and speed for both AHB and APB clocks.
3. Select the desired adaptive voltage scaling settings using the LDO module.
4. (Optional) Disable the retention mode of any enabled RAM banks.
5. Set the pins in the lowest power configuration for this mode.
6. Disable all unused peripherals.
7. Disable the clocks to all unused peripherals.
8. Jump to code in RAM.
9. (Optional) Disable the AHB clock to the flash controller.

To measure the data sheet numbers using an SiM3L1xx MCU Card and the AN725_PowerModes_1_and_5 example:

1. Configure the SiM3L1xx MCU Card according to the instructions in “5.2.1. Hardware Setup”.
2. Open the AN725_PowerModes_1_and_5 example in either Keil µVision or the Precision32 IDE.
3. Select the desired settings using the #defines at the top of the file. There is a set of defines for each data sheet specification.
4. Compile and download the code to the device.
5. Disconnect the USB Debug Adapter.
6. Reset the device.
7. Measure the power consumption of the device.

The code executes from RAM in this mode, so all functions called should also be in RAM.

5.2.4. Configuring a Device for Power Modes 2 and 6

For Power Modes 2 and 6, the core halts and no longer executes code, but the active peripherals still run from the APB clock. The only difference between PM2 and PM6 is the APB clock speed (fast and slow, respectively).

To configure a device to run in PM2 or PM6:

1. Enable the clocks to peripherals that will be configured by firmware.
2. Select the desired clock source (LPOSC0, PLL0, etc.) and speed for both AHB and APB clocks.
3. Select the desired adaptive voltage scaling settings using the LDO module.
4. Disable Power Mode 8 in the Clock Control module.
5. (Optional) Disable the retention mode of any enabled RAM banks.
6. Set the pins in the lowest power configuration for this mode.
7. Set up the wakeup interrupt source (including priority).
8. Disable the SysTick timer to prevent these interrupts from waking the core.
9. Disable all unused peripherals.
10. Disable the clocks to all unused peripherals.
11. Jump to code in RAM.
12. Execute the DSB (Data Synchronization Barrier), ISB (Instruction Synchronization Barrier), and WFI (Wait for Interrupt) or WFE (Wait for Event) instructions. For SiM3L1xx devices, WFI and WFE have the same behavior.

To measure the data sheet numbers using an SiM3L1xx MCU Card and the AN725_PowerModes_2_and_6 example:

1. Configure the SiM3L1xx MCU Card according to the instructions in “5.2.1. Hardware Setup”.
2. Open the AN725_PowerModes_2_and_6 example in either Keil µVision or the Precision32 IDE.
3. Select the desired AHB clock rate using the #defines at the top of the file.
4. Compile and download the code to the device.
5. Disconnect the USB Debug Adapter.
6. Reset the device.
7. Press PB1.7 to place the device in PM2 or PM6. The PB1.5 LED will turn off.
8. Measure the power consumption of the device.
9. Press PB1.5 to wake the device from PM2 or PM6. When this happens, the PB1.7 LED will turn on and the core will sit in an infinite while(1) loop.

The interrupt that wakes the core from the halted state must have sufficient priority if the WFI or WFE instruction was called from within an ISR.

5.2.5. Configuring a Device for Power Mode 3 Fast Wake

In PM3 Fast Wake, all clocks are stopped except for a very low frequency clock (RTC0OSC or LFOSC0) that allows the core to wake up faster than standard PM3.

To configure a device to run in PM3 Fast Wake:

1. Enable the clocks to peripherals that will be configured by firmware.
2. Select the desired adaptive voltage scaling settings using the LDO module.
3. Disable Power Mode 8 mode in the Clock Control module.
4. Set the AHB clock to the low power oscillator (LPOSC0).
5. Select the fast wake clock source using the PM3CSEL field in the CLKCTRL PM3CN register.
6. Enable fast wake mode by setting the PM3CEN bit to 1 in the PM3CN register.
7. (Optional) Disable the retention mode of any enabled RAM banks.
8. Set the pins in the lowest power configuration for this mode.
9. Set up the desired PMU wakeup source.
10. Put all of the LDOs in low bias mode.
11. Clear the PMU wakeup flags.
12. Disable the SysTick timer.
13. Disable all unused peripherals.
14. Disable the clocks to all unused peripherals.
15. Execute the DSB, ISB, and WFI or WFE instructions. For SiM3L1xx devices, WFI and WFE have the same behavior.

To measure the data sheet numbers using an SiM3L1xx MCU Card and the AN725_PowerMode3_Fast_Wake example:

1. Configure the SiM3L1xx MCU Card according to the instructions in “5.2.1. Hardware Setup”.
2. Open the AN725_PowerMode3_Fast_Wake example in either Keil µVision or the Precision32 IDE.
3. Select the desired AHB clock rate using the #defines at the top of the file.
4. Compile and download the code to the device.
5. Disconnect the USB Debug Adapter.
6. Connect PB0.2 to PB1.5 using a short wire and the J21 and J22 headers.
7. Reset the device.
8. Press PB1.7 to place the device in PM3FW. The PB1.5 LED will turn off.
9. Measure the power consumption of the device.
10. Press PB1.5 to wake the device from PM3FW. When this happens, the PB1.7 LED will turn on and the core will sit in an infinite while(1) loop.

