



# AN1430: SiWG917 Low-Power Application Note

---

This document provides information about the SiWG917 low-power modes and current consumption values in different power modes and power states. It also details the low-power reference examples in the WiSeConnect3 SDK.

**Note:** This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit [www.silabs.com/about-us/inclusive-lexicon-project](http://www.silabs.com/about-us/inclusive-lexicon-project).

## KEY POINTS

---

- SiWG917 basic hardware blocks
- M4 Power Modes, Wakeup Sources
- NWP Power Modes, Sleep-Wakeup illustrations
- Typical Current Consumption of SiWG917

## 1. Introduction

The SiWx917 SoC, hereby termed as SiWG917, achieves ultra-low-power without compromising on the features that have been traditionally considered "power-hungry". Through Dynamic Voltage and Frequency Scaling, careful optimization, and hierarchical design for all components of the SoC, the SiWG917 achieves better power consumption using Cortex-M4.

The SiWG917 includes two processors: Silicon Labs' ThreadArch® (TA) and an ARM® Cortex® M4 Processor. All the networking and wireless stacks run on independent threads of the ThreadArch. The Cortex-M4 is dedicated to peripheral and application-related processing.

## 2. SiWG917 Hardware Block Diagram

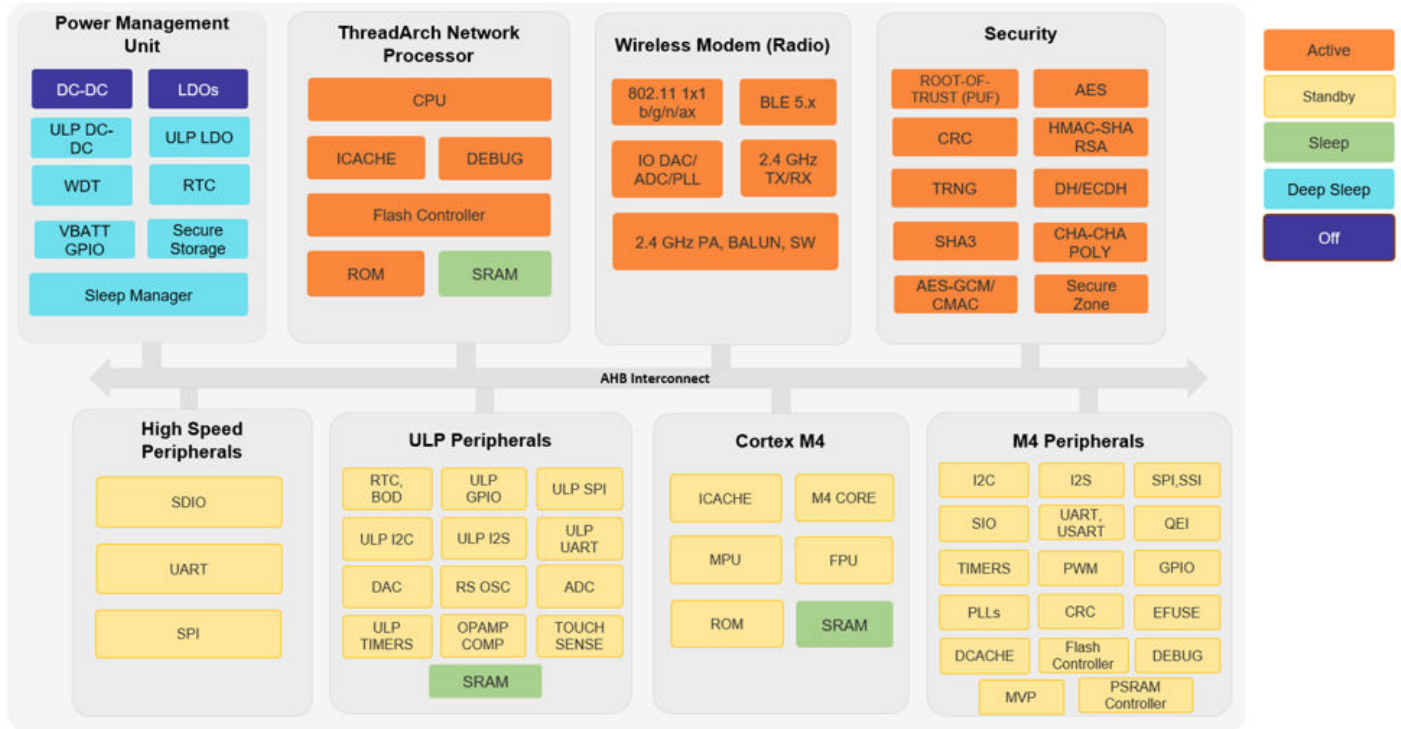


Figure 2.1. SiWG917 Block Diagram










The basic blocks of SiWG917 can be listed as:

- ThreadArch Network Processor: Multi-threaded processor that runs wireless and network stacks on independent threads
- Power Management Unit: Responsible for supplying power required for various domains of SiWG917 chipset
- Wireless Modem (Radio): Includes Wi-Fi, Bluetooth, Analog Front-End, 2.4 GHz RF transceiver, and integrated power amplifier sub-systems
- Security: Contains security feature blocks like hardware accelerators, secure firmware updates, etc.
- Peripherals such as High-speed peripherals, ULP peripherals, M4 peripherals
- Cortex-M4: A high-performance 32-bit processor that runs the microcontroller unit

**Note:** The "ThreadArch Network Processor", "Cortex-M4", and "SiWG917" shall be referred to as "NWP", "M4", and "SoC" respectively, throughout this document for the sake of brevity.

The SoC low-power compatibility matrix can be illustrated as below:

**Table 2.1. SiWG917 M4 vs NWP Power Modes Compatibility Matrix**

	NWP Active	NWP Connected Sleep	NWP Unconnected Sleep with Retention	NWP Unconnected Sleep without Retention
M4 Active (PS4/PS3/PS2)				
M4 PS4/PS3/PS2 Standby				
M4 PS4/PS3/PS2 Sleep or PS1				
M4 DeepSleep				

M4 Application/Firmware should be executed from RAM in the following scenarios:

- M4 Active and NWP Unconnected Sleep without Retention mode
- M4 in PS2 Active and NWP in either Active or Sleep mode

Before delving into the SoC (M4 and NWP combined) power save, it is important to know the M4 power save and NWP power save independently.

### 3. M4 Power Save

The operational states of the SiWG917 M4 sub-system are called power states (PSx) and are numbered from PS0 to PS4. The power states offer different levels of functionality and thus also varying power consumption, allowing designers to scale the resources to fit the bare minimum of what is needed in the application at any given time.

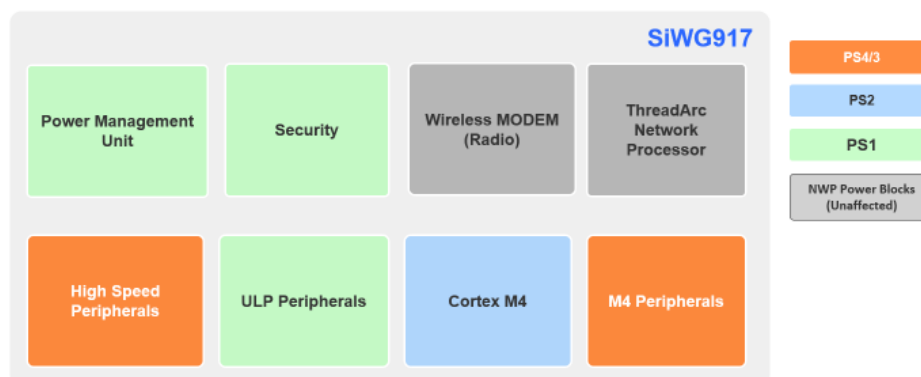
The M4 power states are segregated into power modes as listed below:

- Active mode: CPU is powered ON, operating at the configured frequency
- Standby mode: CPU is clock-gated, the remaining system operating at the configured frequency
- Sleep mode: CPU is power-gated, RAM can be retained
- Deep sleep/Shutdown mode: CPU is power-gated, RAM cannot be retained

### 3.1 Active Mode Power States

The SoC in Active Mode can be in any of the four power states, i.e., PS4/PS3/PS2/PS1.

After reset, the processor starts in the PS4 Active state, the highest activity state where the full functionality is available. The other Active states (PS3/PS2/PS1) will have limited functionality or processing power.



**Figure 3.1. Active Mode Power States**

**PS4 Active:** Highest power consumption state, having complete functionality available for M4

**PS3 Active:** Complete functionality is available, operating at a lower voltage thereby reducing current consumption

**PS2 Active:** Limited functionality is available, operating at a much lower voltage compared to PS4 and PS3

**PS1:** CPU is off, and limited peripherals are powered ON; these peripherals need to be configured for desired functionality before entering this state

The functional characteristics of each of the active modes can be understood using the following table:

**Table 3.1. M4 Active Mode Power States**

Power State	PS4 Active	PS3 Active	PS2 Active	PS1
CPU operating frequency (max)	180 MHz	90 MHz	20 MHz or 32 MHz	OFF
SRAM operating voltage	LDO SoC 1.1 V	LDO SoC 1.0 V	LDO SoC 0.7 V or DC-DC 1.0 V	LDO SoC 0.7 V or DC-DC 1.0 V
Power domains ON	All	All	Applications ULP peripherals UULP peripherals Analog peripherals	ULP peripherals UULP peripherals Analog peripherals
GPIOs available	SoC GPIOs ULP GPIOs UULP GPIOs	SoC GPIOs ULP GPIOs UULP GPIOs	ULP GPIOs UULP GPIOs	ULP GPIOs UULP GPIOs
SRAM powered ON	320 kB LP SRAM 8 kB ULP SRAM	320 kB LP SRAM 8 kB ULP SRAM	320 kB LP SRAM 8 kB ULP SRAM	320 kB LP SRAM (can only be retained) 8 kB ULP SRAM

**Note:** For more information on power domains, refer to the SiWG917 Hardware Reference Manual.

### 3.2 Standby States

The M4 in Standby Mode can be in three power states i.e., PS4-Stansby, PS3-Standby and PS2-Standby. Each power state can be entered from its respective Active state using WFI (Wait For Interrupt) instruction.

The functional characteristics of this mode include:

- CPU clock-gated
- SRAM operate at the same voltage as it's respective Active state
- Peripherals, GPIOs and SRAM available are same as it's respective Active state

Any interrupt can be used as a wakeup source in the Standby mode, to achieve the transition from Standby mode to Active mode.

### 3.3 Sleep Mode Power States

The M4 in Sleep Mode can be in three states, i.e., PS4-Sleep, PS3-Sleep, and PS2-Sleep. Each power state can be entered from its respective Active state using software instruction.

The functional characteristics of this mode include:

- CPU power-gated
- UULP peripherals are functional. Peripherals need to be configured for desired functionality, before entering this state
- UULP-VBATT GPIOs are powered ON
- 320 kB LP-SRAM can be retained

