AN1508: SiWG917 Power Manager Application Note

The Power Manager is a comprehensive solution designed to optimize power usage in SiWG917, helping user achieve better energy efficiency and increased battery life. It incorporates FreeRTOS tickless idle mode, which minimizes power usage by transitioning the SiWG917 M4 into a low-power state when FreeRTOS operates on the Idle task.

This document provides an overview of the key features of Power Manager with tickless idle mode. Additionally, it covers important considerations when working with the Power Manager.

The users using Power Manager must have comprehensive knowledge of SiWG917 M4 power states. The information about the SiWG917 low-power modes and current consumption in different power modes and power states is in AN1430: SiWG917 Low-Power Application Note.

KEY POINTS

- Introduction to Power Manager in SiWG917
- Power Manager Architecture, features and advantages
- Power State Transition with Power Manager APIs
- Considerations while using Power Manager

Table of Contents

1. Power Manager Architecture

The Power Manager is a platform level service that manages the M4 power states and their transitions when requested by the application. These requirements are set by the different software modules (drivers, stacks, application code, and so on).

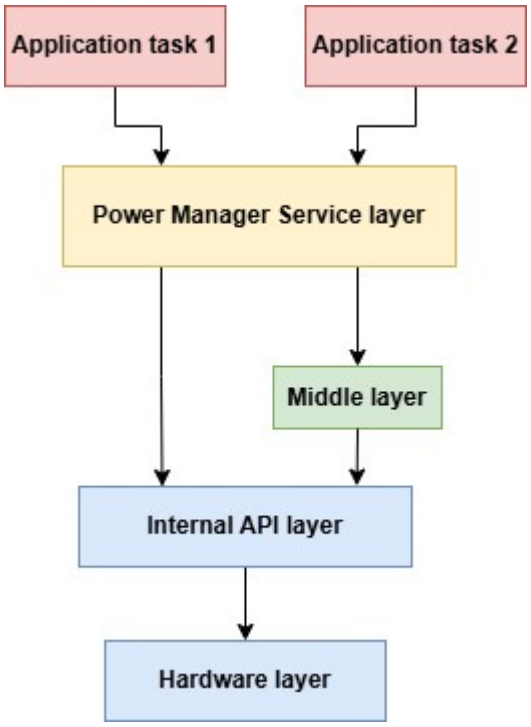

Figure 1.1. Power Manager Architecture

- **Application tasks** represent tasks running in the application. Users call the Power Manager service APIs from the application tasks.
- **Power Manager Service Layer** interacts with the application layer tasks. It is a layer over the existing internal core API layer simplifies the process for users to develop their applications.
- **Middle Layer** is a supporting layer for Power Manager service. It validates the power state transitions.
- **Internal API Layer** is the low-level driver which interacts with **hardware layer** where the register level configurations are done.

1.1 Features of Power Manager

- It is an Interface between software modules and the device, offering APIs to efficiently manage and configure the power state transitions of SiWG917.
- The SiWG917 M4 entry into a power state is determined by the requirements. Using the requirement APIs, requirements for a specific power state can be added or removed.
- It offers a notification mechanism, providing an option for subscribing/unsubscribing to events during state transitions, i.e., while entering or leaving any state. Based on the subscription, the Power Manager provides notifications through a callback function to the application. The notification events include Power State Transition, exit from sleep, and exit from standby mode. The main purpose of these notifications is for different software modules to adapt to the new power state.
- The Power Manager checks if it is ok to sleep and then enters the sleep.
- The clock scaling feature in differentiating system clock configuration in PS4 and PS3 active state is done. It can be switched between performance and power-save. By-default it is configured as power-save at the time of state change.

Note: PS1* check release notes on the current support for PS1 with Power Manager.

1.2 Advantages of Power Manager

- Integrating Power Manager into an application is seamless and provides following configurations. The configurations made in the UC in Simplicity Studio will be saved, allowing the Power Manager APIs to be reused as needed.
  - Handles peripherals, RAM banks and wake-up source configuration. Required peripherals can be enabled/disabled to reduce power consumption.
  - Enabling RAM retention, peripherals to be powered on, setting the wake-up sources.

- Provides easy and convenient ways to control all the power modes i.e., Active, Standby and Sleep as well as supporting all the power state transition i.e., PS4, PS3, PS2, PS1, PS0.
- To prevent race conditions and ensure secure concurrent access, the Power Manager APIs include safeguards such as enabling and disabling interrupts.
- Power Manager uses sleep timer for precision timing.

Note:

- The state hierarchy ranges from PS4 to PS0, with PS4 being the highest power state. However, the sleep mode in any power state is considered the lowest state. The Power Manager, with tickless mode, enables the system to enter sleep mode when the scheduler has no tasks to process.
- To know more about the Power states, please visit AN1430: SiWG917 Low-Power Application Note.

2. Components

Power Manager Components facilitate the user to configure the RAM retained, the peripherals to remain powered on/off, and sets the wake-up sources prior to the M4 entering sleep mode. Following are the Power Manager components (enabled using the UC in Simplicity Studio):

- Power Manager
  - Power Manager Configuration
  - Wakeup source Configuration
    - Calendar Wakeup
    - GPIO Wakeup
    - Deep Sleep Timer
    - Wireless Wakeup
- UULP peripheral component for the selected wake-up source
- Low Power

To know more about installation of components for integration of Power Manager, please visit Power Manager Integration.

2.1 Power Manager Component

Power Manager service is initialized with the installation of this component, user need not initialize the Power Manager service.

Figure 2.1. Power Manager Component

The following components are automatically installed when Power Manager component is installed.