5.2.6. Configuring a Device for Power Mode 8

PM8 is the lowest power mode where all clocks are stopped and only the PM8 peripherals are available. The RAM banks do not have retention enabled by default, so firmware must enable each bank to retain data through PM8. For most applications, it is recommended to enable retention for all RAM banks, but the specific bank containing the stack pointer must be enabled to ensure proper code operation on an exit from PM8. See “5.5. Retention RAM” for more information on changing the location of the stack pointer.

To configure a device to run in PM8:

1. Enable the clocks to peripherals that will be configured by firmware.
2. Enable PM8 mode by setting the PMSEL bit to 1 in the CLKCTRL CONFIG register.
3. Set up the desired PMU wakeup source.
4. Switch the AHB clock to the 20 MHz Low Power Oscillator and adjust the flash read timing as necessary.
5. Enable the retention mode of RAM banks needed by the application (all eight banks recommended for most applications).

Note: To properly execute code after exiting PM8, the stack pointer must be located in a RAM bank with retention enabled.

6. Set the pins in the lowest power configuration for this mode.
7. Set all LDOs to 1.5 V for VBAT between 1.8 to 2.9 V and to 1.9 V for VBAT between 2.0 to 3.8 V output.
8. (Optional) If VBAT is between 2.0 to 3.8 V, set the VBAT Monitor threshold to the high setting.
9. Set up the low power peripherals as desired (RTC0, ACCTR0, LCD0, etc.).
10. (Optional) If using the RTC in low-frequency oscillator mode, enable the RTC0 module using the steps in the Reference Manual. If using the RTC in crystal mode, the recommended initialization sequence is:
    a. Disable RTC0 automatic gain control (AGC) and enable the bias doubler.
    b. Set up the RTC0 in crystal mode using the list of steps in the Reference Manual.
    c. (Optional) Enable the charge pump (see Step 10).
    d. Enable RTC0 automatic gain control and disable the bias doubler.
11. (Optional) Configure and enable the charge pump:
    a. Enable the low power charge pump monitor in the PMU.
    b. Enable the RTC0 clock to other modules (CLKOEN = 1 in the RTC0 module).
    c. Enable the low power charge pump monitor as a reset source.
    d. Set the low power charge pump load to the appropriate value for the application. See “5.3. Low Power Mode Charge Pump” for more information.
    e. Enable the low power charge pump and enable it as a PMU wake up source.
    f. Enable the low power charge pump interrupt.
12. Disable all undesired reset sources.

Note: The Watchdog Timer (WDTIMER0) and PVTOSC0 modules will behave as if a power-on reset occurred after exiting PM8, so disabling the Watchdog Timer as a reset source will prevent unwanted resets.

13. Disable the SysTick timer.
14. Disable all unused peripherals.
15. (Optional) Disable the clocks to all unused peripherals. Entering PM8 stops all clocks, so this step is not necessary for this mode.
16. Clear the PMU wakeup flags.
17. Enable the interrupts for the PMU wakeup source (enabled in Step 3).
18. Set the SLEEPDEEP bit in the core.
19. Execute the DSB, ISB, and WFI or WFE instructions. For SiM3L1xx devices, WFI and WFE have the same behavior.

To measure the data sheet numbers using an SiM3L1xx MCU Card and the AN725_PowerMode_8 example:

1. Configure the SiM3L1xx MCU Card according to the instructions in Section “5.2.1. Hardware Setup”.
2. Open the AN725_PowerMode_8 example in either Keil µVision or the Precision32 IDE.
3. Select the desired AHB clock rate using the #defines at the top of the file.
4. Compile and download the code to the device.
5. Disconnect the USB Debug Adapter.
6. Connect PB0.2 to PB1.5 using a short wire and the J21 and J22 headers.
7. Power down the board.
8. Power up the board. This will cause a power-on reset that will clear the settings of some modules (i.e., PMU).
9. Press PB1.7 to place the device in PM8. The PB1.5 LED will turn off.
10. Measure the power consumption of the device.
11. Press PB1.5 to wake the device from PM8. When this happens, the PB1.7 LED will turn on and the core will sit in an infinite while(1) loop.

5.3. Low Power Mode Charge Pump

The SiM3L1xx devices feature a low-power, voltage-halving charge pump used to power most of the functions operating in the lowest power mode (PM8). This charge pump always powers the low-frequency and 32 kHz crystal oscillators. When in PM8, this charge pump also powers the following digital components: RTC counters and alarms, LPTIMER0, UART0, ACCTR0, retention RAM banks, LCD0, and the Power Management Unit (PMU) and reset controllers. When not in PM8, these modules are powered by the Memory regulator (LDO0). The analog circuitry of the LCD and Pulse Counter is always powered directly from the VBAT pin.

The low power mode charge pump is a voltage-halving circuit which reduces the sleep-mode supply current using two methods:

1. It provides approximately VBAT divided-by-2 to the oscillators and digital logic, reducing their inherent power consumption.
2. It provides a transformer effect which causes the current pulled from the VBAT input to be ideally one-half of the current actually consumed by the circuitry powered by the charge pump.

This charge pump can be modeled as a VBAT divided-by-2 supply with a series impedance of R_{CPLOAD}. Equation 1 shows the equation for the output voltage of the charge pump.

$$V_{CP} = \frac{VBAT}{2} - I_{LOAD} \times R_{CPLOAD}$$

Equation 1. Charge Pump Output Voltage

The value of R_{CPLOAD} depends on the output drive setting field (CPLOAD) in the PMU module and the clock selected by the RTC0 module, as shown in Table 3.