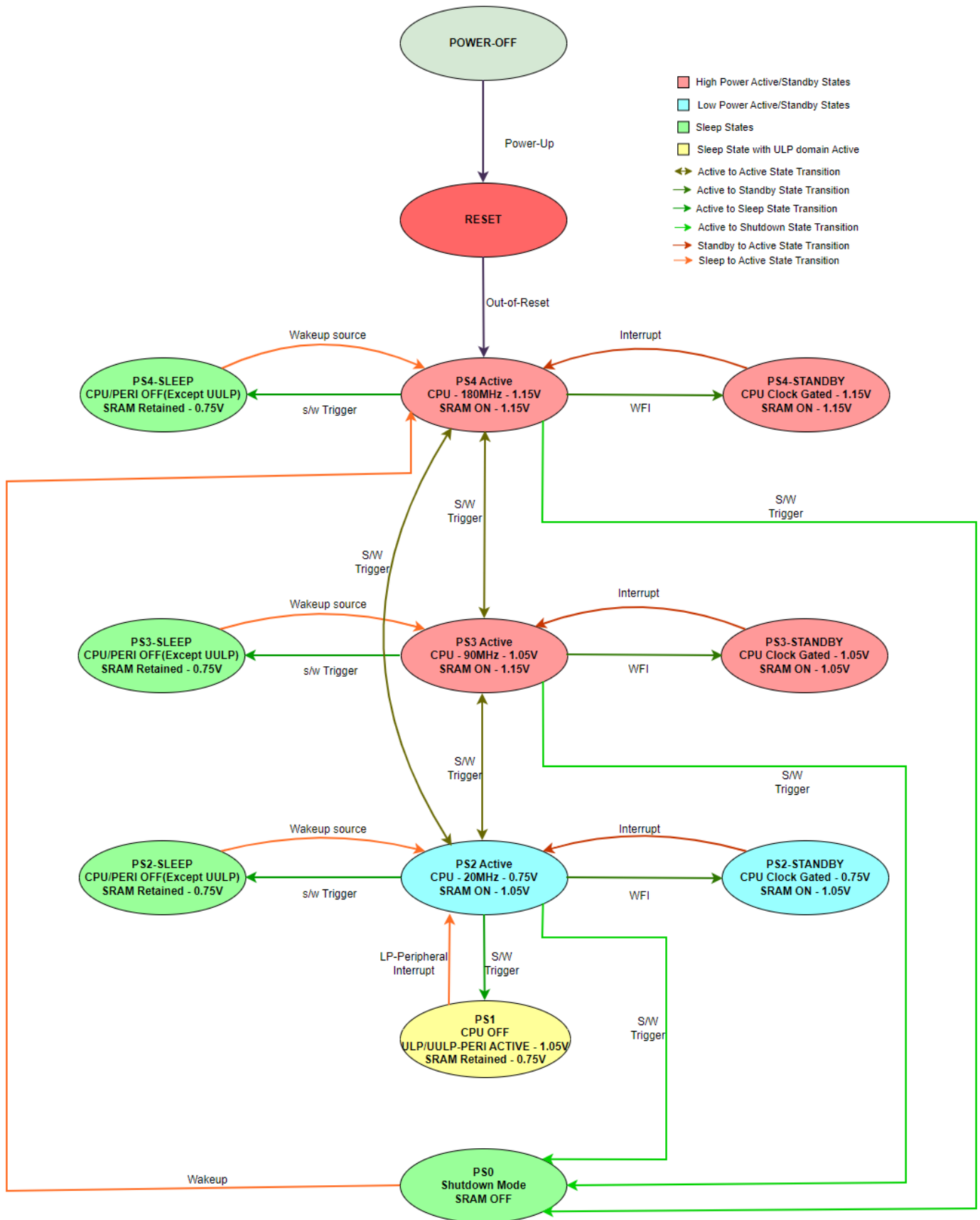
### 3.4 Deepsleep/Shutdown Mode

PS0, also known as deepsleep state, can be entered from any active state through software instruction.

The functional characteristics of this mode include:

- CPU power-gated
- UULP peripherals are functional and need to be configured before entering this state
- UULP-VBATT GPIOs are powered ON
- SRAM cannot be retained

The flowchart below gives an overview on the power state transitions, CPU, and SRAM configurations available in various power states.





A transition from active states to any other state can be triggered through software.

A transition from Standby/Sleep/Shutdown state can be triggered by an enabled interrupt as configured by software before entering the state.

#### 4. M4 Power Save Wakeup Sources

The table below indicates the wakeup sources available in Standby/Sleep/Shutdown states.

**Table 4.1. Standby/Sleep/Shutdown Wakeup Sources**

Wakeup Source	Wakeup Source, as present in the software	PS1	PS2- Stand-By	PS4- Sleep	PS3- Sleep	PS2- Sleep	PS0 (Deep sleep)
Deep-Sleep Timer (DST) Interrupt	DST_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
Wireless Processor Interrupt	WIRELESS_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
GPIO (UULP)	GPIO_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
Analog comparator Interrupt	COMPR_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
BOD							
SYSRTC Interrupt	SYSRTC_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
ULP Peripheral SDC (Sensor Data Collector)	SDCSS_BASED_WAKEUP	Yes	Yes				
Alarm Interrupt	ALARM_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
Second Interrupt	SEC_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
Milli-Second	MSEC_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
Watch Dog Interrupt	WDT_INTR_BASED_WAKEUP	Yes	Yes	Yes	Yes	Yes	Yes
ULP timer interrupt	ULPSS_BASED_WAKEUP	Yes	Yes				
ULP Touch Sensor Interrupt							
ULP GPIO Pin/ Group Interrupt							
ULP Peripheral DMA Interrupt							
ULP Peripheral ADC/DAC Interrupt							
ULP Peripheral UART Interrupt							
ULP Peripheral I2C Interrupt							
ULP Peripheral I2S Interrupt							
ULP Peripheral IR Interrupt							
ULP Peripheral SPI /SSI Master Interrupt							

**Note:** For more information on M4 Wakeup Sources, refer to the SiWx917 SoC Hardware Reference Manual.

## 5. NWP Power Save

The SiWG917 SoC NWP can be in any of the following power states:

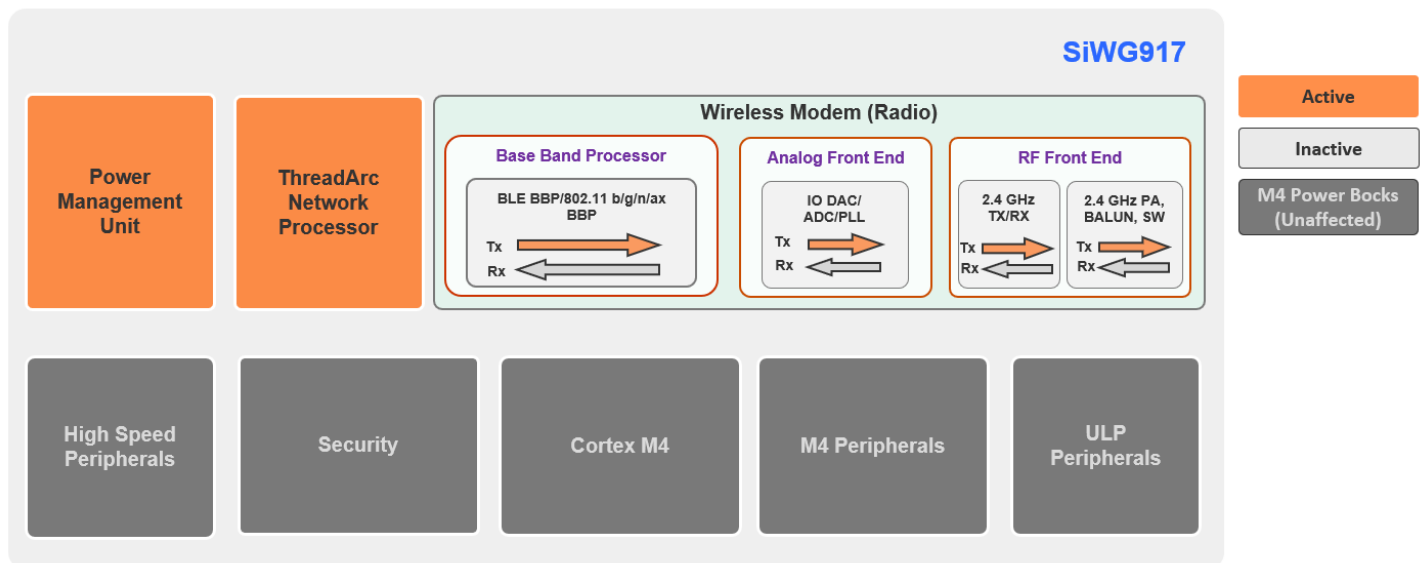
- Active or High-Performance State: All the power domains are powered ON and are active.
- Power Save State: Based on the power save mode used, certain power domains are active, certain domains are powered OFF, and certain domains are in sleep (consume low power).

### 5.1 Active or High-Performance State

The NWP can be in three operational modes in Active State as described below:

#### 5.1.1 Transmit Mode

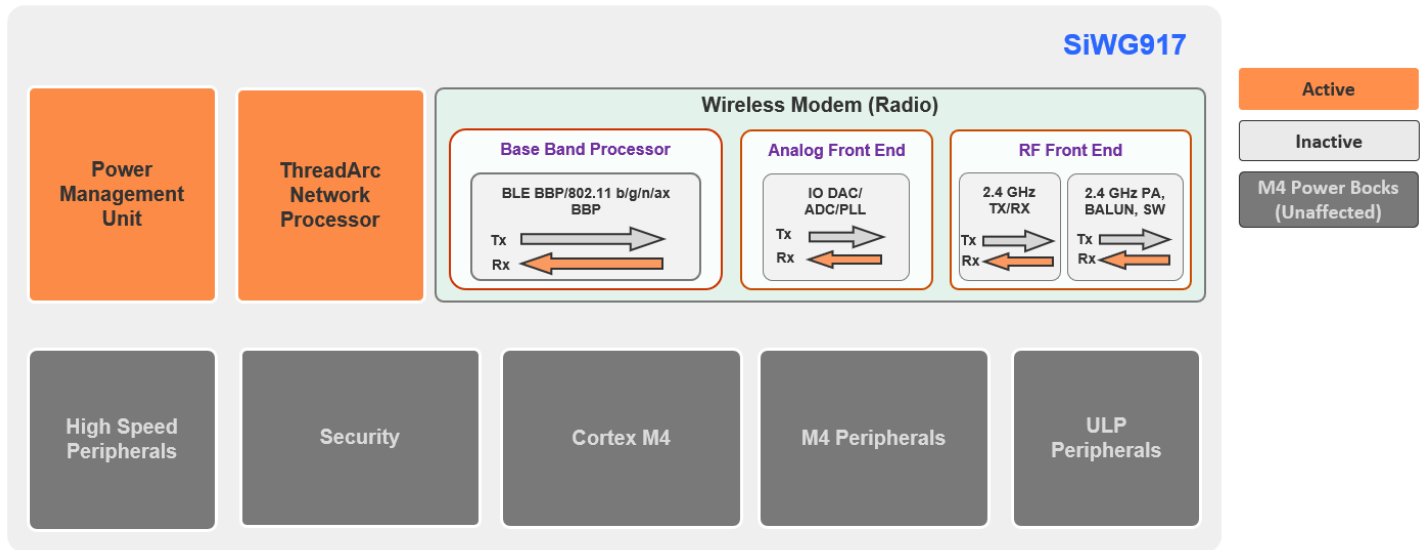
In the transmit mode, all the power domains of the NWP are active/operational except the receiver section of the Base Band Processor (BBP), Analog Front End (AFE), and RF Front End (RFFE). This is the highest power-consuming mode.



**Figure 5.1. Transmit Mode**

### 5.1.2 Receive Mode

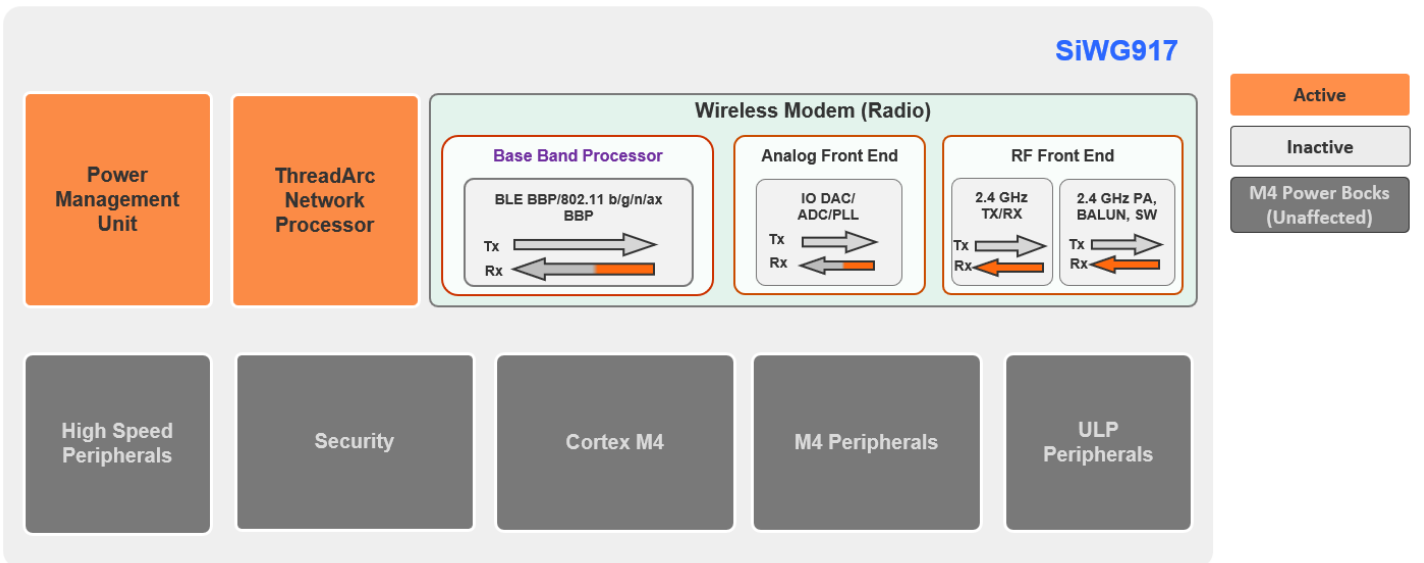
In the receive mode, the transmit sections of the BBP, AFE, and RFFE are inactive.



**Figure 5.2. Receive Mode**

### 5.1.3 Listen Mode

This mode is a subset of the receive mode where certain portions of the BBP and AFE are inactive as no packet reception is in progress.