2.1.1 SI91X TICK-LESS MODE Component

The Tickless idle mode feature will suspend the SiWG917 M4 and keep it in sleep when no tasks are scheduled to run.

2.1.2 Power Manager Configuration Component

There are two components pertaining to the Power Manager configuration,

- Power Manager Advance Configuration
- Power Manager Configuration

These components provide control over peripheral enable/disable and RAM retention. Peripheral and RAM retention is configured with the installation of this component, user intervention is not required.

Note: The user can install either 'Power Manager Configuration' or 'Power manager Advance Configuration' component.

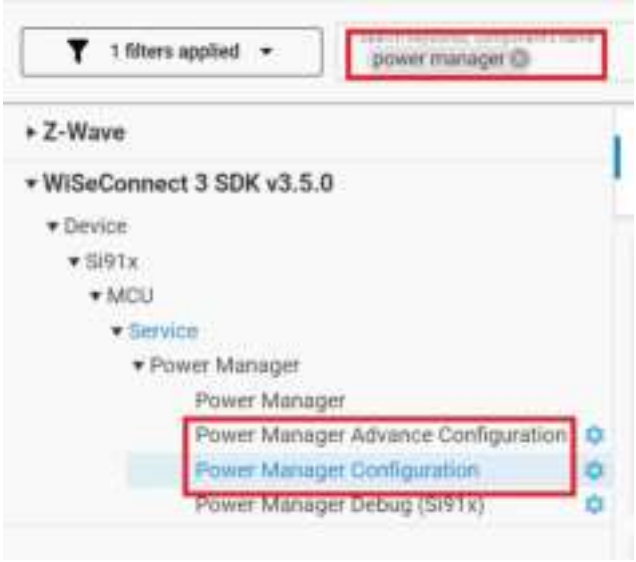

Figure 2.2. Power Manager Configuration Component

2.1.2.1 Power Manager Configuration

This component is a basic configuration component in which the user can select which peripherals need to be powered on/off according to the domain of the peripheral.

Peripheral Configuration

- High Power Peripherals (available in PS4/PS3 power states)
- Low Power Peripherals (available in PS2 power states)
- Ultra Low Power Peripherals (available in all power states)

The peripheral availability in different power states and the list of different group of peripherals for which power is controlled through software is given in 'M4 Power States' section of AN1430: SiWG917 Low-Power Application Note.

Figure 2.3. Power Manager Configuration – Peripheral Configuration

RAM Configuration

This component also provides configuration for RAM retention. Two options are provided to configure the RAM banks i.e. using size or bank number. The user needs to select any one option.

Figure 2.4. Power Manager Configuration – RAM Configuration

Note: If both RAM banks using size & bank numbers are enabled, RAM banks using size will be configured by default.

2.1.2.2 Power Manager Advance Configuration

This component is an advance configuration component in which the unwanted peripherals and RAM banks are powered off as shown in the following figure by default. Configurations in this are made to get the optimum power consumption.

Peripheral Configuration

- High Power Peripherals (available in PS4/PS3 power states)
- Low Power Peripherals (available in PS2 power states)
- Ultra Low Power Peripherals (available in all power states)

Note: SiWG917 M4 current may vary as per the selection of peripherals in addition to the default configuration.

Figure 2.5. Power Manager Advance Configuration – Peripheral Configuration

RAM Configuration

The default configuration for RAM retention is already configured to provide the lowest SiWG917 M4 current but if required, the user can configure the RAM banks using size or bank number.

Note: SiWG917 M4 current may vary as per the selection of RAM banks in addition to the default configuration.

Figure 2.6. Power Manager Advance Configuration – RAM Configuration

2.1.3 Wakeup Source Configuration Component

This component provides the initialization and configuration for the UULP wake-up sources used in PS4 sleep/PS3 sleep/PS2 sleep.

The UULP wakeup sources that are provided are as follows,

- Calendar or Alarm Wakeup
- GPIO Wakeup
- Deep Sleep Timer Wakeup
- Wireless Wakeup

Figure 2.7. Power Manager Wakeup Source Configuration Component

Note: The user must install the respective **UULP peripheral components** that are selected as a wake-up source. This would get all the dependencies and files related to the peripheral selected.

2.1.3.1 Calendar or Alarm Wakeup

Calendar wakeup or Alarm wakeup source can be enabled and configured in terms of seconds and milliseconds. By default, the alarm wake-up source is configured with 5 second alarm trigger. This wakeup source can be enabled using the toggle.

The alarm timer starts running as soon as the wakeup source is initialized prior to M4 sleep and triggers an interrupt upon the timer expiry. The M4 wakes up upon this interrupt.

Note: Milli second timer is recommended to be used, when M4 is in active state. Because the sleep and wake-up transition of M4 takes ~8 ms. The suggested minimum value for this timer is 100 ms.

Alarm time reconfiguration during runtime is not possible with Power Manager. If a user wants to change alarm time in runtime, the user should configure the alarm wake-up source manually with *'sl_si91x_power_manager_set_wakeup_sources()*' API without using UC. So, in this case, the initialization of the wake-up source should be done by the user in the application.

Note: The alarm timer is periodic as it is configured every time M4 enters sleep inside the '*sl_si91x_power_manager_sleep()*'.

Figure 2.8. Calendar Wakeup

2.1.3.2 GPIO Wakeup

There are 4 UULP GPIO (0 to 3) available as mentioned following figure which can act as a wake-up source. Enabling the GPIO Wake-up allows the user to select the desired GPIO pin as a wake-up source.

The SiWG917 M4 wakes up based on the input from the configured UULP GPIO. Users can configure the polarity of the GPIO wakeup source.