Table 3. Charge Pump Output Drive Settings

CPLOAD field	R_{CPLOAD} Value		Overhead Current	
	LFOSC (16.4 kHz)	RTCOSC (32 kHz)	LFOSC (16.4 kHz)	RTCOSC (32 kHz)
0	~600 kΩ	~300 kΩ	~0 nA	~0 nA
1	~200 kΩ	~100 kΩ	~25 nA	~50 nA
2	~90 kΩ	~45 kΩ	~50 nA	~100 nA
3	~40 kΩ	~20 kΩ	~75 nA	~150 nA

All circuitry powered by the charge pump operates correctly at V_{CP} voltages greater than or equal to 0.95 V. Thus, for a given charge pump load current, Equation 2 must be true.

$$\frac{VBAT}{2} - I_{LOAD} \times R_{CPLOAD} \geq 0.95\ V$$

Equation 2. Charge Pump Output Voltage Requirement

In all applications with a VBAT voltage of 2.4 V or above, the charge pump can operate at its minimum drive impedance (CPLOAD = 3). However, as shown in Table 3, the overhead current consumed by the charge pump increases as its drive impedance decreases. Therefore, at lower values of I_{LOAD}, the total PM8 current consumption can be reduced by increasing the charge pump's drive impedance. In addition, since a higher drive impedance corresponds to a lower output voltage for a given load current, increasing the drive impedance also reduces the current consumed by the load circuitry.

At higher temperatures, using the low power charge pump and RTC0 can help reduce the leakage current of the device. Reducing CPLOAD can further reduce the leakage at these temperatures.

5.3.1. Monitor

The SiM3L1xx devices include a low-power voltage monitor that compares the output of the low power mode charge pump against a factory-trimmed 0.95 V reference voltage. This monitor can serve as an interrupt, a PM8 wakeup source, and a reset source. It is highly recommended that this monitor be enabled and used as a wakeup or interrupt source. However, if Equation 2 will always be true under all valid operating conditions, the monitor is not strictly required. Leaving the monitor disabled introduces the risk that the device could lock-up, requiring an external reset or power cycle to recover. This can occur if the selected RTC0 oscillator is disturbed and stops oscillating, thereby causing the charge pump output to fail. Since the charge pump powers the RTC0 oscillators, the RTC fail notification is not sufficient to detect this condition, and it will not correct itself without external intervention. The charge pump voltage monitor consumes approximately 15 nA, when enabled.

When used as a wakeup and interrupt source, the charge pump monitor can alert the firmware that the charge pump output is nearing 0.95 V. This may occur as temperature rises, for example, causing additional thermal leakage from the PM8 circuitry. The firmware can respond by decreasing the charge pump drive impedance by increasing CPLOAD, assuming the drive is not yet at its minimum. Otherwise, firmware must either reduce the PM8 load (by disabling RAM banks or other PM8 active modules) or place the charge pump in bypass mode. When changing the charge pump mode, it is recommended that all digital logic using the RTC0 oscillators be temporarily disabled. Otherwise, a glitch from the oscillators due to the sudden change in the charge pump output may disturb the logic’s operation.

5.3.2. Measuring PM8 Charge Pump Current

To measure the charge pump load current, use CMP0 to detect the output voltage of the charge pump during PM8. This output voltage, combined with an ADC measurement of the VBAT value, can help estimate the present charge pump load in PM8. To do this:

1. Enable the charge pump with CPLOAD set to 3.
2. Configure the charge pump monitor to wake up and interrupt the device.
3. Configure CMP0 to wake and interrupt the device if the charge pump output voltage is equal to 16 x VBAT/64 (0x10) using the comparator DAC feature.
4. Schedule an RTC0 alarm to wake and interrupt the device in 1 ms.
5. Enter PM8 and wait for a wakeup event.
6. If the wakeup event occurred from the voltage monitor, this indicates the present VBAT voltage is too low to operate the charge pump. The charge pump should be configured for bypass mode and this measurement aborted.
7. If the wakeup event was the comparator, change the CMP0 to compare against 8 x VBAT/64 (0x08).
8. If the wakeup event was the RTC0 alarm, change the CMP0 to compare against 24 x VBAT/64 (0x18).
9. Repeat this process starting at step 4, continuing to change the CMP0 DAC using a successive-approximation method to obtain a 5-bit digital value for the charge pump output voltage, where 0x1F corresponds to 31 x VBAT/64.
10. Use the SARADC0 module to measure the voltage on the VBAT input.
11. Calculate the estimated charge pump load current using Equation 1, the VBAT and final CMP0 DAC value, and the appropriate value for R_{CPLOAD}.

If the resulting load current is small enough that the next lowest setting of CPLOAD results in a charge pump output greater than 0.95 V, this sequence can be repeated with CPLOAD set to 2 to obtain a more accurate measurement of the load current, and so forth.

5.3.3. Dynamic Charge Pump Drive Impedance

The low power mode charge pump supports four drive impedance settings. For applications requiring aggressive PM8 current reduction, these settings can be adjusted dynamically to minimize the total low power mode current consumed. This adjustment could be performed by measuring the charge pump load current as described in Section 5.3.2 before each entry into PM8 and setting the appropriate drive impedance. However, a faster and simpler method is available.