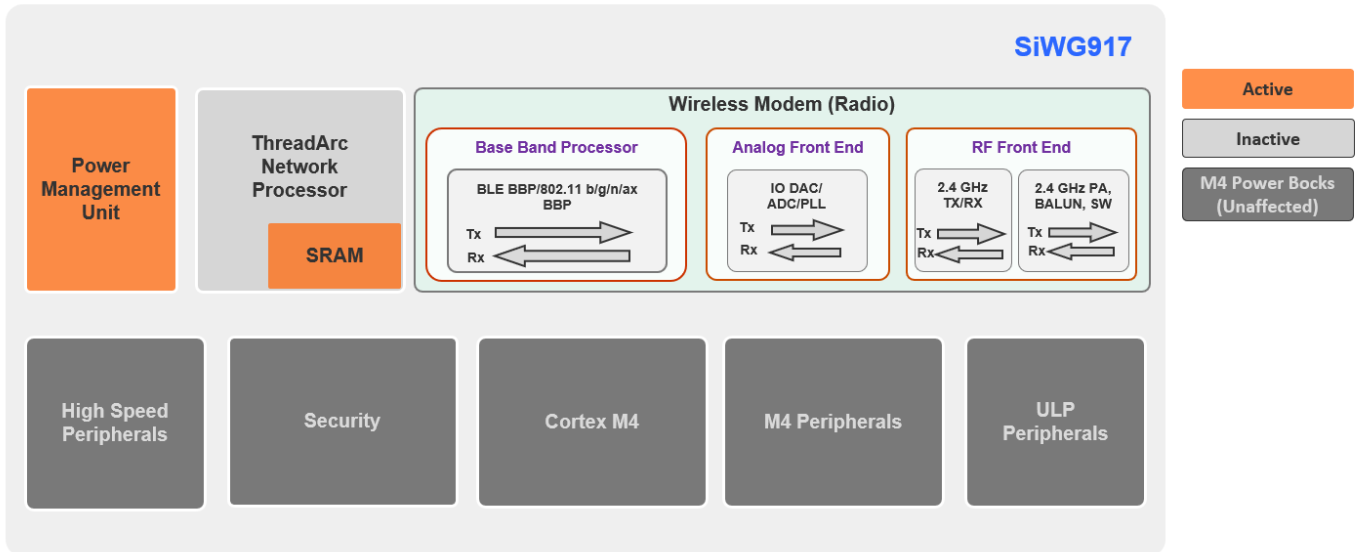
**Figure 5.3. Listen Mode**

## 5.2 Power Save State (Ultra-low-power Mode)

The Wireless Modem, NWP, and Security sections are inactive in the ultra-low-power mode. The Power Management Unit (PMU) has control over the other sections of the chip.

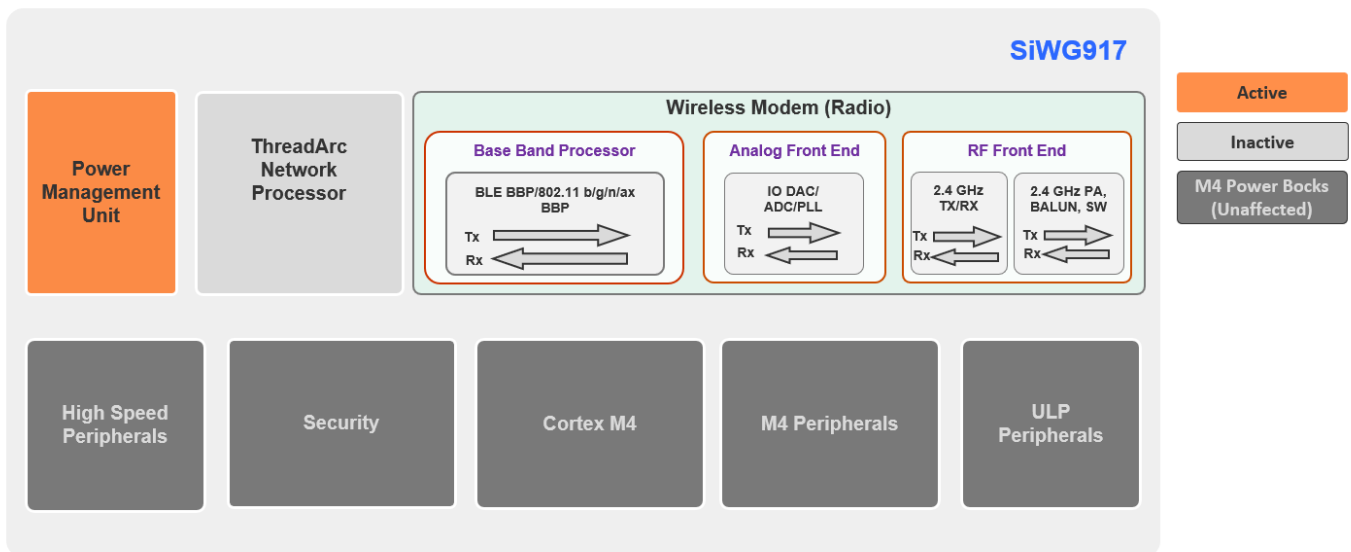
The ULP mode can be categorized into the following modes:

- With RAM Retention - the SiWx917 NWP RAM contents and current state are retained.



**Figure 5.4. NWP Sleep With RAM Retention Mode**

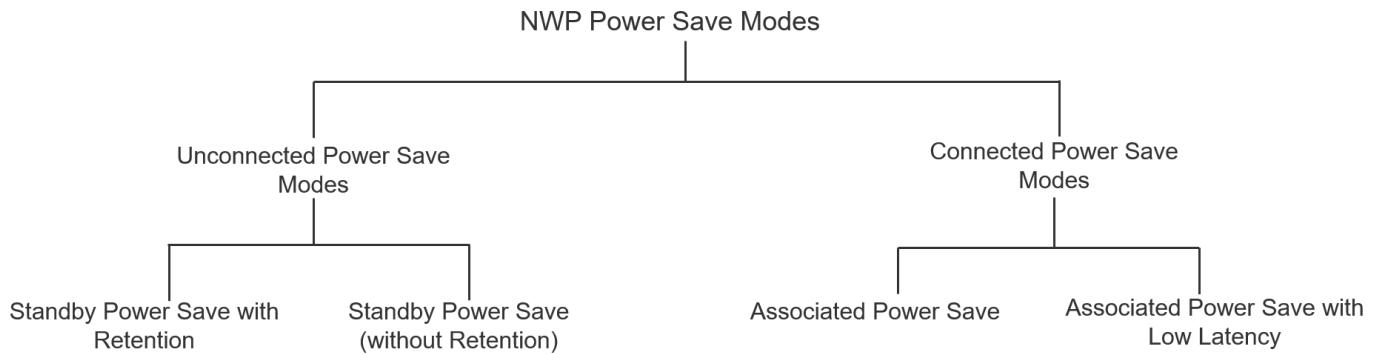
- Without RAM Retention - the SiWx917's RAM contents and current state are not retained.



**Figure 5.5. NWP Sleep Without RAM Retention Mode**

## 6. NWP Power Save Modes

The SiWG917 NWP can be set to Power Save state via any of the below Power Save Modes based on the application use case. The NWP Power Save Modes are broadly classified into the following:



**Figure 6.1. NWP Power Save Modes**

**Note:** For Connected Power Save, the RAM Retention configuration alone is supported, while for Unconnected Power Save, with and without RAM Retention configurations are supported.

The SiWG917 uses the following four flags as a handshake mechanism between M4 and NWP over the sleep-wakeup cycles:

1. TA\_wakeup\_M4: Set when NWP needs M4 to be awake. Cleared to indicate NWP allowing M4 to sleep.
2. TA\_is\_active: Set when NWP is in high-power mode. Cleared when NWP is going to sleep.
3. M4\_wakeup\_TA: Set when M4 needs NWP to be awake. Cleared to indicate M4 allowing NWP to sleep.
4. M4\_is\_active: Set when M4 is in high-power mode. Cleared when M4 is going to sleep.

**Note:** The above flags are modified by the internal driver APIs. The user must not edit any of them from the application layer.

### 6.1 Unconnected Power Save Modes

The NWP can be set into Unconnected Power Save before establishing a wireless connection.

### 6.1.1 Standby Power Save with RAM Retention

In Standby Power Save with RAM Retention mode, the NWP RAM contents and current state are retained. After wake up, the NWP can continue the execution.

#### Configuration:

1. After wireless initialization and before establishing a wireless connection, the NWP can be set into Standby Power Save with RAM Retention, by calling the `sl_wifi_set_performance_profile` API with the profile structure variable's member `sl_wifi_performance_profile_t.sl_performance_profile_t` to `STANDBY_POWER_SAVE_WITH_RAM_RETENTION`.
2. To switch the NWP from Standby Power Save with RAM Retention Mode to High Performance mode, call the `sl_wifi_set_performance_profile` API with the profile structure variable's member (`sl_wifi_performance_profile_t.sl_performance_profile_t`) to `HIGH_PERFORMANCE`.
3. After setting the NWP to High Performance mode, NWP need not be initialized again, as the previous state of the device is retained. The host can directly call the wireless connection APIs.

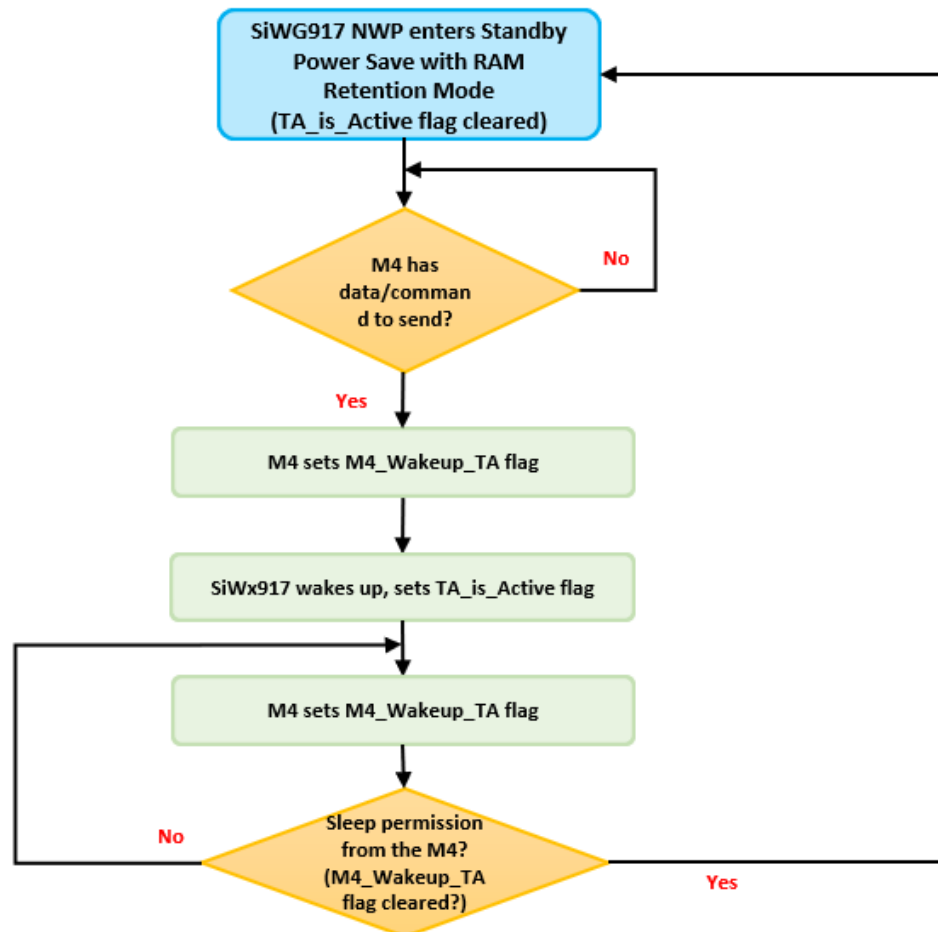


Figure 6.2. Sleep Wake-up Sequence of NWP during Standby/Unconnected Power Save with Retention

### 6.1.2 Standby Power Save (Without RAM Retention)

In Standby Power Save mode, the NWP RAM contents and current state are not retained. Upon wake up, the NWP needs to be re-initialized again.

#### Configuration:

1. After the wireless initialization and before establishing wireless connection, the NWP can be set into Standby Power Save (Without RAM Retention) by calling the `sl_wifi_set_performance_profile` API with the profile structure variable's member `sl_wifi_performance_profile_t.sl_performance_profile_t` to `STANDBY_POWER_SAVE`.
2. To switch the NWP from Standby Power Save to High-performance mode, the NWP needs to be initialized again, as the previous state of the NWP is not retained. Call the `sl_net_init()` API to initialize the NWP and bring it back to Active mode.

## 6.2 Connected Power Save Modes

NWP can be set into Connected Power Save after wireless connection when idle and can switch back to Active state for transmitting/receiving data to/from AP/M4.