Figure 2.9. GPIO Wakeup

2.1.3.3 Deep Sleep Timer

Deep Sleep Timer is used as wakeup source for the SiWG917 M4 in sleep. This timer works only during M4 sleep. The sleep time is configurable in micro-seconds.

Note: Even if the user configured the timer when M4 is in active state, this timer will start when M4 enters sleep mode. The maximum configurable time is 2^{32} micro-seconds (~71 minutes).

Figure 2.10. Deep Sleep Timer Wakeup

2.1.3.4 Wireless Wakeup

This wake-up source is used when the user wants to wake up M4, when a wireless remote message is received by the NWP. Upon receiving the message, NWP will trigger the M4 to wake up.

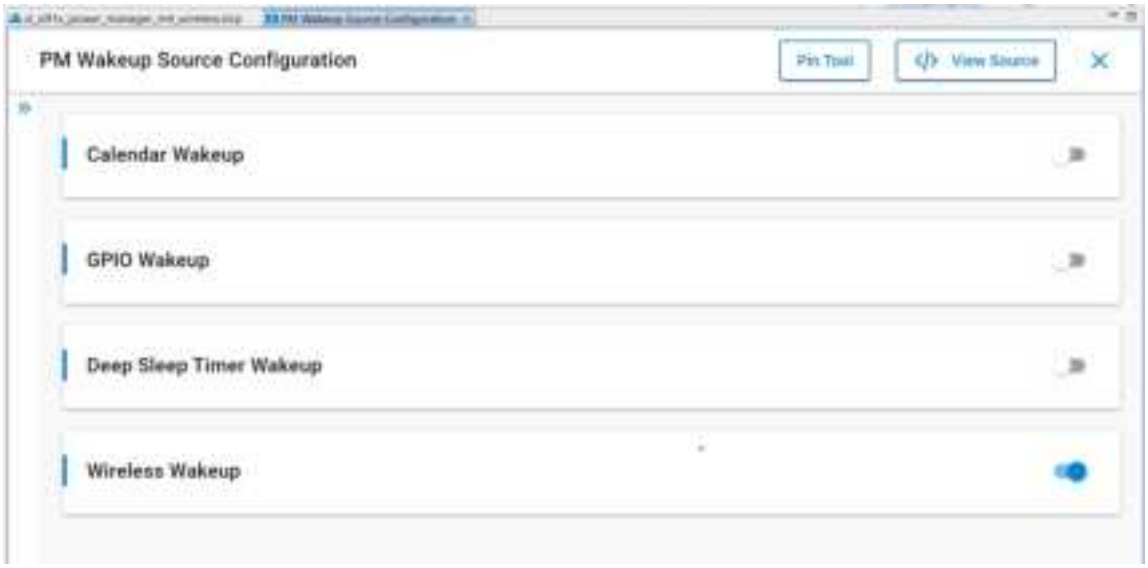

Figure 2.11. Wireless Wakeup

2.2 Low Power Peripheral Component

Low Power Peripheral component is for using the ULP peripheral in PS2 state. The installation of this component will move the required driver files to RAM as in PS2 state, the flash will be turned off and M4 can execute from RAM.

Note: Initialization and configuration of the peripheral should be done by the user manually in the application.

For more information, please refer to Power Manager Components integration.

3. Power Manager APIs

This section describes some of the APIs offered by Power Manager. For more information on APIs and API examples, see the Power manager APIs documentation.

Table 3.1. Power Manager APIs

API	Description
sl_si91x_power_manager_init	To initialize the Power Manager service.
sl_si91x_power_manager_add_ps_requirement	To add a requirement on power states.
sl_si91x_power_manager_remove_ps_requirement	To remove the requirement on power states.
sl_si91x_power_manager_set_clock_scaling	To configure the clock scaling – Powersave and Performance.
sl_si91x_power_manager_add_peripheral_requirement	Adds the peripheral requirement.
sl_si91x_power_manager_remove_peripheral_requirement	To remove the peripheral requirement.
sl_si91x_power_manager_subscribe_ps_transition_event	To register a callback to be called on given power state transition(s).
sl_si91x_power_manager_unsubscribe_ps_transition_event	To unregister an event callback handle on power state transition.
sl_si91x_power_manager_sleep	To move into sleep mode and wait for the peripheral to be set as a wakeup source to trigger and wake up the M4. **Note:** Applications using RTOS with tickless mode enabled must not call this function. This API is not supposed to be used directly in the application and is called automatically when system is in idle state with tickless mode.
sl_si91x_power_manager_set_wakeup_sources	To configure the wakeup sources.
sl_si91x_power_manager_configure_ram_retention	To retain the RAM in low power state either by using size or RAM bank as input parameters.
sl_si91x_power_manager_get_current_state	To return the current power state.
sl_si91x_power_manager_get_requirement_table	To get the current requirements on all the power states.
sl_si91x_power_manager_deinit	To de-initialize the Power Manager service.

4. Power Manager with Tickless Idle

Power Manager Tickless Mode is a feature provided by the FreeRTOS that allows the SiWG917 to reduce power consumption by entering a low-power mode when there are no tasks to execute.

In traditional RTOS implementations, a periodic timer interrupt, known as a tick, is used to keep track of time and trigger the RTOS scheduler to determine which task should run next. However, in applications where power consumption is critical, the continuous generation of timer interrupts can be inefficient and lead to unnecessary power consumption.

Tickless mode in FreeRTOS addresses this issue by dynamically adjusting the timing of the tick interrupt based on the requirements of the tasks. When there are no tasks ready to execute, the SiWG917 M4 can enter sleep mode, effectively reducing the frequency of timer interrupts and conserving power.