The charge pump dynamic scaling method consists of two parts:

1. Determine when the drive impedance should be decreased by using the charge pump voltage monitor. Each time the monitor trips, firmware should reduce the drive impedance setting. If the setting is already at its minimum, firmware can place the charge pump in bypass mode or disable active PM8 circuitry.
2. Determine when the drive impedance can be increased. On a regular but relatively infrequent basis, firmware can determine if the CPLOAD setting can be decreased by one without causing the charge pump output to fall below 0.95 V as described in this section.

The output state of the charge pump in PM8 is given by Equation 1. The required state of the charge pump if CPLOAD is reduced by 1 is given by:

$$\frac{VBAT}{2} - I_{LOAD} \times R_{CPLOAD-1} \geq 0.95\ V$$

Equation 3. Charge Pump Output Voltage at an Increased Impedance

In Equation 3, the $R_{CPLOAD-1}$ term is the drive impedance for CPLOAD-1. Putting these equations together and solving for I_{LOAD} gives:

$$V_{CP} \geq \frac{VBAT}{2}\left(1 - \frac{R_{CPLOAD}}{R_{CPLOAD-1}}\right) + 0.95\ V\left(\frac{R_{CPLOAD}}{R_{CPLOAD-1}}\right)$$

Equation 4. Minimum Charge Pump Output Voltage to Increase Drive Impedance

Equation 4 provides the minimum V_{CP} value for the present CPLOAD setting that allows CPLOAD to be reduced by 1 without the output dropping below 0.95 V. Using this equation, the CMP0 DAC setting that corresponds to this equation can be written as:

$$DAC = \text{ceiling}\left(32\left(1 - \frac{R_{CPLOAD}}{R_{CPLOAD-1}}\right) + \frac{64 \times 0.95\ V}{VBAT}\left(\frac{R_{CPLOAD}}{R_{CPLOAD-1}}\right)\right)$$

Equation 5. CMP0 DAC Setting for Minimum Charge Pump Output Voltage to Increase Drive Impedance

On a regular basis, firmware can perform the following steps:

1. Use the SARADC0 module to measure VBAT.
2. Set the CMP0 DAC to the value determined by Equation 5.
3. Configure CMP0 to wake and interrupt the device if the charge pump output voltage is less than the CMP0 DAC.
4. Schedule an RTC0 alarm to wake and interrupt the device in 1 ms.
5. Enter PM8 and wait for a wakeup event.
6. If the wakeup event was the comparator, CPLOAD should not be reduced by 1. Otherwise, CPLOAD can be reduced by 1.

Equation 4 includes some safety margins that might not be immediately obvious, since it assumes that the charge pump load current is independent of the charge pump output voltage. In fact, the PM8 circuitry requires less current as its supply lowers. Since reducing CPLOAD by 1 will decrease the charge pump output voltage, the resulting output voltage should always be greater than 0.95 V, assuming Equation 4 is satisfied.

5.4. LCD

For applications that use the LCD0 module, the SiM3L1xx devices have several features that help reduce power consumption.

5.4.1. Segment Resetting

Each segment of an LCD can be modeled as a capacitor in the order of tens of pF. A typical load current is 1 nA per segment per pF, so a 160-segment LCD with 50 pF per segment can draw 8 µA of load current. The SiM3L1xx devices include a segment resetting feature to reduce this load current. Figure 7 illustrates the basic architecture model of an LCD.

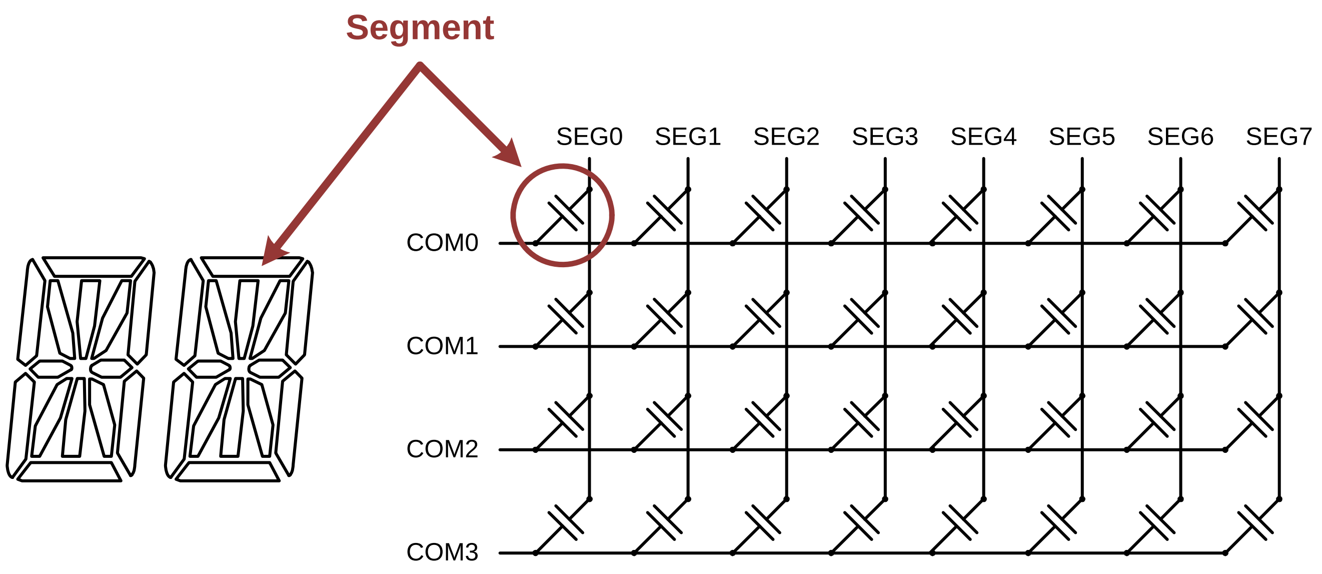


Figure 7. LCD Model

LCD segments are energized (i.e., turn opaque) by ac potential waveforms between the segment and the common signal for that segment. These waveforms transition between 0 and 3 V, for example, which results in an amplitude of –3 to 3 V on the segment, depending on the phase of the segment and common waveforms. Figure 9 shows an example of the common waveforms for 4-mux mode.