Before going through the types of Connected Power Save Modes, it is important to understand the sleep/active state switching mechanism and wake interval concepts of NWP during Connected Power Save Mode.

### Sleep/Active State Switching

1. The connecting station (here SiWG917 NWP) makes an Association request to the AP. In the Association frame, it sends the Listen Interval which indicates how often the station wakes to listen to the AP beacon frames.
2. The AP shall respond with an Association Response frame in which it specifies the Association Identifier (AID) to its connecting station.
3. When the station is set in Power Save Mode by the M4 application, it sends a Null data frame to the AP with the Power Management (PWR MGT) bit set in the Frame Control field, indicating that it is entering the Sleep State and enters Sleep State. The AP acknowledges the Station's Null data frame.
4. When the NWP is in Connected Power Save Mode, it can be configured to wake up every Delivery Traffic Indication Message (DTIM) Interval or Beacon Interval (BI) or Listen Interval (LI) or Target Wake Time (TWT) Wake Interval during its Connected Power Save Mode.
  - **Beacon Interval-based wakeup:** Beacon Interval is the period between two subsequent beacon frames transmitted by AP. The station wakes every beacon interval.

**DTIM Interval-based wakeup:** DTIM period specifies how often an AP beacon includes Buffered Traffic Indication to its connected clients via TIM element in the beacon frame. When the AP includes TIM information in a beacon frame, the beacon is called DTIM beacon. DTIM interval is the time between two subsequent DTIM beacons transmitted by AP.  $DTIM\ Interval = Beacon\ Interval * DTIM\ Period$ .

**Listen Interval-based wakeup:** Based on the Listen Interval configured, the station wakes up at the nearest integral multiples of DTIM beacon/beacon interval broadcasted by the connected AP which is just less than or equal the Listen Interval.

**TWT Wake Interval-based wakeup:** The station wakes every TWT Wake Interval configured in the application. APs supporting Wi-Fi6 support TWT.

**Note:** Configuring larger listen intervals greater than 1000 milliseconds might lead to AP disconnecting the NWP. It is highly recommended to use 1000 ms as Listen Interval for low-power consumption.

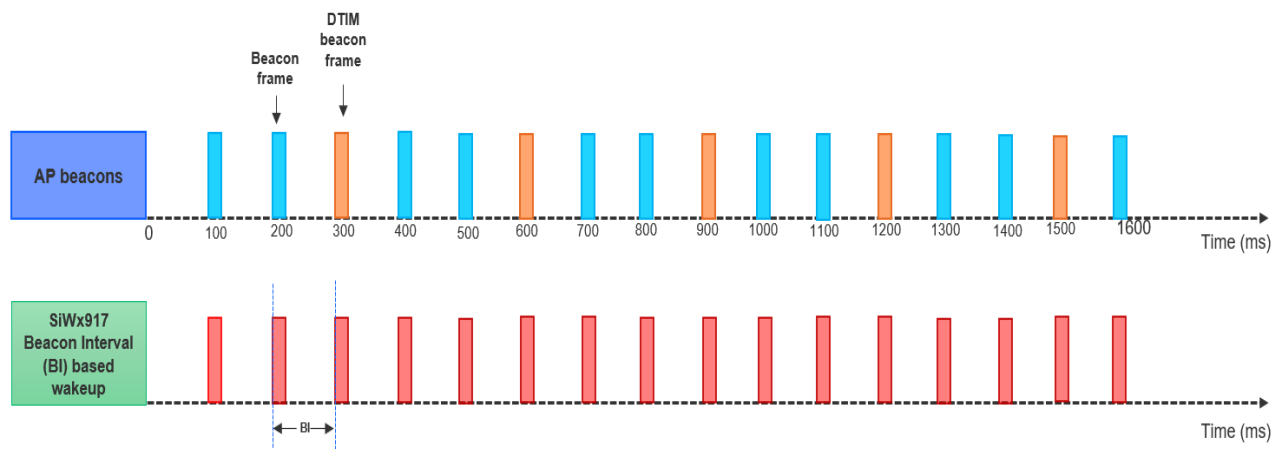
5. When its connected stations are asleep:
  - a. If the AP has any broadcast/multicast frames to transmit, it buffers them. It indicates this information to its connected stations in the immediate DTIM beacon with the Multicast field in the Traffic Indication Map (TIM) set to "True" and transmits the broadcast/multicast frames to all its connected stations.
  - b. If the AP has unicast data frames for a station, it buffers the frames. The AP informs the station about the buffered data via the Partial Virtual Bitmap (PVB) field in the TIM element in the immediate DTIM beacon. The PVB determines the list of AIDs set. For example, if a station is assigned with AID "3" during its association, whenever there are data packets buffered for this station, the AP sets bit "3" in the PVB.

**Note:** If the broadcast/multicast frames are not important for your application, call `sl_wifi_filter_broadcast()` API to ignore these frames before calling `sl_wifi_set_performance_profile()` API.
6. When the station wakes up to receive the beacon, it checks whether its AID is set in the Partial Virtual Bitmap (PVB) and if set, retrieves the buffered frames from the AP.
7. The process of retrieving the unicast data from AP when the SiWx91x is in sleep mode is different for different connected sleep modes. The Associated Power Save Mode uses Maximum Power Save Polling (Max PSP) Mechanism and Associated Power Save with Low Latency uses Fast Power Save Polling (Fast PSP) mechanism. The Max PSP and Fast PSP mechanisms shall be explained in detail in the upcoming sections of the document.
8. After receiving the data frames, the station goes back to sleep state.

### Beacon Interval

- The SiWG917 NWP wakes every Beacon Interval configured in the AP. The more the Beacon Interval, the less frequent the NWP wakes, thereby reducing the current consumption.
- The following figure illustrates the BI-based wakeup of NWP when AP BI = 100 ms and DTIM period = 3. In this case, the SiWx917's wake interval = 100 ms.



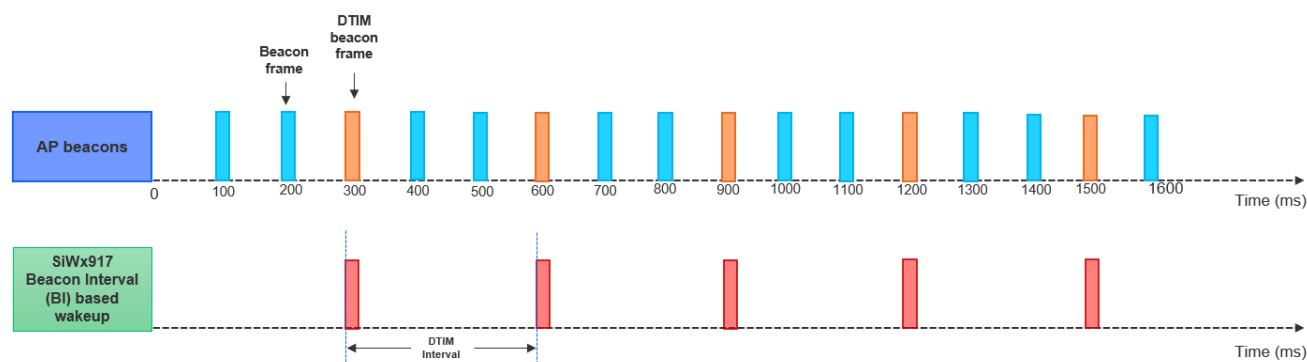


**Figure 6.3. BI-based Wakeup of NWP when AP BI = 100 ms and DTIM Period = 3**

- Configuration: Set the structure variable `sl_wifi_performance_profile_t.dtim_aligned_type` to `SL_SI91X_SLIGN_WITH_BEACON` while calling `sl_wifi_set_performance_profile()` API.

### DTIM Interval

- The SiWG917 NWP wakes every DTIM Interval, as per DTIM period configured in the AP. The less the DTIM interval, the less the RX latency to retrieve the buffered frames from AP.
- The following figure illustrates the DTIM Interval-based wakeup of NWP when AP BI = 100 ms and DTIM period = 3. In this case, the NWP's wake interval = 300 ms.



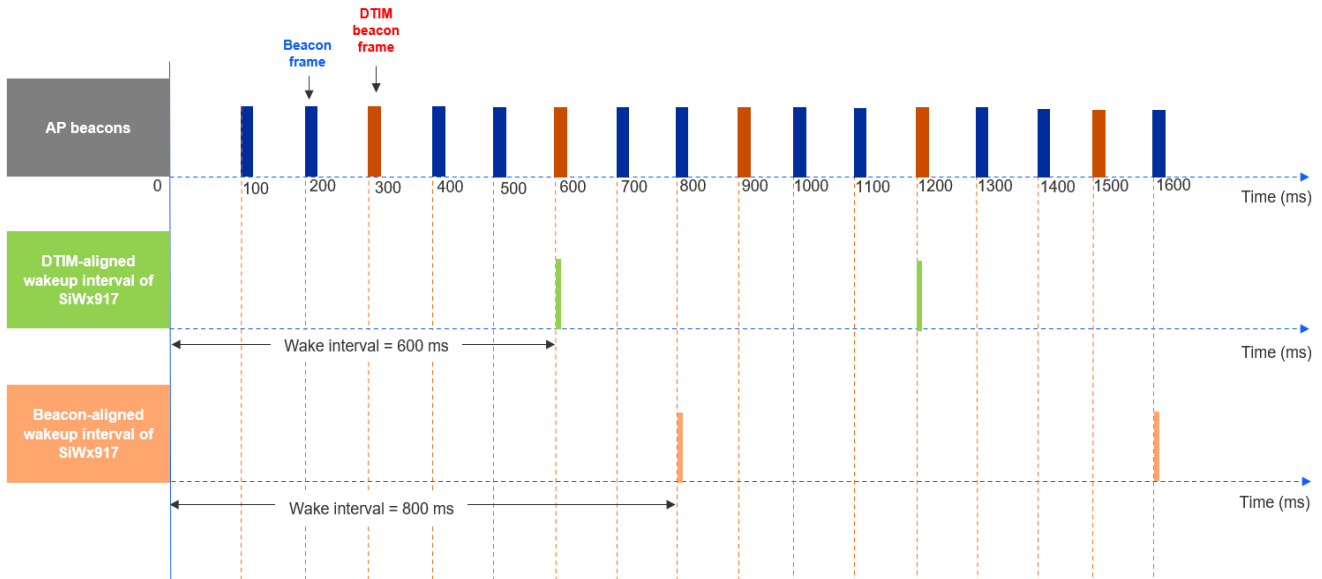
**Figure 6.4. DTIM Interval-based Wakeup of NWP when AP BI = 100 ms and DTIM Period = 3**

Configuration: Set the structure variable `sl_wifi_performance_profile_t.dtim_aligned_type` to `SL_SI91X_SLIGN_WITH_DTIM_BEACON` while calling `sl_wifi_set_performance_profile()` API.

### Listen Interval

- The Listen Interval indicates how often the SiWG917 NWP in Connected Power Save Mode wakes to listen to AP beacons.
- By default, the Listen Interval is set to 1000 ms in the WiSeConnect SDK v3.x.
- The Wake Interval of NWP can be aligned with the DTIM interval or Beacon Interval of the AP.
- For example, if LI is 1000 ms, the DTIM period is 3 and BI is 100 ms at AP (It means every third beacon contains DTIM information).
  - The NWP wakes every 900 ms ( $\leq$  Listen Interval) if configured to align with DTIM period of AP (DTIM Interval-based wakeup)
  - The NWP wakes every 1000 ms ( $\leq$  Listen Interval) if configured to align with BI of AP (Beacon Interval-based wakeup). In this case, the NWP does not take DTIM period of AP into consideration.

- Configuration: The Listen Interval can be configured in two methods:
  - During association with an AP
    - Set `listen_interval` to a desired value  $n_i$  multiples of DTIM period of AP and call the following API: `void sl_si91x_set_listen_interval(uint32_t listen_interval)`
    - Set `join_feature_bitmap` to `SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID` and call the following API before calling Wi-Fi connection APIs: `sl_status_t sl_si91x_set_join_configuration(sl_wifi_interface_t interface, uint8_t join_feature_bitmap)`
    - Call `sl_net_up()` API.
  - After establishing a Wi-Fi connection with the AP, pass the Listen Interval as an argument for power save API.
    - Set the `join_feature_bitmap` to `SI91X_JOIN_FEAT_PS_CMD_LISTEN_INTERVAL_VALID`.
    - Call the `sl_net_up()` for associating with an AP.
    - Call the following API with the profile structure variable's member (`sl_wifi_performance_profile_t.listen_interval`) set to a desired Listen Interval. This Listen Interval should be less than the Listen Interval set during association request: `sl_status_t sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile)`
- The following figure illustrates the LI-based wakeup of SiWx917 for AP's BI = 100 ms and DTIM period = 3, LI = 800 ms. In this case, the SiWx917's wake interval = 600 ms with DTIM-aligned configuration and 800 ms with Beacon-aligned configuration.