Note: As Tickless Idle mode configures SysRTC as a mechanism to generate the Real-Time Operating System (RTOS) ticks. These ticks are essential for the RTOS to manage timing and scheduling tasks effectively. It is strongly advised against utilizing the SysRTC peripheral for any purpose other than its designated functionality.

4.1 Implementation

A low-power idle hook function that FreeRTOS will call when there are no tasks ready to execute is implemented. This hook function should put the SiWG917 M4 into sleep.

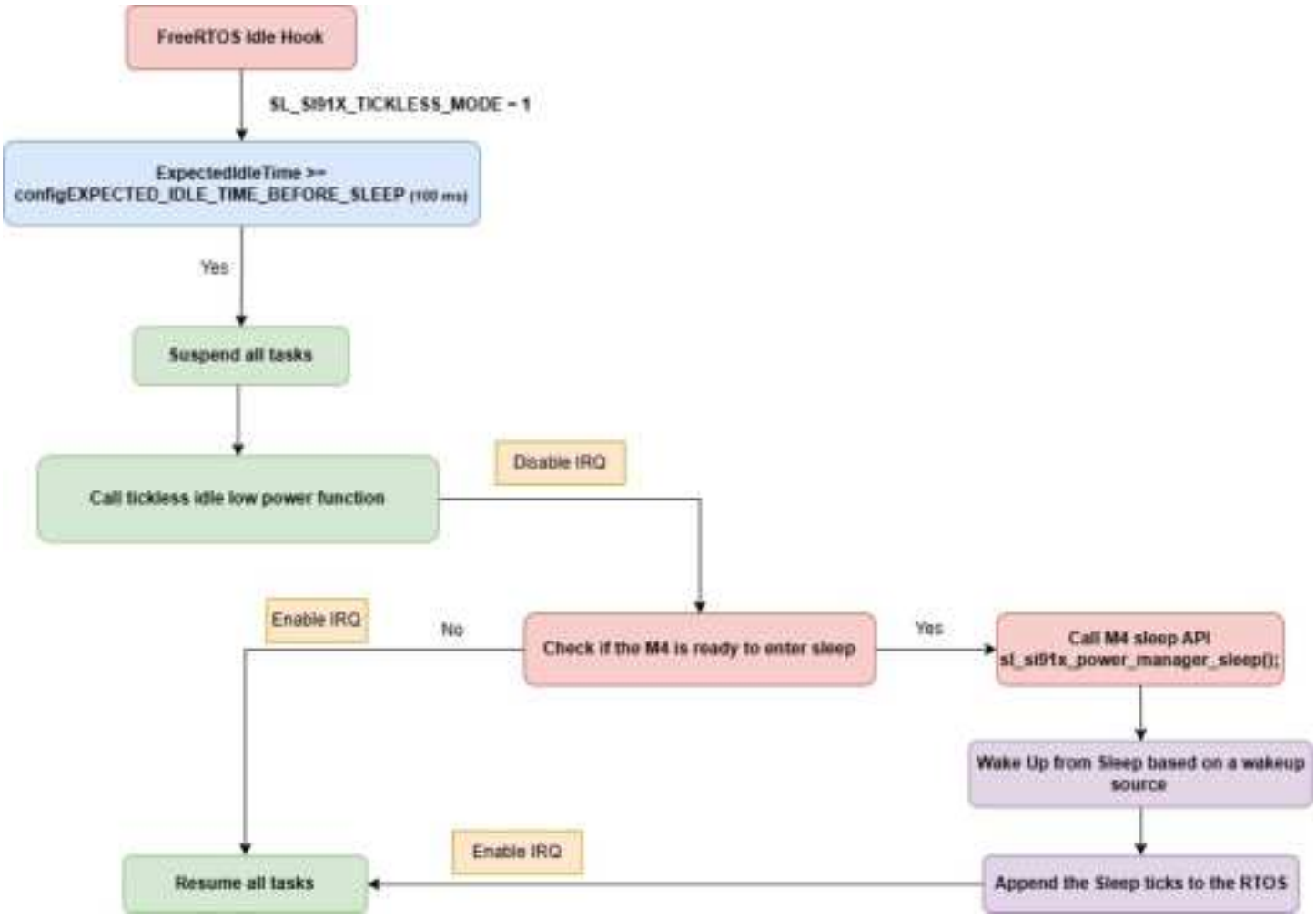

Figure 4.1. Tickless Idle Mode Implementation in FreeRTOS

1. The application tasks call **osDelay, vTaskDelay, osSemaphoreAcquire** or other blocking functions, leading the scheduler to consider them idle. When there are no tasks to be executed, the application will execute FreeRTOS idle hook.
2. The application checks if the expected idle time is greater than or equal to a predefined threshold (configEXPECTED_IDLE_TIME_BEFORE_SLEEP), which is set to 100 milliseconds. If the condition is met, the application suspends all tasks.