To turn on a segment, the controller drives the segment to 3 V or 0 V when the common is driven to 0 V or 3 V, which leads to ±3 V on the segment. To turn a segment off, the controller drives the segment to 1 V or 2 V when the common signal is 0 V or 3 V, which leads to ±1 V on the segment.

Most LCD controllers transition directly from the most positive amplitude (3 V) to the most negative amplitude (–3 V), as shown in Figure 8. This behavior leads to extra charge being pulled from the battery, since the battery must drive a 6 V transition overall.

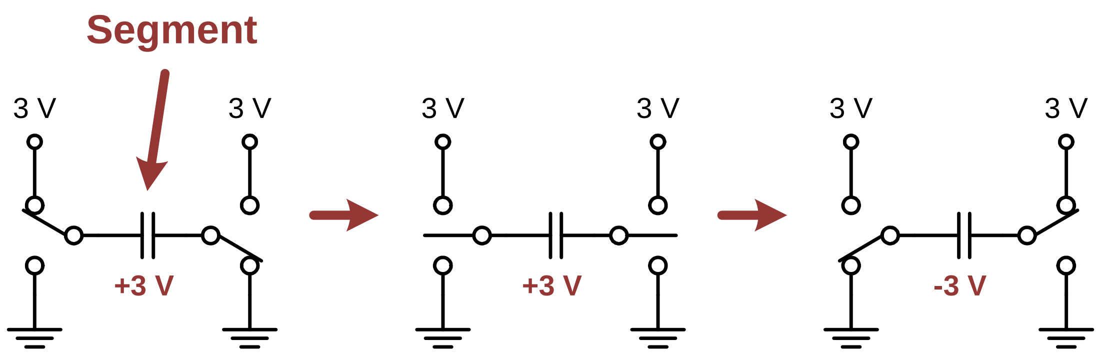


Figure 8. Traditional LCD Segment Transition

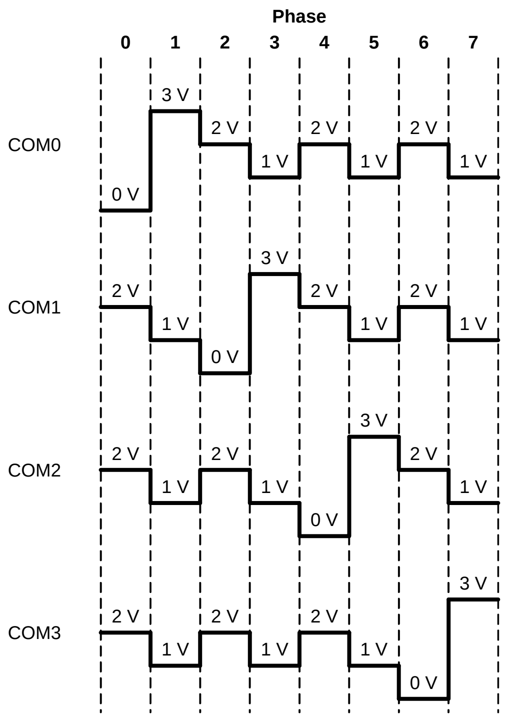


Figure 9. LCD Common Waveforms

For a 4 MUX LCD architecture, Figure 9 demonstrates that three of the four commons are at the same potential in every phase. This potential is 2 V for the even phases and 1 V for the odd phases. Instead of the traditional segment switching method, the SiM3L1xx devices reset the segments to 2 V for even phases and 1 V for odd phases before switching to the next segment potential to reduce the overall load current of the LCD.

The waveforms shown in Figure 10 show the SiM3L1xx segment resetting behavior in the common and segment waveforms. The areas marked in red are the reset events to the 1 V or 2 V potential, depending on the phase.

Overall, this reset waveform scheme reduces the current load of an LCD by ~40%, regardless of the number of segments in the display. Power-conscious applications using an LCD should enable this feature (RPHEN = 1) and set the number of RTC0 clocks to reset the segment (RPHMD field) using the LCD0 SEGCONTROL register. Generally, a RPHMD value of 2 provides significant power savings without introducing visual artifacts. However, it is recommended to try a range of values to find the best value for a given LCD.