**Figure 6.5. LI-based Wakeup of SiWx917 for AP's BI = 100 ms and DTIM period = 3, LI = 800 ms**

**Note:** For Listen Interval-based wakeup, the broadcast and multicast frames transmitted by the AP when the device is asleep may be lost.

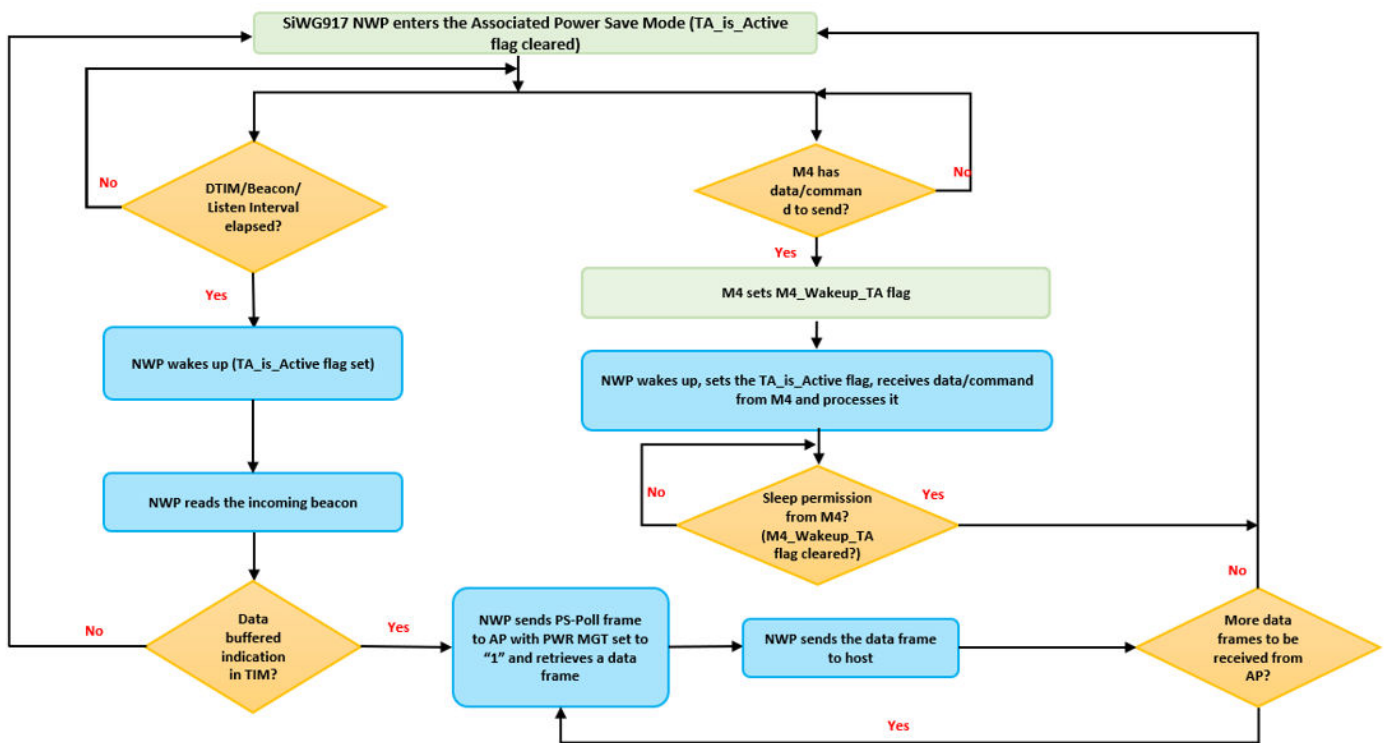
### 6.2.1 Associated Power Save Mode

The Associated Power Save Mode follows Max PSP mode.

**Communication between AP and SiWG917 NWP:** When in the Associated Power Save Mode, the NWP can send data to AP at any instance. For retrieving unicast data buffered at the AP, the following mechanism is used:

#### Maximum Power Save Polling (Max PSP):

1. Whenever the AP receives packets that are destined for a station (NWP), it buffers the frames.
2. The AP indicates this to the station by setting the corresponding station's AID in the TIM element of the next DTIM beacon.
3. On the next wake-up (based on DTIM or listen interval), the station receives the beacon and checks for the TIM.
4. If the AID of the station is set, it sends a Power Save Polling (PS-Poll) frame with its AID, to the AP, to retrieve the data frame.
5. The AP acknowledges the PS-Poll frame and transmits a data frame with the "More Data" field set to 1 in case there are more data frames buffered for the station.
6. After receiving the data frame, the NWP sends it to M4.
7. NWP sends a PS-Poll frame to retrieve each data frame from the AP.
8. While sending the last data frame to the station, the AP shall set the "More Data" field to 0.
9. After receiving the last data frame, the NWP goes into the sleep state.



**Figure 6.6. Sleep Wake-up Sequence of NWP during Associated Power Save (Max PSP)**

The Max PSP saves more power but produces lower throughputs. Sending a PS-Poll frame to retrieve each packet affects the throughput as it induces a considerable amount of delay when bulk data is to be retrieved.

#### Configuration:

- After a wireless connection, the NWP can be set into Associated Power Save by calling the following API with the profile structure variable's member (sl\_wifi\_performance\_profile\_t.sl\_performance\_profile\_t) set to ASSOCIATED\_POWER\_SAVE: `sl_status_t sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile)`.
- The Listen Interval can be configured as described at the Sleep/Wake State Switching subsection at the beginning of the Connected Power Save Modes section.

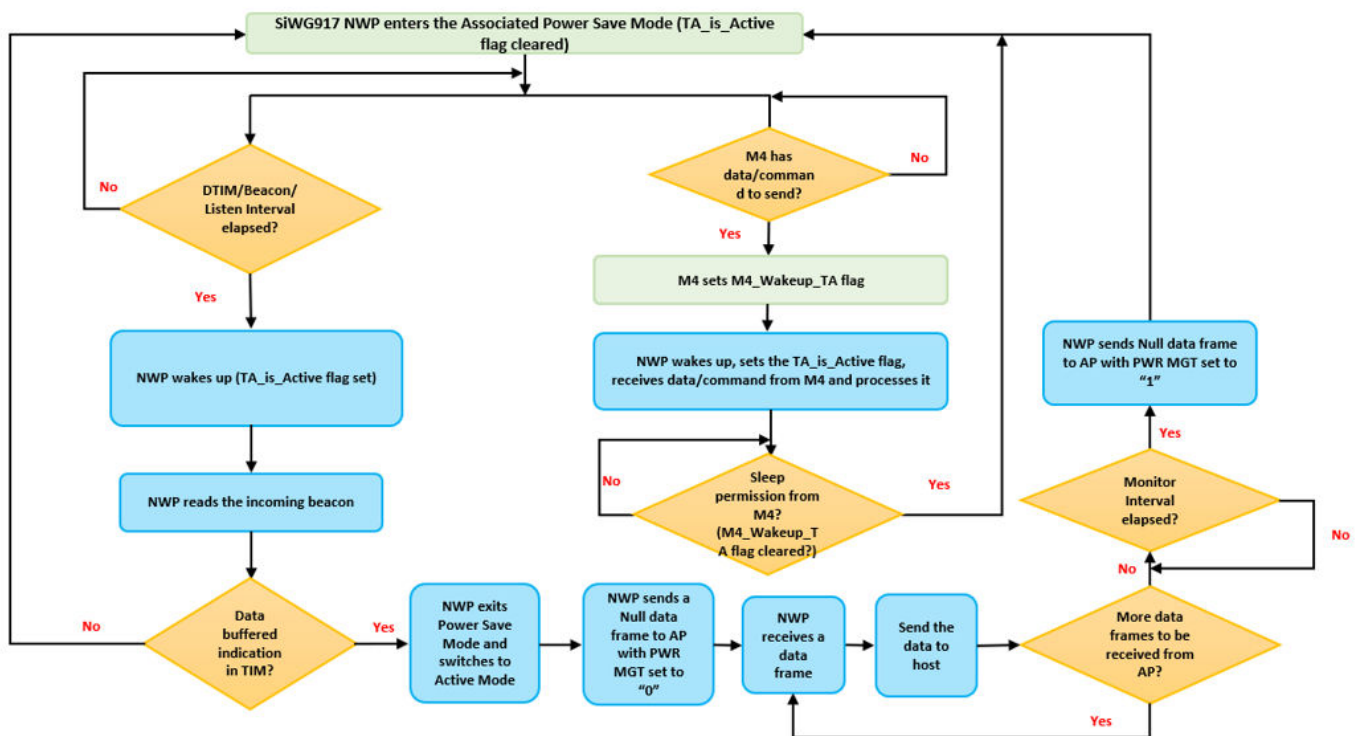
### 6.2.2 Associated Power Save with Low Latency

The Associated Power Save with Low Latency Mode follows Fast PSP mode.

**Communication between AP and NWP:** When in the Associated Power Save Mode, the station can send data to AP at any instance. For retrieving unicast data buffered at the AP, the following mechanism is used.

#### Fast Power Save Polling (Fast PSP):

1. Whenever the AP receives data frames that are destined for a Station (NWP), it buffers the packets.
2. The AP indicates this to the Station by setting the corresponding AID in the TIM element of the next beacon.
3. On the next wake-up (based on DTIM or listen interval), the Station receives the beacon and checks for the TIM.
4. If the AID of the Station is set, the Station exits power save mode, switches to active mode, and sends a Null data frame with PWR MGT bit set to "0" to the AP to retrieve all the data frames from AP.
5. The AP acknowledges the Null data frame and it AP transmits a data packet with the "More Data" field set to 1 in case there are more data packets buffered for the Station.
6. After receiving a data packet, the NWP Station sends it to the M4.



**Figure 6.7. Sleep Wake-up Sequence of NWP during Associated Power Save with Low Latency (Fast PSP)**

If the Station does not receive any data for the monitor interval of time configured, the Station goes back to sleep by sending a Null data frame with the PWR MGT bit set to "1" to the AP.

#### Configuration:

1. After a wireless connection, the NWP can be set into Associated Power Save by calling the following API with the profile structure variable's member (sl\_wifi\_performance\_profile\_t.sl\_performance\_profile\_t) set to ASSOCIATED\_POWER\_SAVE.
2. `sl_status_t sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile)`
3. The monitor interval can be configured by defining the profile structure variable's member (sl\_wifi\_performance\_profile\_t.monitor\_interval).
4. The listen interval can be configured as described at the Sleep/Wake State Switching subsection at the beginning of the Connected Power Save Modes section.

**Enhanced Max PSP Feature:**

In Associated Power Save Mode, during unicast data retrieval, some Access Points do not acknowledge the PS-Poll frames and do not deliver buffered data destined for the Station. This interoperability issue can be avoided with the Enhanced Max PSP feature.

**Configuration:**

1. Enable the `SL_SI91X_ENABLE_ENHANCED_MAX_PSP` in `config_feature_bit_map`.
2. Set the Performance Profile to `ASSOCIATED_POWER_SAVE_LOW_LATENCY`.

**Functionality and usage:**

1. Initially, the SiWx91x's Power Save Mode should be set to is `ASSOCIATED_POWER_SAVE`.
2. After sending a PS-Poll frame to AP, if the AP delivers the buffered data within 20 ms, the SiWx91x remains to be in `ASSOCIATED_POWER_SAVE`.
3. When the AP does acknowledge PS-Poll and does not deliver the buffered data within 20 ms, the Power Save Mode is switched to `ASSOCIATED_POWER_SAVE_LOW_LATENCY`.

**Note:** To switch the SiWx91x from connected power save to active mode, call the [sl\\_wifi\\_set\\_performance\\_profile](#) API with the profile structure variable's member (`sl_wifi_performance_profile_t.sl_performance_profile_t`) set to `HIGH_PERFORMANCE`.

### 6.2.3 Target Wake Time (TWT)

TWT is a beneficial Wi-Fi6 feature which allows connected stations to manage power efficiency with the reduced network contention.

**Reduced network contention:** In the Legacy Power Save Modes, the Wi-Fi stations go to sleep and wakeup at random times to perform data transfer unrelated to the wakeup timings of other stations. With TWT, the AP schedules wake timings for its connected stations, ensuring no two Wi-Fi stations wake up at the same time. This method helps avoid packet collisions, thus reducing retransmissions and in turn reducing the station's current consumption.

Along with this, TWT allows the Wi-Fi stations to be in sleep for longer durations.

There are two types of TWT:

- Individual TWT: The connected stations negotiate the TWT session parameters (TWT Wake Interval and Wake duration) independently with the AP.
- Broadcast TWT: The AP broadcasts predefined TWT parameters (TWT Wake Interval and Wake duration) in its beacon and schedules wake times for different connected stations.