3. The application then calls a function specifically designed to handle the M4 transition to low power mode **portSUPPRESS_TICKS_AND_SLEEP(xExpectedIdleTime)** and IRQs are disabled to ensure that no interrupts occur while the system is preparing to enter low power mode.
4. Then the application checks for the following before letting M4 go to sleep,
   - **Check Active Events:** It checks if any ongoing tasks or operations require the system to remain active, for example, data transmission, sensor readings, or critical computation. Also checks if there is any requirement added, and the device goes to sleep if there is no power manager requirement added.
   - **System Flags:** Internal flags or states that indicate whether it is appropriate to sleep or not are likely checked. Checking the pending packets from NWP while M4 going to sleep.
   - **Flash operation:** Check if there are any flash write operations like NVM command is in progress and the device goes to sleep if there are no flash operations.
5. If the M4 processor is ready to go to sleep mode, it calls the M4 sleep API (sl_si91x_power_manager_sleep()) to put the M4 processor into sleep mode.
6. The M4 processor wakes up from sleep mode when a predefined wakeup source triggers it. Once it wakes up, the duration for which the system was in sleep mode (known as sleep ticks) is added to the RTOS.
7. IRQs are enabled after the system wakes up from sleep mode and all tasks that were suspended are resumed, and the application continues normal operation.

Note:

- The user must be cautious when using ***osSemaphoreAcquire*** and ***osDelay***, as the M4 enters sleep during this configured delay time and will not serve any event interrupts.
- Power Manager component shall install the wireless wake-up and SysRTC component for sleeptimer as wake-up resource by default. It is highly recommended not to install these components explicitly.

If no tasks are ready to run, the kernel enters tickless idle mode. The system wakes up upon sleep timer expiration or any wake-up source configured. Here's a basic flow chart to illustrate the system's behavior.

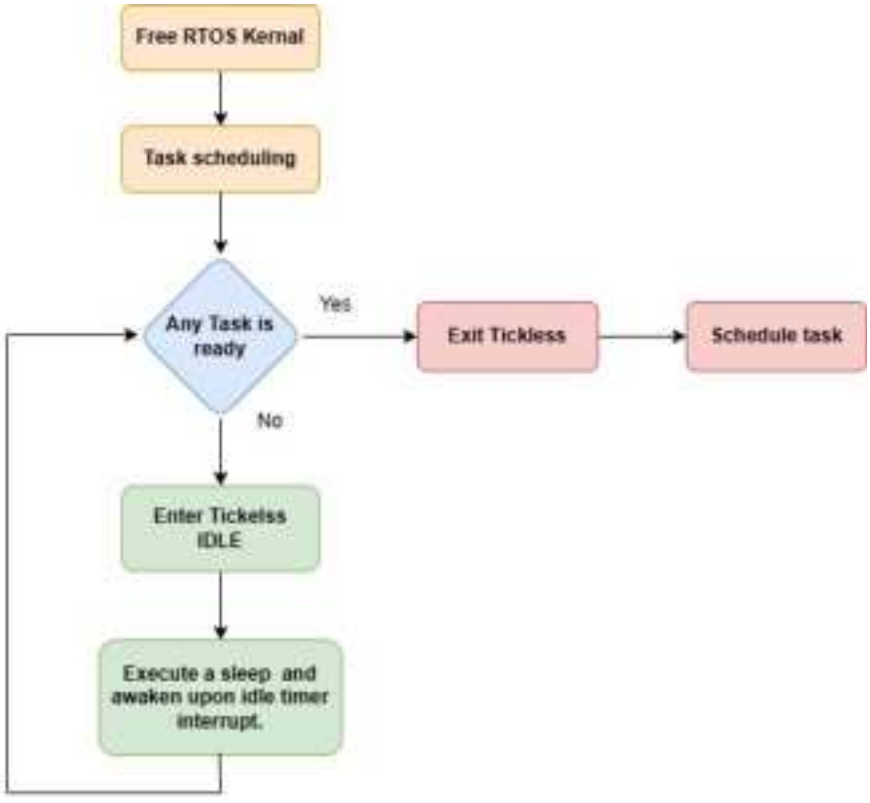

Figure 4.2. Interaction between the Task, Power Manager, and FreeRTOS kernel

Note: Users can take SL POWER MANAGER TICKLESS IDLE existing application in WiSeConnect release as reference, which demonstrates the power manager service APIs, state transitions(PS4, PS3 and PS2) and sleep-wake up (PS4, PS3 and PS2 sleep with RAM retention) with tickless idle mode.

4.2 Use Cases

4.2.1 Application with Peripheral only

Tickless idle mode activates sleep functionality automatically when scheduler goes to idle, Power Manager has APIs that prevents the system from entering sleep mode.

When a requirement is added for a power state, then the system is prevented from entering sleep as there is a requirement higher than sleep mode is added. Make sure to remove the higher requirement added if the user wants to let the M4 enter sleep mode in tickless idle mode. For instance, during data transmission on a USART, the system can enter sleep mode if there is any delay during transmission. To address this, the Power Manager adds a requirement on PS4 before initiating the transfer and removes the requirement once the transfer is complete, thereby preventing the M4 from entering sleep mode.

Below is the flow for how to handle peripherals across sleep wakeups with tickless idle.