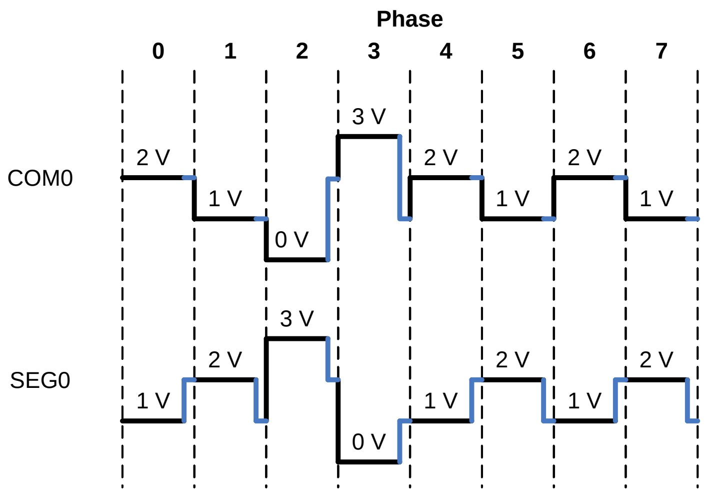


Figure 10. SiM3L1xx LCD Waveforms with Segment Resetting

5.4.2. Contrast Modes

The LCD0 module features four contrast modes: bypass, minimum, constant, and auto-bypass. The selection of the contrast mode is governed by the application requirements and LCD specifications. The power consumption of these modes will vary due to the internal charge pump and resistor string that generates the VLCD voltage. Reducing the time this hardware is active will reduce power consumption.

In bypass mode shown in Figure 11, the VLCD voltage tracks with the VBAT input voltage. This mode reduces power consumption by disabling the LCD charge pump and is most appropriate for systems where the VBAT voltage is relatively constant.

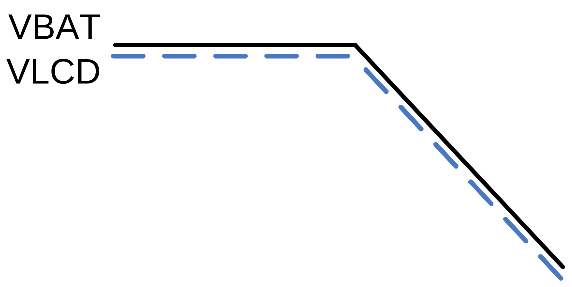


Figure 11. LCD0 Bypass Contrast Mode

In minimum contrast mode shown in Figure 12, the VLCD voltage will track the VBAT voltage to a minimum level set by the VBAT monitor threshold (VBMTH). When VBAT is equal to this voltage, VLCD will switch to the contrast level set by the CTRST field. The threshold and contrast voltages do not have to be set to the same level, as shown in Figure 13. If the VBAT voltage goes back above the threshold, the LCD contrast charge pump will turn off. This mode is most efficient for systems that can use the battery voltage most of the time but require a minimum contrast level to see the LCD segments.

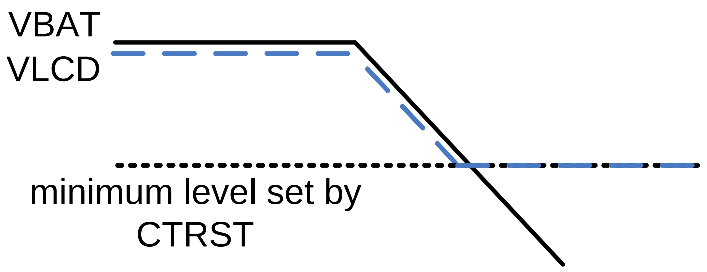


Figure 12. LCD0 Minimum Contrast Mode

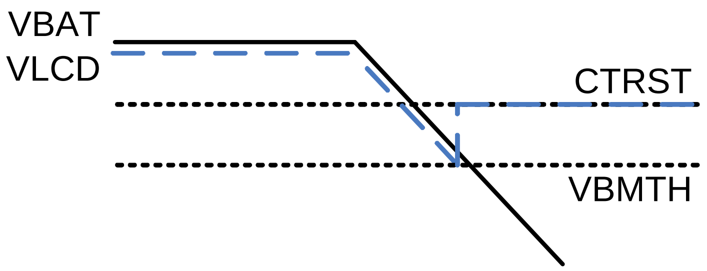


Figure 13. LCD0 Minimum Contrast Mode with Different Threshold and Voltage Settings

In constant contrast mode, the VLCD voltage is held constant at all times, regardless of the voltage on VBAT. When VBAT is above the threshold set by the VBMTH field, the LCD charge pump regulates to the contrast level set by the CTRST field using a variable resistor. When VBAT reaches the threshold, the charge pump enables automatically to keep VLCD at the desired value. This mode results in the highest power consumption, but may be required by some system and LCD architectures.

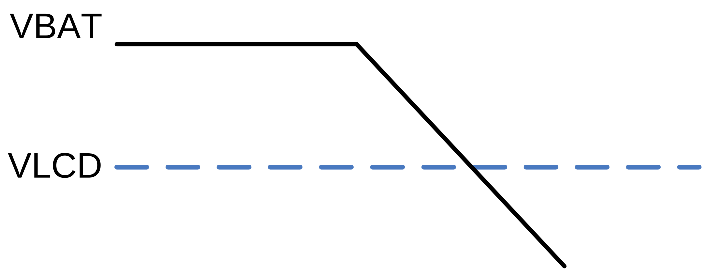


Figure 14. LCD0 Constant Contrast Mode

In auto-bypass contrast mode shown in Figure 15, the VLCD voltage is held constant and tracks the VBAT voltage below the threshold. When VBAT is above the threshold voltage set by VBMTH, VLCD regulates to the programmed contrast voltage (CTRST) using a variable resistor between VBAT and VLCD. When VBAT reaches the monitor threshold, the controller automatically enters bypass mode and powers VLCD directly from VBAT. The charge pump is always disabled in this mode, and the threshold and contrast levels do not have to be set to the same value, as shown in Figure 16.