The Wake duration, Wake Interval, and sleep duration are derived from TWT parameters.

- Wake duration: The duration for which the station is awake to transmit/receive data frames to/from the AP. This duration is also called as TWT Service Period (SP).
- Wake Interval: The time interval between two successive SPs' start times.
- Sleep duration: The time between end of current SP and start of next/future SP.

#### Configuration:

- Configure the TWT parameters using the `sl_wifi_performance_profile_t.sl_wifi_twt_request_t` structure.

```
typedef struct {
    uint8_t wake_duration;
    uint8_t wake_duration_tol;
    uint8_t wake_int_exp;
    uint8_t wake_int_exp_tol;
    uint16_t wake_int_mantissa;
    uint16_t wake_int_mantissa_tol;
    uint8_t implicit_twt;
    uint8_t un_announced_twt;
    uint8_t triggered_twt;
    uint8_t negotiation_type;
    uint8_t twt_channel;
    uint8_t twt_protection;
    uint8_t twt_flow_id;
    uint8_t restrict_tx_outside_tsp;
    uint8_t twt_retry_limit;
    uint8_t twt_retry_interval;
    uint8_t req_type;
    uint8_t twt_enable;
    uint8_t wake_duration_unit;
} sl_wifi_twt_request_t;
```

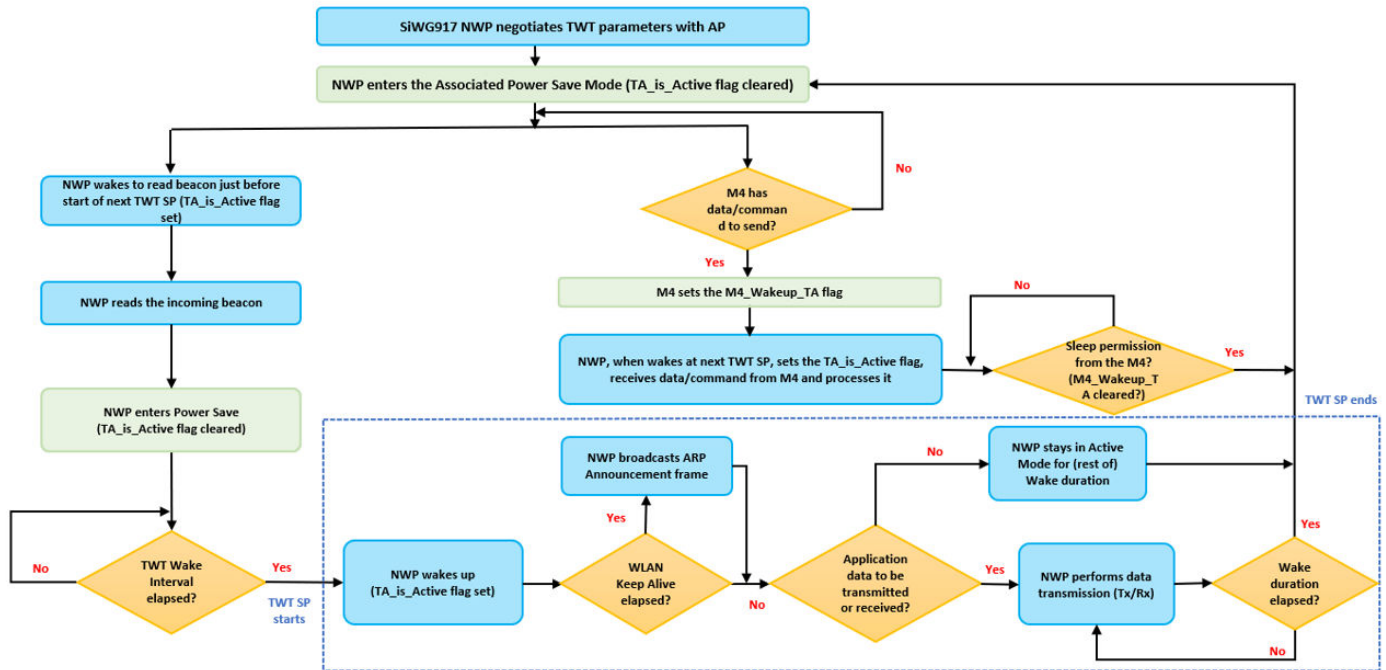
**Note:** For more information about the above structure and its configuration, refer to [TWT TCP Client](#) example in the WiSeConnect SDK v3.x.

- As per above structure parameters, below are the derivations:
  - The Wake duration or TWT Service Period is calculated as `wake_duration*wake_duration_unit`
  - The TWT Wake Interval is calculated as  $(\text{wake\_int\_mantissa}) \times (2^{\text{wake\_int\_exp}})$
  - The TWT Sleep duration is calculated as TWT Wake Interval-Wake duration
- Define a `sl_wifi_performance_profile_t.sl_wifi_twt_request_t` structure variable and pass it as an argument to the following API: `sl_status_t sl_wifi_enable_target_wake_time(sl_wifi_twt_request_t *twt_req)`.

### 6.2.4 TWT-based Power Save Mechanism

This section explains the TWT Power Saving Mechanism of SiWG917 NWP with the following TWT parameters set:

- Individual TWT- The connected station (NWP) indicates a negotiation with its individual TWT Wake Interval and Wake duration.
- Implicit TWT- The TWT requesting station (NWP) calculated the next TWT by adding a fixed value to the current TWT value.
- Unannounced TWT- The TWT requesting STA does not announce its wake up to AP through PS-POLLS or UAPSD Trigger frames.
- Non-triggered TWT- The TWT SP does not contain any triggered frames.



**Figure 6.8. Sleep Wake-up Sequence of NWP during Associated Power Save with Legacy TWT**

**Communication between AP and NWP:** When in the Associated Power Save Mode with TWT, the station can send or receive data to/from AP only during the `tw_twake_duration`.

1. The SiWG917 NWP sends an Association frame with TWT Requester Support bit set in the High Efficiency (HE) MAC capabilities field.
2. After connecting to an Access Point, the NWP transmits a TWT Setup frame with the TWT parameters depending on the request type (Request TWT/Suggest TWT/Demand TWT).
3. If the AP agrees on the TWT parameters, it transmits a TWT Accept frame.
4. The NWP sends a Null data frame and goes to sleep.
5. Whenever the AP receives destined packets for a station (NWP), it buffers them.
6. The AP indicates this to the station by setting the corresponding AID in the TIM element of the next beacon.
7. The NWP wakes just before its TWT to read the incoming beacon from the AP for Time Synchronization.
8. As soon as it reads the beacon, the NWP goes back to sleep and wakes at start of TWT SP.
9. During its Wake duration/TWT SP, if the AID of NWP is set in the TIM element, the NWP exits power save mode, switches to active mode, and retrieves all the data packets from AP during TWT SP.
10. The AP transmits a data frame with the "More Data" field set to 1 in case there are more data packets buffered for the Station.
11. After receiving a data packet, the NWP sends it to the host.
12. In case there are no data frames to be received, the station will be active for the rest of wake duration (if remains) and goes back to Sleep state.
13. After TWT SP, if there are more data frames to be transmitted or received from the AP, the device still goes to sleep and shall carry out these data transfers at its next TWT SP.

TWT Mechanism can be used when your M4 application has predictable and deterministic data traffic. The M4 application should be confident about the times at which data traffic is expected should be known prior.

**Configuration:**

- After calling `sl_net_up()` API, register a callback function for TWT negotiation response events using the following API: **static inline** `sl_status_t sl_wifi_set_twt_config_callback` (`sl_wifi_twt_config_callback_t` function, **void** \*optional\_arg).
- Trigger the TWT parameters negotiation by calling the below API with the profile structure variable member `sl_wifi_performance_profile_t.sl_wifi_twt_request_t` set with TWT setup configuration: `sl_status_t sl_wifi_enable_target_wake_time` (`sl_wifi_twt_request_t *twt_req`).
- Once the AP sends a TWT response frame, the registered callback function gets triggered. The callback function gives the agreed TWT parameters during TWT parameters negotiation.
- Next, set the SiWx917 into Associated Power Save (or with Low Latency) by calling the following API with the profile structure variable's member (`sl_wifi_performance_profile_t.sl_performance_profile_t`) set to `ASSOCIATED_POWER_SAVE` (or `ASSOCIATED_POWER_SAVE_LOW_LATENCY`): `sl_status_t sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile)`



## 6.2.5 Automatic TWT

Automatic TWT (Auto TWT) is a Silicon Labs implementation that configures TWT parameters automatically based on the user application requirements. The user can provide the application requirements in terms of average Throughput and RX latency. Based on these inputs, the SiWG917 NWP automatically configures the TWT parameters and negotiates them with the AP.

### Working and Configuration:

- After calling `sl_net_up()` API, register a callback function for TWT negotiation response events using the following API: static inline `sl_status_t sl_wifi_set_twt_config_callback(sl_wifi_twt_config_callback_t function, void *optional_arg)`
- The Auto TWT selection parameters are configured using the `sl_wifi_performance_profile_t.sl_wifi_twt_selection_t` structure.

```
typedef struct {
    uint8_t twt_enable;
    uint16_t average_tx_throughput;
    uint32_t tx_latency;
    uint32_t rx_latency;
    uint16_t device_average_throughput;
    uint8_t estimated_extra_wake_duration_percent;
    uint8_t twt_tolerable_deviation;
    uint32_t default_wake_interval_ms;
    uint32_t default_minimum_wake_duration_ms;
    uint8_t beacon_wake_up_count_after_sp;
} sl_wifi_twt_selection_t;
```

- From the above structure, the configurable values are:
  - `twt_enable` : 1- Setup ; 0 - teardown
  - `rx_latency` : The maximum allowed receive latency, in milliseconds, when an RX packet is buffered at the AP. If `rx_latency` is less than  $\leq 2$  sec, session creation is not possible. If configured as 0, then by default 2 sec is considered.
  - `avg_tx_throughput` : The expected average TX throughput in Kbps should be between 0 and half of device average throughput.
- Define a `sl_wifi_performance_profile_t.sl_wifi_twt_selection_t` structure variable and pass it as an argument to the following API: `sl_status_t sl_wifi_target_wake_time_auto_selection(sl_wifi_twt_selection_t *twt_auto_request)`
- When the above API is called, the SiWx917 automatically configures the TWT parameters and triggers a TWT negotiation with the AP.
- Once the AP sends a TWT response frame, the registered callback function gets triggered. The callback function gives the agreed TWT parameters during TWT parameters negotiation.
- Next, set the SiWx917 into Associated Power Save (or with Low Latency) by calling the following API with the profile structure variable's member (`sl_wifi_performance_profile_t.sl_performance_profile_t`) set to `ASSOCIATED_POWER_SAVE` (or `ASSOCIATED_POWER_SAVE_LOW_LATENCY`). `sl_status_t sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile)`
- The SiWx917 performs TX/RX based on the user configured parameters ensuring low latency for transmitting and receiving data.

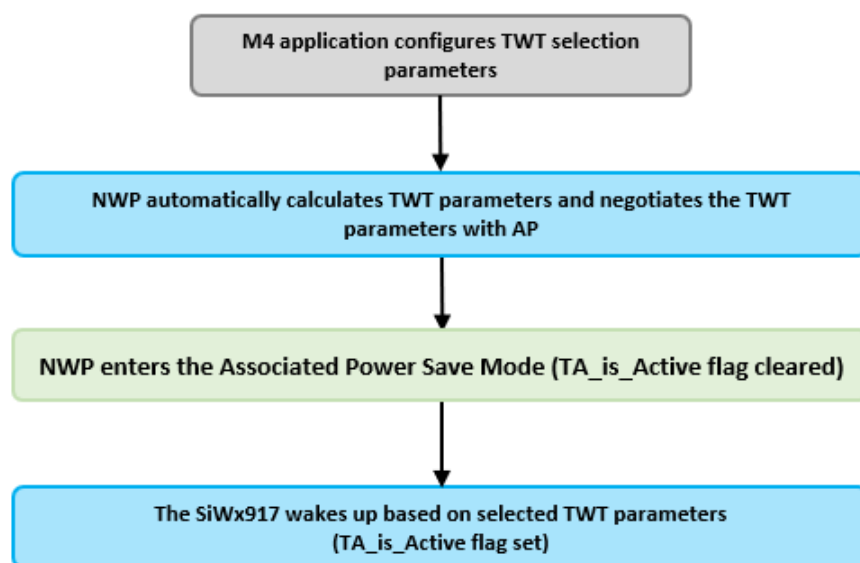


Figure 6.9. Sleep Wake-up Sequence of NWP during Auto TWT

### **Benefits of Automatic TWT over TWT**

- Power optimization when compared to standard TWT
- Handles TX and RX activities based on latency that the application can withhold, which is not possible with standard TWT.