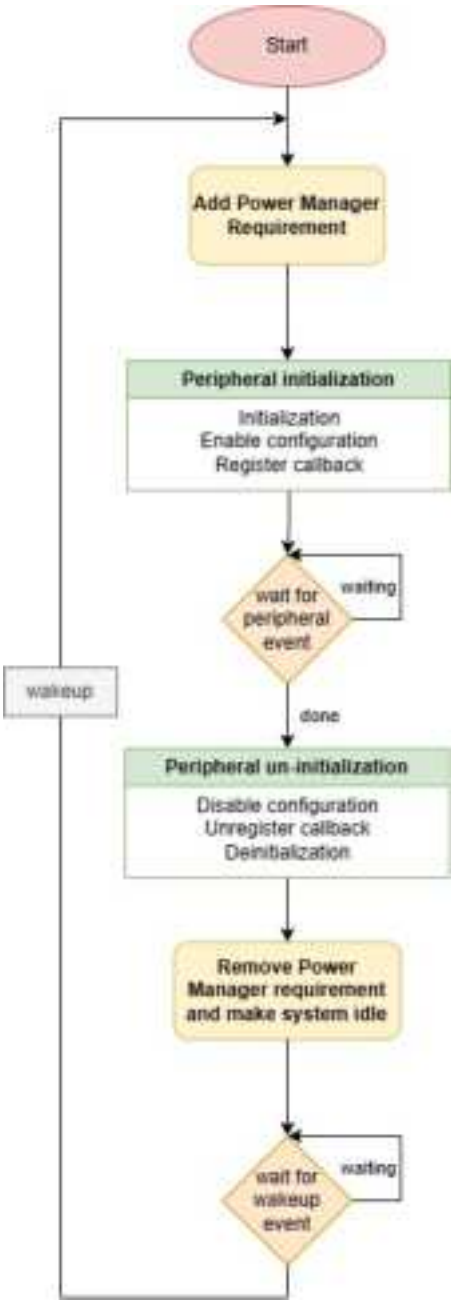

Figure 4.3. Handling Peripherals with Tickles Idle Mode

4.2.2 Application with Wireless Functionalities and Peripherals

A sequence flow illustrating a use case for wireless applications that utilize peripherals with tickles idle mode.

1. Power state (PS4) requirement is added to prevent the M4 entering sleep during any delay unintended.
2. The SiWG917 initializes the NWP and M4 peripheral.
3. The NWP is configured as a WLAN station, connects to an Access Point (AP) and establishes a connection to the Cloud.
4. The NWP is set to Connected Power Save mode (Associated Power Save).
5. The peripheral activity is carried out and data is collected. The peripheral is un-initialized before entering M4 sleep.
6. The power state requirement is removed allowing the M4 to enter sleep.
7. The M4 is waiting on a semaphore with defined wait time. The RTC timer, UULP GPIO, and Wireless message are defined as wakeup sources.
8. The M4 enters last active state (PS4) sleep mode with retention as the scheduler is idle.
9. Upon a GPIO interrupt/ RTC timer expiry/ wireless message/ semaphore wait time expiry, the M4 wakes up.
10. Upon M4 wakeup, add the power state requirement (PS4).
11. For performing the peripheral activity again, re-initialize the peripheral and execution continues from step 5.

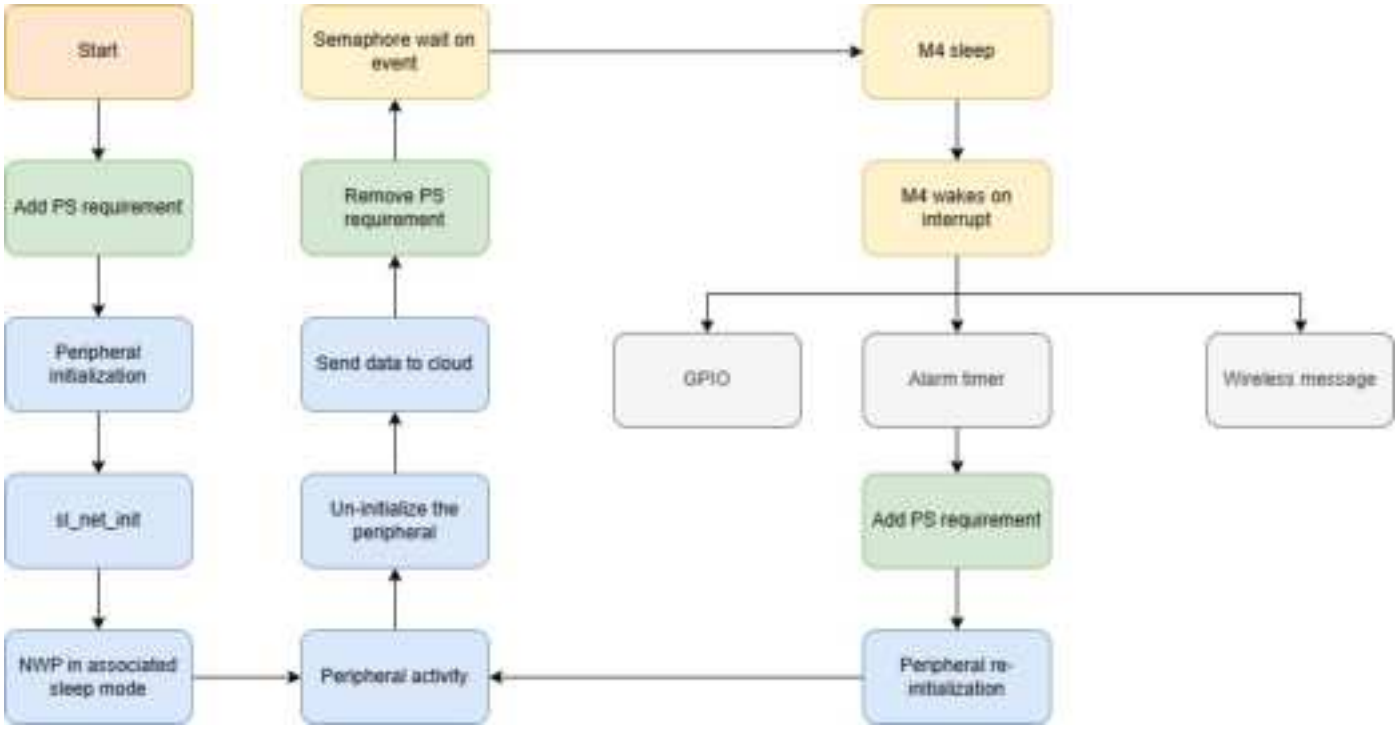

Figure 4.4. Handling Peripherals with Wireless Activity in Tickles Idle Mode

5. Sequence Diagram

On boot up the power manager starts with PS3 powersave state (40MHz) and after that application adds its required state through API calls.