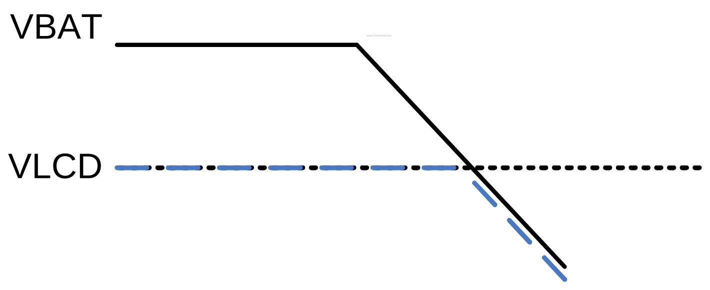


Figure 15. LCD0 Auto-Bypass Contrast Mode

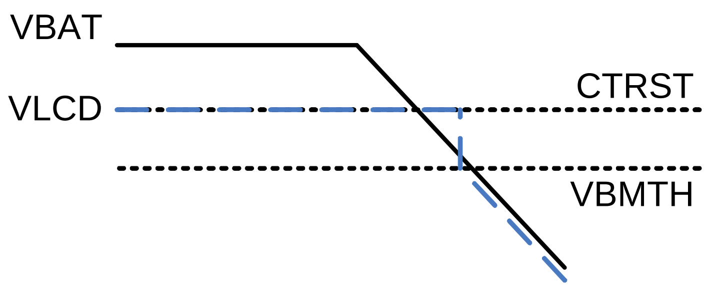


Figure 16. LCD0 Auto-Bypass Contrast Mode with Different Threshold and Voltage Settings

For lowest power consumption, applications should use bypass or minimum contrast modes whenever possible.

5.4.3. Refresh Rate

The LCD0 module on SiM3L1xx devices includes a programmable refresh rate by setting the CLKDIV bits in the CLKCONTROL register. The LCD multiplexor mode must be taken into account when determining the prescaler value. The LCD power consumption will scale proportionally with the refresh rate. For maximum power savings, choose the slowest LCD refresh rate and the minimum LCD0 clock frequency that provides acceptable performance for the application. For the least flicker, choose a fast LCD refresh rate.

5.5. Retention RAM

The retention RAM on SiM3L1xx devices is in eight 4 kB banks. All eight banks are available when not in PM8, but can be individually powered down when in PM8. The current leakage of the RAM at higher temperatures may increase, so disabling any unused banks when entering a low power mode may reduce system power consumption.

For most applications, it is recommended to enable retention for all RAM banks. The specific bank containing the stack pointer must be enabled to ensure proper code operation on an exit from PM8.

5.5.1. Adjusting the Stack Pointer with Keil µVision

With Keil µVision projects, the stack location can be controlled in the scatterfile. The default **linker_sim3l1xx_arm.sct** scatterfile can be found in the C:\SiLabs\32bit\si32-**x.y.z**\si32Hal\sim3l1xx directory, where **x** is the major si32HAL version, **y** is the minor version, and **z** is the trivial version.

Inside the scatterfile:

```
ARM_LIB_STACK (SI32_MCU_RAM_BASE + SI32_MCU_RAM_SIZE)
```

The value in the parentheses after the ARM_LIB_STACK label is the location of the stack pointer in memory. Change this value as needed for the application and ensure this RAM bank has retention enabled.

5.5.2. Adjusting the Stack Pointer with the Precision32 IDE

In the Precision32 IDE, projects can use a custom linker script file edited manually or the IDE dialogs to modify the stack pointer address. To add a custom linker script file manually:

Right-click on the project name in the Project Explorer view and select **New→Folder**. This folder must use the **linkscripts** name.

Copy the **link_template.ld** file from the Precision32 installation directory **..\IDE\precision32\Wizards\linker** to the project's **linkscripts** folder.

Open the project's link_template.ld file and edit the stack line near the bottom of the file:

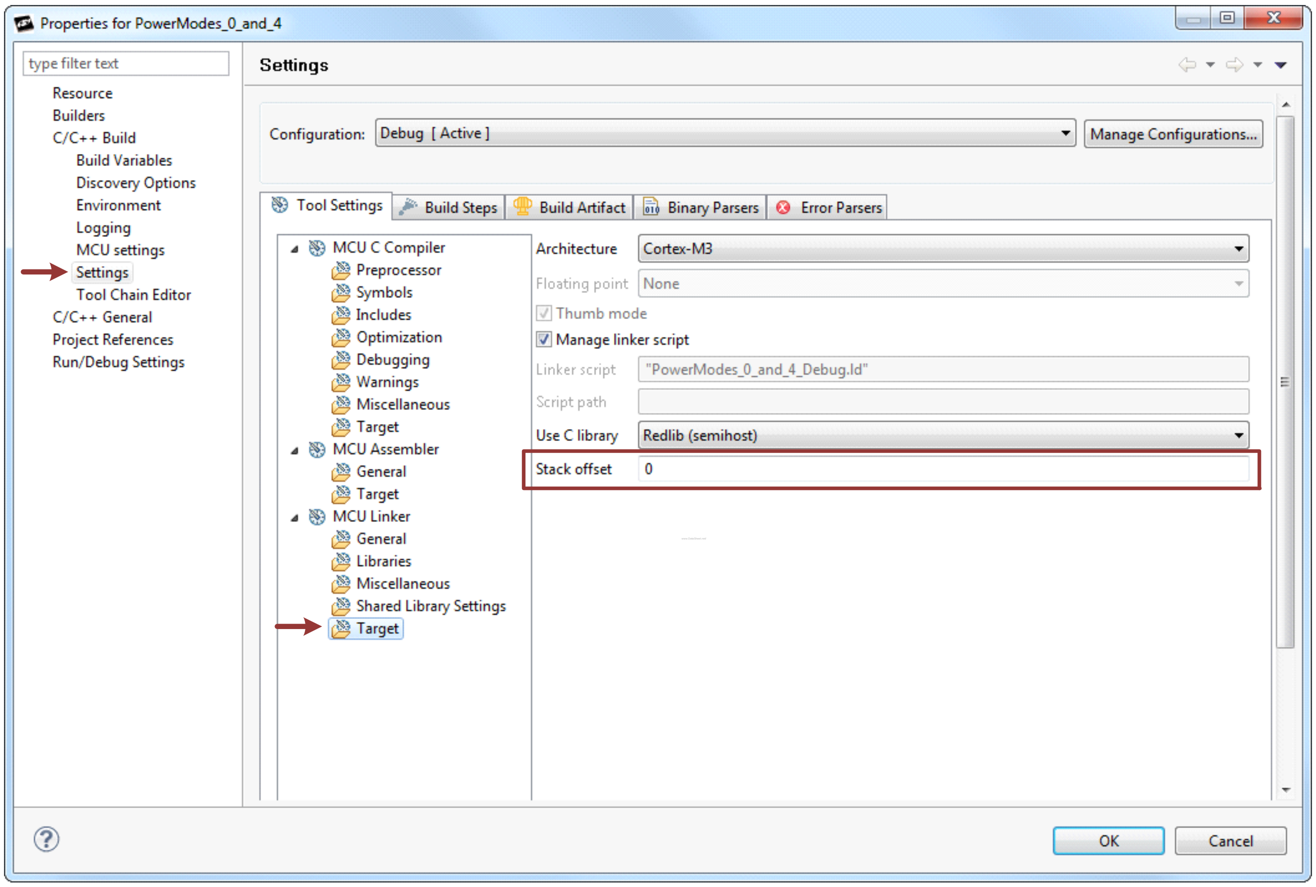
```
PROVIDE(_vStackTop = __top_${DATA} - ${STACK_OFFSET});
```

For example, to locate the stack at address 0x2000_6000:

```
PROVIDE(_vStackTop = 0x20006000);
```

Instead of editing the link scripts file, the stack offset change can be made inside the IDE:

1. Right-click on the **project_name** in the **Project Explorer** view.
2. Select **Properties**.
3. Click on **C/C++ Build→Settings→Tool Settings** tab→**MCU Linker→Target** and input the desired stack offset into the **Stack offset** field. This offset will occur from the top address of RAM, and the value must be a multiple of 4. Figure 17 shows this dialog in the Precision32 IDE.



Configuration: Debug [Active]

Architecture: Cortex-M3
Floating point: None
Thumb mode
☑ Manage linker script
Linker script: "PowerModes_0_and_4_Debug.ld"
Script path:
Use C library: Redlib (semihost)
Stack offset: 0

Figure 17. Using the Precision32 IDE to Select the Project Library

After changing either the custom linker script or IDE settings, clean and rebuild the project. View the map file for the project to verify the stack location changed as desired.

6. General Power-Saving Tips

6.1. Pins

Placing any unused pins in analog mode using the PBSTD module (PBMDSEL.x = 0) disables the drivers and weak pull-ups on the pin. The pin will float and consume little power.

For pins that are connected to external hardware, place the pins in analog mode if possible. If this is not possible, place the pins in a natural state that will consume no power. For example, if a pin has an external pull-down resistor, putting a ‘0’ in the latch will draw no current from the external pull-down.

If the system configuration allows it, disabling weak pull-ups on a port-by-port basis may also help reduce unnecessary power consumption in the pins.

6.2. Peripherals

Before entering the low power mode, disable any unwanted peripherals. Stopping the clock to the peripheral may disable the peripheral, but if the peripherals operate on a clock independent from the AHB or APB clocks, only the peripheral registers will be disabled. Disabling the module explicitly (DCDCEN = 0 for the DCDC0 module, for example) will ensure the module does not draw extra power in the low power mode.

6.3. Biases

In addition to peripherals, disable any unneeded bias sources in the device before entering the low power mode. These biases draw current to provide voltage references inside the chip. One example of a bias is the 1.2/2.4 V VREF0 module or the ADC internal 1.65 V voltage reference.

6.4. Clocks

External oscillators require a bias to start and maintain oscillation. If possible, switching to the internal oscillators and stopping the external oscillator will reduce the power consumption. The internal precision oscillator also requires extra current to operate. In most cases, switching the AHB and APB clock to the Low Power Oscillator (LPOSC0) or RTC0 oscillator before entering the low power mode will result in the lowest power consumption.

The SiM3L1xx devices feature programmable clocks for each peripheral. When entering a low power mode, disabling the clocks to any unused peripherals will reduce the system power consumption.

CONTACT INFORMATION

Silicon Laboratories Inc.

400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.