## 7. SiWG917 Typical Current Consumption

This section lists the typical current consumption of SiWG917 observed at various stages of operation such as during Wireless connection and data transfer, either of NWP or M4 or both being in low-power state etc., along with the reference examples and setup used to measure the current consumption.

**Note:** The current consumption values present in this document are taken during the first version.

### 7.1 Reference Examples

The following are the reference examples in the [WiSeConnect 3 SDK](#) for configuring Power Save Modes:

- For M4 active, NWP in unconnected power save modes configuration, refer to [Power Save Deep Sleep](#) example.
- For M4 in sleep with retention, NWP in Connected Power Save Modes configuration, refer to [Power Save Standby Associated](#) example.
- For M4 sleep with retention, NWP in Connected Power Save Mode with TWT, refer to [TWT TCP Client](#) example.
- For M4 in low-power active (PS2), NWP in shutdown, refer to [ULP GPIO](#) example.
- To analyze M4 Power State transitions with NWP in Unconnected Power Save Mode, refer to the [Power Manager](#) example.

### 7.2 Setup

#### 7.2.1 Hardware

- SiWG917 Evaluation Kit (BRD4338A SoC Radio Board + BRD4002A Wireless Pro Kit Mainboard)
- A PC to install and run Simplicity Studio
- Access Point (Used only in case of Connected Power Save Modes) (11ax TWT supported AP is needed to evaluate TWT current consumption)

#### 7.2.2 Software

- Simplicity Studio IDE
- [WiSeConnect 3 SDK](#)

**Note:** The current consumption details present in this section are measured using the [Energy Profiler](#) tool from Simplicity Studio IDE.

### 7.2.3 Setup Diagram

For measuring current, the following setup is used:

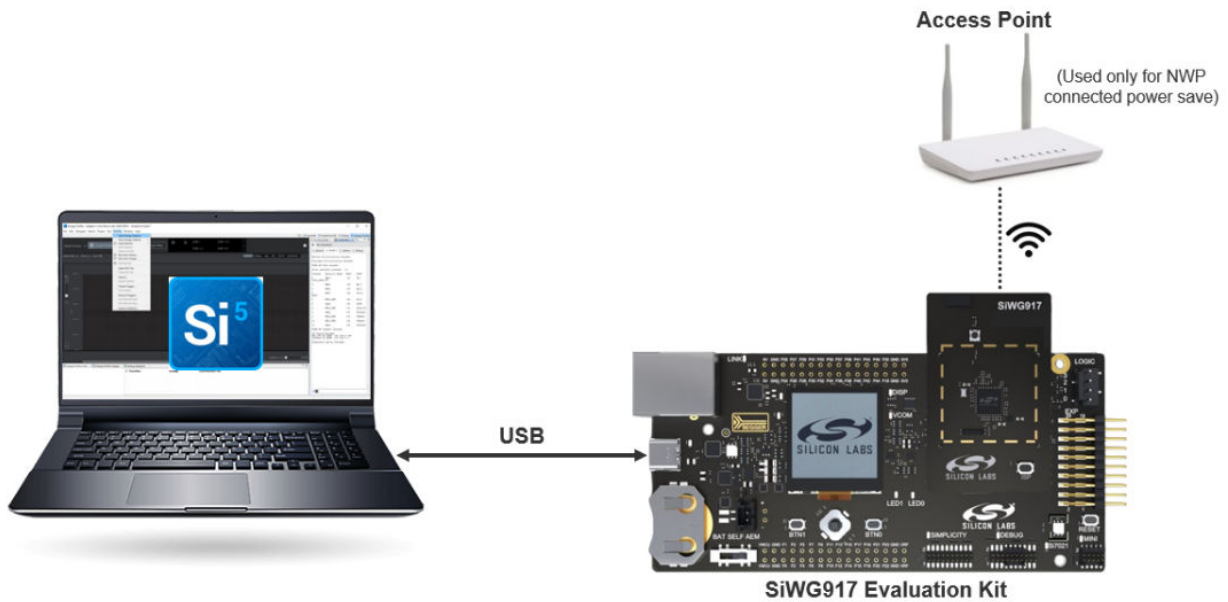


Figure 7.1. Setup Diagram

### 7.3 SiWG917 (M4+NWP) Power Consumption

This section describes the current consumption values at different stages of SoC.

**Note:**

1. All the current consumption values mentioned in this section are measured with 352-320KB MEMORY configuration for NWP and M4 respectively.
2. A variation of current consumption by 10% is expected from board to board.
3. For the latest/updated current consumption numbers, refer to the SiWG917 data sheet.

### 7.3.1 SiWG917 Comes Out of RESET (M4 and NWP Initialization)



Figure 7.2. SiWG917 Current Consumption during Initialization

Table 7.1. SiWG917 Average Current Consumption during Initialization

State	Average Current Consumption	Time Taken
SoC power-up to firmware load	15.93 mA	1.50 s
SoC power-up to radio initialization	15.94 mA	1.54 s

### 7.3.2 SiWG917 during WLAN Connection

By default, the applications are configured to scan on 2.4 GHz channels (1-11) for the specified SSID. The SoC sends a directed Probe Request specifying the SSID it is looking for on the 2.4 GHz channels.

By default, the SoC sends a unicast probe request, gets a probe response, and then connects to the AP.

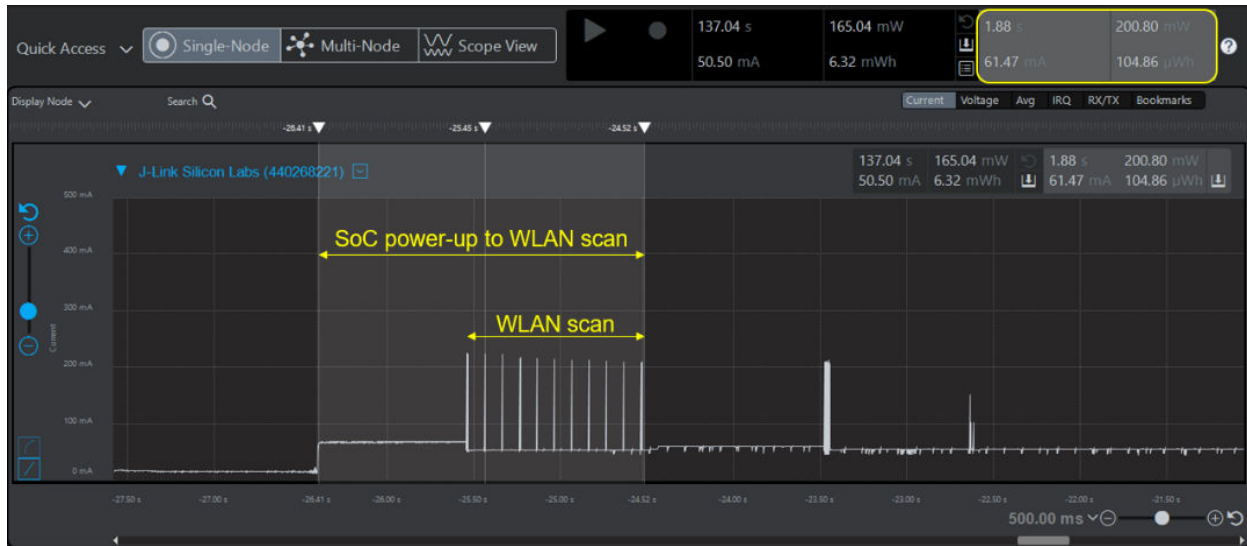


Figure 7.3. SiWG917 NWP during WLAN Connection

Table 7.2. SiWG917 Average Current Consumption until Wireless Connection

State	Average Current Consumption	Time Taken
Radio initialization to wireless scan	61.5 mA	1.88 s
WLAN scan	56.5 mA	1.01 s
WLAN connection	59.8 mA	1.08 s
SoC power-up to IP configuration (Using DHCP)	47 mA	5.33 s

**Notes:**

1. Quick scan feature in the SoC enables the NWP to scan for a particular access point in a particular channel. This feature benefits you if your application connects to a known SSID on a specific channel number. This helps in reducing the time taken for scanning for APs as well as lowers the current consumption.
2. The dwell time of the scan in each channel can be configured as needed to reduce current consumption further.
3. Quick join feature in the SoC enables the NWP to send authentication and association frames without unicast probe requests. To use the Quick Join feature, enable SI91X\_JOIN\_FEAT\_QUICK\_JOIN feature in the **join\_feature\_bitmap** and call the below API.

### 7.3.3 SiWG917 during WLAN Data Transfer

The NWP can be in the following atomic states:

- TX\_ACTIVE: Current consumption during active transmission at a given on-air data rate such as 6 Mbps, and a given output power level
- RX\_ACTIVE: Current consumption during active reception at a given on-air data rate

The above current consumption values are mentioned in the WLAN 2 GHz section of the SiWG917 data sheet.

### 7.3.4 M4 PS4 Active, NWP in Associated Power Save

M4 being in PS4 active state, NWP is configured to ASSOCIATED\_PWSAVE after WLAN connection.

**Note:** The measurement is taken in Listen Interval-based power save mode.

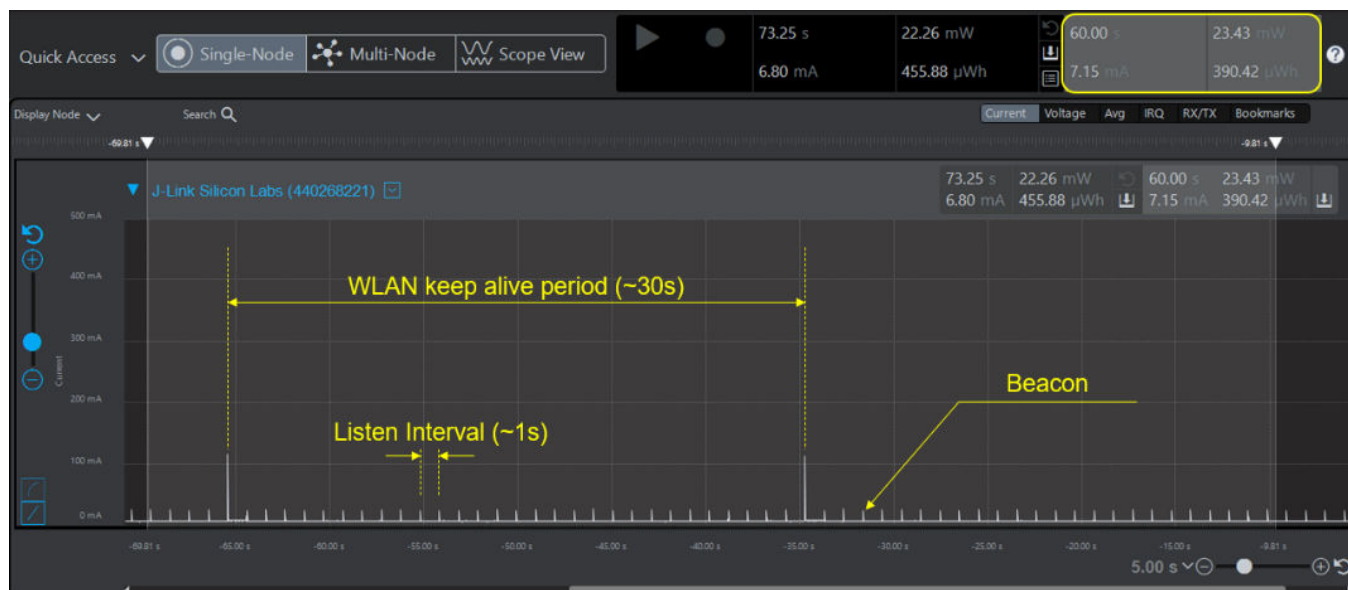


Figure 7.4. NWP in Associated Power Save with 1 s Listen Interval

Table 7.3. Current Consumption with NWP in Connected Sleep

	Sleep Current (Between Beacons)	Beacon to Beacon Current	Average Current Consumption
Associated power save	7.07 mA	7.12 mA	7.15 mA

### 7.3.5 M4 Active, NWP in Standby Power Save

M4 being in PS4 active, NWP is configured to STANDBY\_PWSAVE/\_WITH\_RETENTION after WLAN initialization.

Table 7.4. Current Consumption with NWP in Unconnected Sleep

	Sleep Current	Units
Standby power save with retention	8.53	mA
Standby power save without retention	8.52	mA

### 7.3.6 M4 in PS4 Sleep with Retention, NWP in Associated Power Save with Retention

**Note:** The measurement is taken in Listen Interval-based power save mode.



Figure 7.5. M4 in PS4 Sleep, NWP in Associated Power Save

### 7.3.7 M4 in Sleep with Retention, NWP in Associated Power Save with TWT Enabled

By default, the TWT parameters configured in TWT TCP client example brings in a TWT Wake Interval of 61.44 seconds, Wake duration of 24.576 ms.