The sequence diagram for transition from PS3 to PS2 state is as follows.

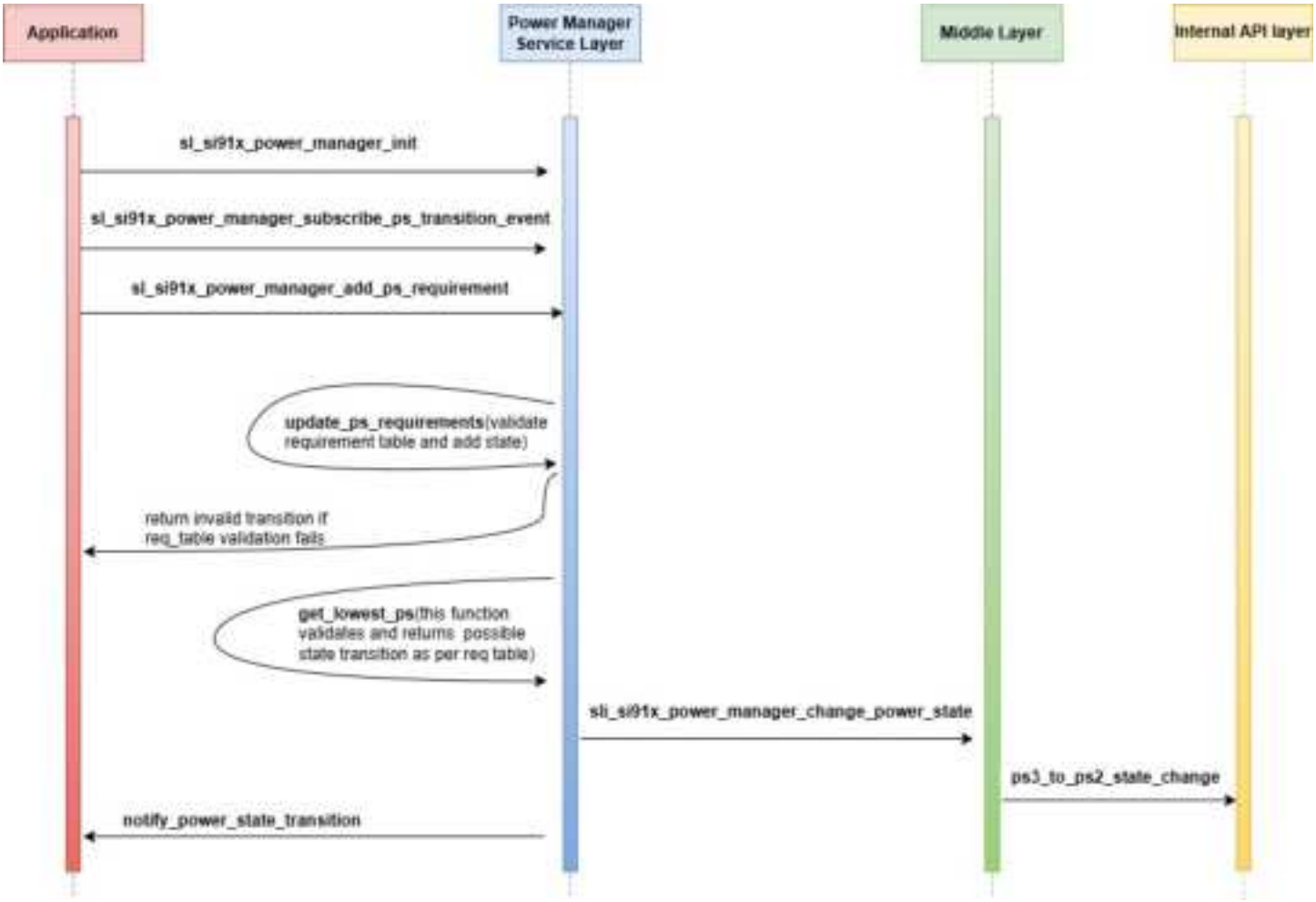

Figure 5.1. Sequence diagram showing transition from PS4 to PS2 state

The Power Manager service is initialized by default using the **sl_si91x_power_manager_init()** function, which sets the processor to the PS3 state with a clock speed of 40 MHz (Power Save). All possible events are subscribed to using the **sl_si91x_power_manager_subscribe_ps_transition_event()** function, which includes the callback function address.

To add a requirement for the PS2 state, the **sl_si91x_power_manager_add_ps_requirement()** function is used. The **sli_si91x_power_manager_update_ps_requirement()** function then updates the power state requirement, current state, and requirement table, ensuring that the added requirement is a valid transition.

The **sl_si91x_get_lowest_ps()** function validates all power state requirements and returns the possible state transition based on the state hierarchy. For example, if both PS4 and PS2 state requirements are added, the function will return PS4 as the possible transition since it is considered higher than PS2. The system will not transition to PS2 until the PS4 requirement is removed. However, here as only the PS2 state requirement is added, the function will return PS2 as the possible transition.

Finally, the **sli_si91x_power_manager_change_power_state()** function updates the power state from PS3 to PS2, and the **ps3_to_ps2_state_change()** function executes this transition using internal APIs.

5.1 Power State Transitions

This section provides a simple illustration for power state transition using Power Manager APIs.