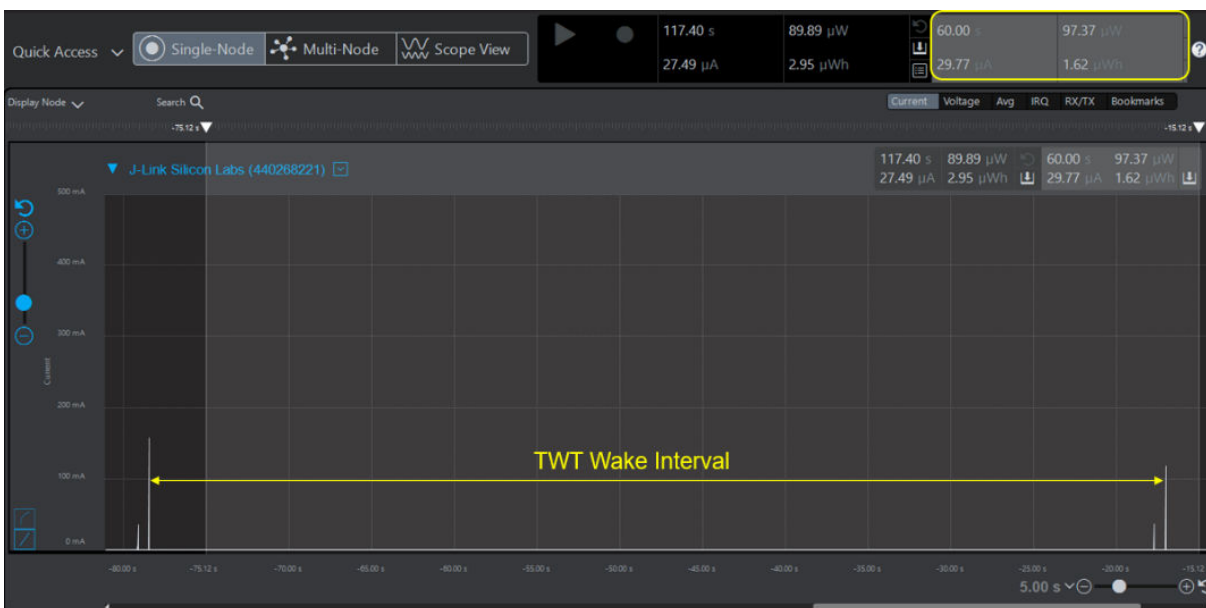


Figure 7.6. M4 in PS4 Sleep, NWP in Associated Power Save with TWT

The above current consumption values are mentioned in the WLAN 2 GHz section of the SiWG917x data sheet.

**Note:** Current consumption for both M4 and NWP in sleep without RAM retention can also be referenced in the data sheet.



## 7.4 M4 Power States Active and Sleep Currents (NWP Shutdown)

The following current consumption details are obtained by configuring NWP in shutdown that consumes minimal current consumption.

**Table 7.5. SiWG917 M4 Power States Current Consumption**

Power State	Mode	Current Consumption	Units
PS4	Active	8.6	mA
	Sleep with 320 kB RAM retention	10.3	μA
	Sleep with 192 kB RAM retention	6.9	μA
	Sleep with 64 kB RAM retention	5	μA
	Sleep with 16 kB RAM retention	2.6	μA
PS3	Active	7	mA
	Sleep current	Same as PS4 Sleep	—
PS2	Active with 320 kB RAM	741	μA
	Active with 64 kB RAM	660	μA
	Sleep with 64 kB RAM retention	5.7	μA
PS1	Active with 320 kB RAM	404	μA
	Active with 64 kB RAM	240	μA
PS0	Sleep with 0 kB RAM retention	1.5	μA

**Note:** All the current numbers mentioned in this section are subject to change.

## 7.5 Low Power and Interoperability Considerations

- It is recommended to set the device into Connected Power Save Mode only after IP configuration.
- If a wireless disconnection happens when the SiWG917 NWP is in Power Save Mode, disable the power save and try to reconnect to the AP.
- To avoid interoperability issues with various APs, enable the **Enhanced Max PSP feature**.
- For applications where throughput is not a major concern, consider disabling the higher data rates (MCS5, MCS6, and MCS7). To do this, make sure BIT(19) - SL\_SI91X\_FEAT\_DISABLE\_MCS\_5\_6\_7\_DATARATES in **config\_feature\_bit\_map** is not enabled in the boot configuration.
- Make a smart configuration of WLAN Keep-Alive, TCP Keep-Alive, and MQTT Keep-Alive parameters as per your application to reduce the current consumption.
- In power save modes, if the DNS requests fail with a few APs, the SL\_SI91X\_FEAT\_AGGREGATION in **feature\_bit\_map** is to be enabled in boot configuration. If this does not help, it is recommended to disable the power save and then make a DNS request API call and configure the device back into power save mode.
- If broadcast/multicast data is not important for your application, further power consumption can be reduced using the broadcast filter command.

## 7.6 Example Use Cases

### 7.6.1 M4 Sensing over a ULP Peripheral with Intermittent/Sparse Wi-Fi Communication

The application flow can be understood as:

1. SiWG917 comes out of RESET, NWP, and M4 initialization.
2. NWP set to unconnected sleep (Standby Power Save/without Retention).
3. M4 set to PS2 Sleep with retention for 200 milliseconds.
4. M4 gets back to PS2 Active upon RTC timer expiry; M4 receives data from ULP peripheral/sensor.
5. M4 processes the data received, based on data, and a decision is taken to execute either step-6 or step-8.
6. If wireless activity is needed, M4 switches back to PS4, and NWP is switched back to High Performance Mode.
7. NWP is configured as WLAN station, connects to a AP, performs data transfer over WLAN, disconnects from WLAN, and execution continues from step-2.
8. If wireless activity is not needed, execution continues from step-3.

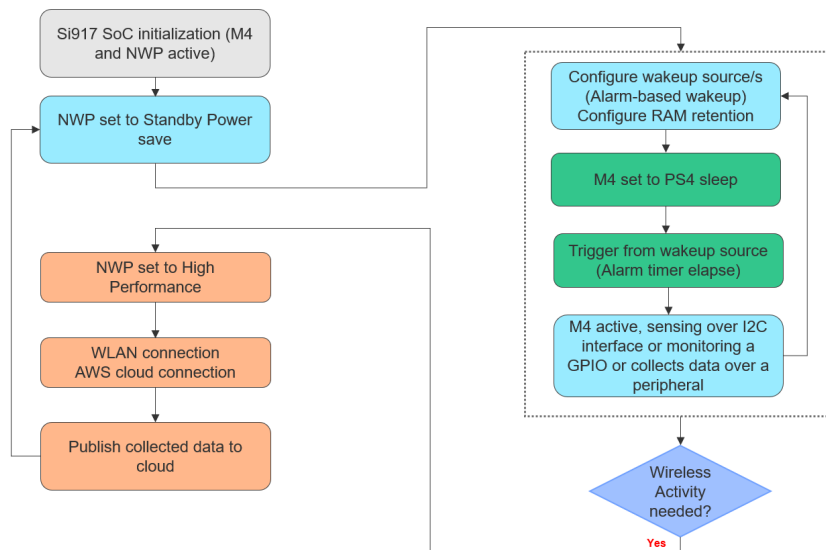


Figure 7.7. SiWG917 Low-power Mode Use Case 1

### 7.6.2 NWP in Connected Power Save, M4 Performing Activities Based on the Wireless Message Received (Smart Lock)

The application flow can be understood as:

1. SiWG917 comes out of RESET, NWP, and M4 initialization.
2. NWP configured as WLAN station, connects to AP, connects to AWS Cloud over MQTT.
3. NWP set to Connected Power Save (Associated Power Save).
4. M4 set to PS4 sleep with retention having RTC timer, UULP GPIO, and Wireless message as wakeup sources.
5. Upon Wireless based wakeup, based on the wireless message received, the smart lock is either locked, unlocked, or simply the current state of lock is published to AWS Cloud.
6. Upon GPIO based wakeup, the smart lock status is toggled (if the device is in Locked state, the state is switched to Unlocked), and updated lock state is published to AWS Cloud.
7. Upon every RTC timer expiry, the device wakes up and publishes the current state of lock to AWS Cloud.
8. Once the desired functionality is complete, the M4 permits NWP to sleep and execution continues from step-4.

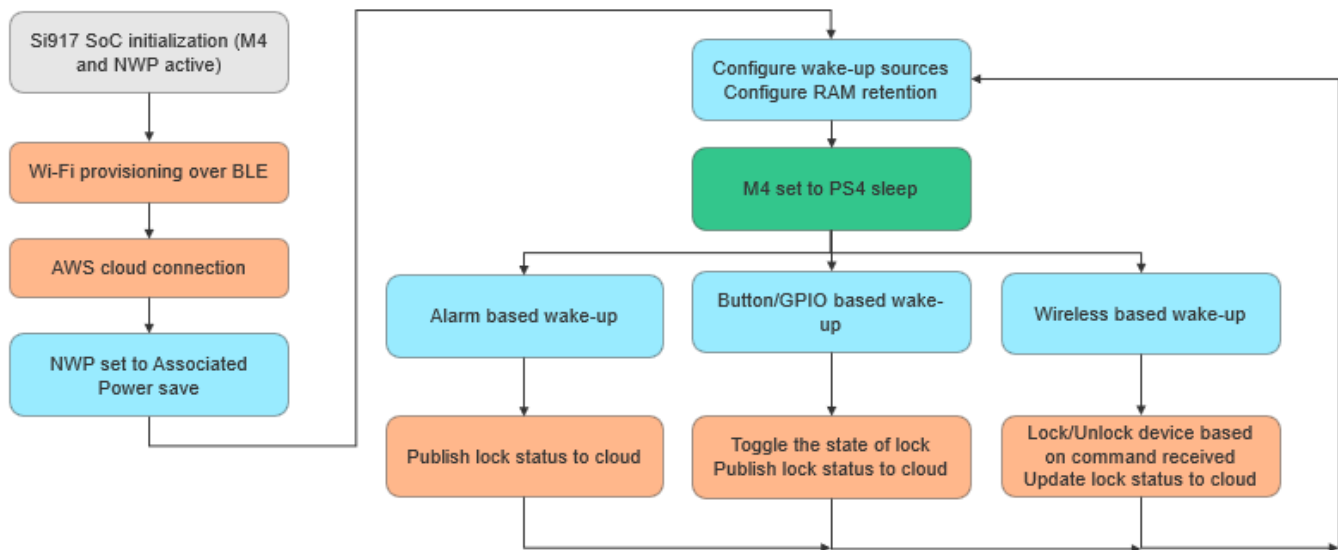


Figure 7.8. SiWG917 Low-power Mode Use Case 2

## 8. Future Scope

The following features are planned to be added in the SiWG917 low-power application note in the upcoming versions:

- Details of OFF state in SiWG917 Block Diagram.
- NWP Current Consumption in Power Save Modes with various APs.
- Current Consumption vs Listen Interval and Channel Utilization.
- Current Consumption when using Quick Scan and Quick Join features.
- Wake interval aligned with beacon vs wake interval aligned with DTIM beacon.
- More details on TWT based Power Save Mode.
- Current consumption details of SoC M4 Standby mode.
- Transition times for different power states in M4.
- Wakeup modes present for M4 Low-power Modes.

## 9. Revision History

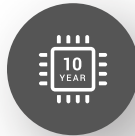
### January 2024

- Initial Revision

# Smart. Connected. Energy-Friendly.



**IoT Portfolio**  
[www.silabs.com/products](http://www.silabs.com/products)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support & Community**  
[www.silabs.com/community](http://www.silabs.com/community)

## Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

**Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit [www.silabs.com/about-us/inclusive-lexicon-project](http://www.silabs.com/about-us/inclusive-lexicon-project)**

## Trademark Information

Silicon Laboratories Inc.<sup>®</sup>, Silicon Laboratories<sup>®</sup>, Silicon Labs<sup>®</sup>, SiLabs<sup>®</sup> and the Silicon Labs logo<sup>®</sup>, Bluegiga<sup>®</sup>, Bluegiga Logo<sup>®</sup>, EFM<sup>®</sup>, EFM32<sup>®</sup>, EFR, Ember<sup>®</sup>, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals<sup>®</sup>, WiSeConnect<sup>®</sup>, n-Link, ThreadArch<sup>®</sup>, EZLink<sup>®</sup>, EZRadio<sup>®</sup>, EZRadioPRO<sup>®</sup>, Gecko<sup>®</sup>, Gecko OS, Gecko OS Studio, Precision32<sup>®</sup>, Simplicity Studio<sup>®</sup>, Telegesis, the Telegesis Logo<sup>®</sup>, USBXpress<sup>®</sup>, Zentri, the Zentri logo and Zentri DMS, Z-Wave<sup>®</sup>, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



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

[www.silabs.com](http://www.silabs.com)