- To add a power state requirement - `sl_si91x_power_manager_add_ps_requirement (sl_power_state_t state`
- To remove a power state requirement - `sl_si91x_power_manager_remove_ps_requirement(sl_power_state_t state);`

- The power states are,
  - `SL_SI91X_POWER_MANAGER_PS0 - PS0 Power State`
  - `SL_SI91X_POWER_MANAGER_PS2 - PS2 Power State`
  - `SL_SI91X_POWER_MANAGER_PS3 - PS3 Power State`
  - `SL_SI91X_POWER_MANAGER_PS4 - PS4 Power State`

Note:

- Requirement table stores the requirements for each power state and based on that power manager decides the current state. **sl_si91x_power_manager_get_requirement_table** API is used to get the current requirements on all the power states.
- If a task wants to enter a low power mode (such as PS3 or PS2), it first checks if any other task has a higher requirement.
- When multiple tasks request a state transition, the Power Manager maintains the possible state transitions. It is recommended that a task checks the requirement table before switching and performs the state change accordingly.

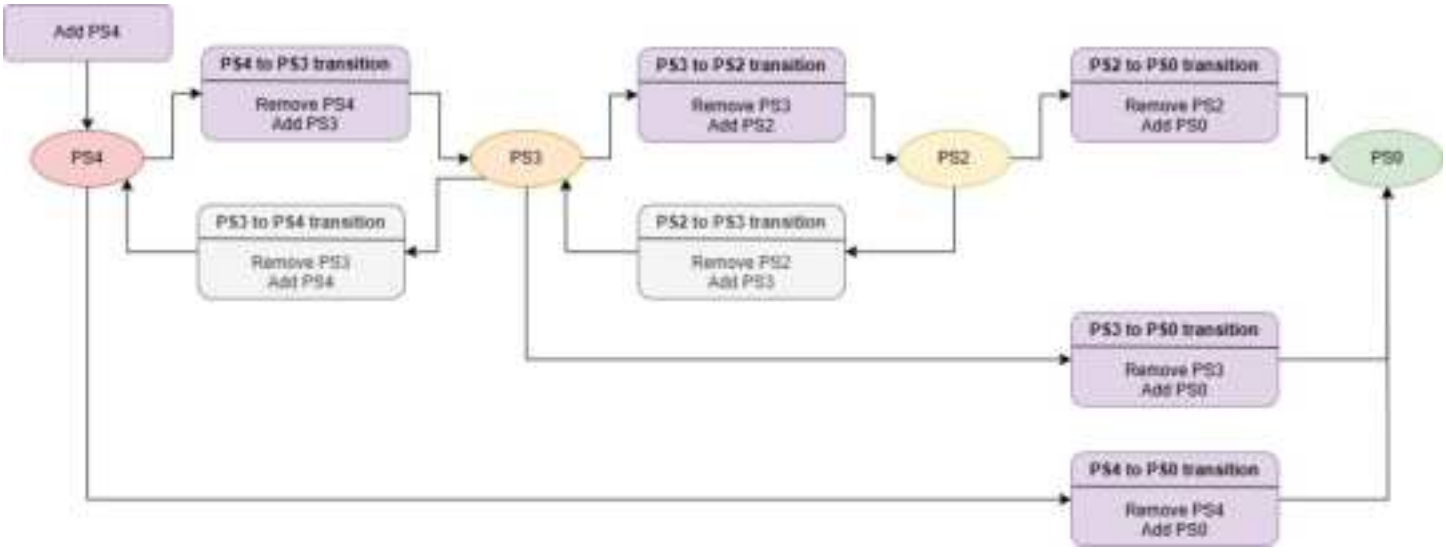

Figure 5.2. Power State Transitions

6. Considerations

This section focuses on considerations when working with Power Manager.

6.1 Add/Remove PS requirement

The current state of the M4 is the most recent active state it transitioned to. For instance, if the PS4 state requirement is added and then removed, followed by the addition and removal of the PS3 state, the current state of the M4 would be PS3.

In a multi-threaded application,

- There may be many requirements added, and Power Manager is maintaining the highest state requested.
- All the requirements of the higher power state must be removed for the M4 to transition to the next lower power state.
- If multiple requirements are added, the state will change upon removing all the requirements.

6.2 Remove peripheral requirement

If any peripheral is disabled in the PS4 state (for example, disabling SSI), upon switching to the PS2 state and then back to the PS4 state, all peripherals will be powered on during the state transition. The ***sl_si91x_power_manager_remove_peripheral_requirement()*** can be used again to disable the required peripherals.

6.3 Measuring power consumption

To measure the power consumption of only,

- NWP - Configure M4 in PS0 without any RAM retention.
- M4 - Configure NWP in DEEP_SLEEP_WITHOUT_RAM_RETENTION (Deep Sleep without RAM retention when the device is not associated with AP).

7. Appendix

Table 7.1. Abbreviation table

S.N.	Abbreviation	Description
1	AP	Access Point
2	API	Application Programming Interface
3	GPIO	General Purpose Input/Output
4	NWP	Network Processor
5	PS	Power States (PS4, PS3, PS2, PS1, PS0)
6	QSPI	Quad Serial Peripheral Interface
7	RAM	Random Access Memory
8	RTC	Real Time Clock
9	RTOS	Real Time Operating System
10	SSI	Synchronous Serial Interface
11	UC	Universal Configurator
12	ULP	Ultra Low Power
13	UULP	Ultra Ultra Low Power
14	USART	Universal Synchronous/Asynchronous Receiver/Transmitter
15	WLAN	Wireless Local Area Network

8. Revision History

Revision 1.0

June, 2025

Initial release.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com