AN1506: SiWx917 RCP Low Power Application Note

Power is critical for any battery-operated wireless device. Based on the user application, the wireless device can be idle for more time or for a very short time. During this idle period, the wireless devices can be set to sleep to conserve battery power. The SiWT917 wireless device is an optimal ultra-low power wireless (WLAN and Bluetooth) solution for developing applications requiring long battery life.

This application note describes the SiWx917 Power Save modes, how to choose Power Save modes based on application requirements, and current consumption analysis during various operational modes in RCP mode. It also explains how to configure SiWT917 in different Power Save modes using example commands and power optimization techniques that reduce SiWT917's current consumption.

KEY POINTS

- SiWx917's current consumption during various operational modes
- Choosing Power Save modes as per application requirements
- Supported commands and their usage
- Current measurement methods
- Power optimization techniques

1. Power Domains

The following block diagram illustrates various power domains of the SiWx917 chip in RCP mode.

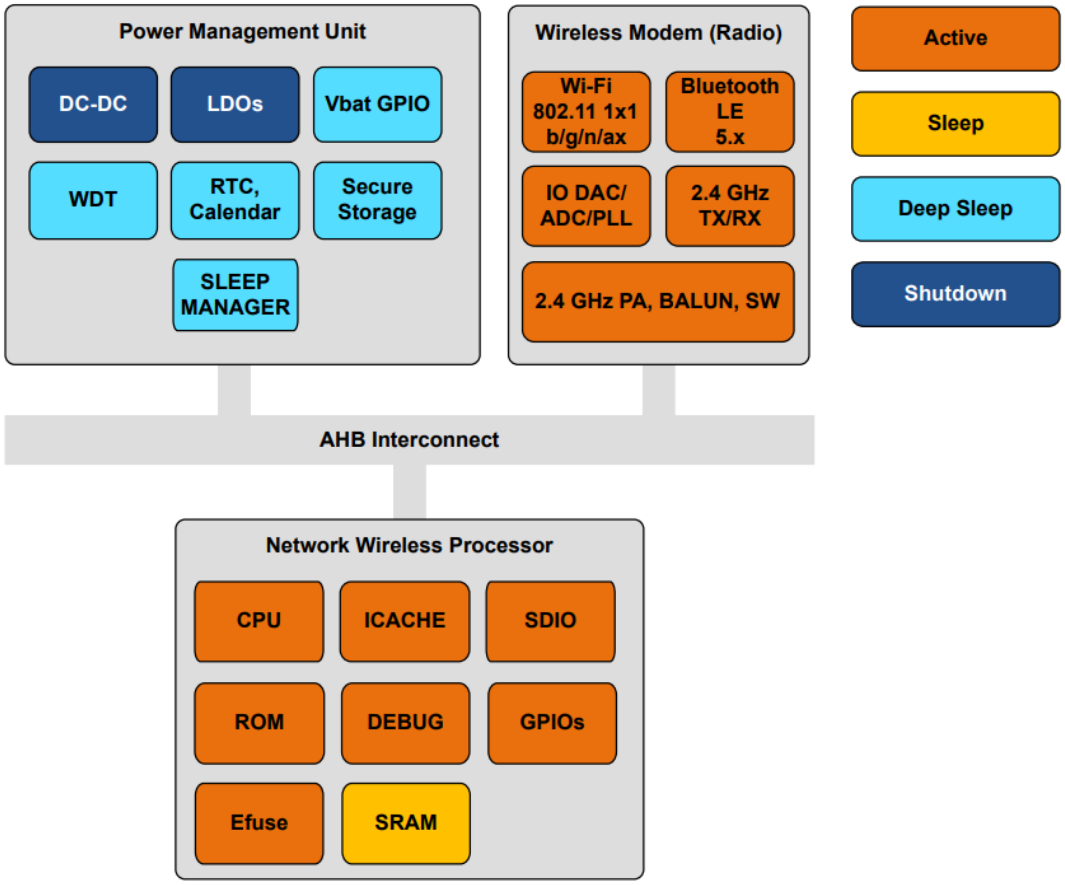

Figure 1.1. SiWx917 Power Domains

The power domains of SiWx917 in RCP mode are:

- ThreadArch® Network Processor (hereafter referred to as "NWP"): The multi-threaded processor that runs wireless and network stacks on independent threads.
- Power Management Unit (hereafter referred to as PMU): Responsible for supplying power required by various sections of SiWx917.
- Wireless Modem (Radio): Includes Wi-Fi, Bluetooth, Analog front-end, 4 GHz RF transceiver, and integrated power amplifier subsystems.

2. Power States

The SiWT917 can be in one of the following power states:

- **Active state**: All the power domains are powered-on.
- **Sleep state**: The PMU and SRAM domains are powered-on, and the remaining power domains are powered-off. The SRAM's contents are retained.
- **Off state**: Most sections in the PMU are powered-off. More details about this state will be added in the next version.

The operational modes in Active state are explained in detail in 3. Active State Operational Modes section. The operational mode in Sleep state is explained in detail in 4. Sleep State Operational Mode

3. Active State Operational Modes

The SiWx917 can be in any of the three operational modes in its Active state. Each operational mode consumes a different amount of current.

- Transmit Mode
- Receive Mode
- Listen Mode

3.1 Transmit Mode

In the Transmit Mode, all the power domains of the SiWx917 are powered-on, except the receiver sections of the Base Band Processor (BBP), Analog Front End (AFE), and RF Front End (RFFE) domains. This is the highest power-consuming mode.

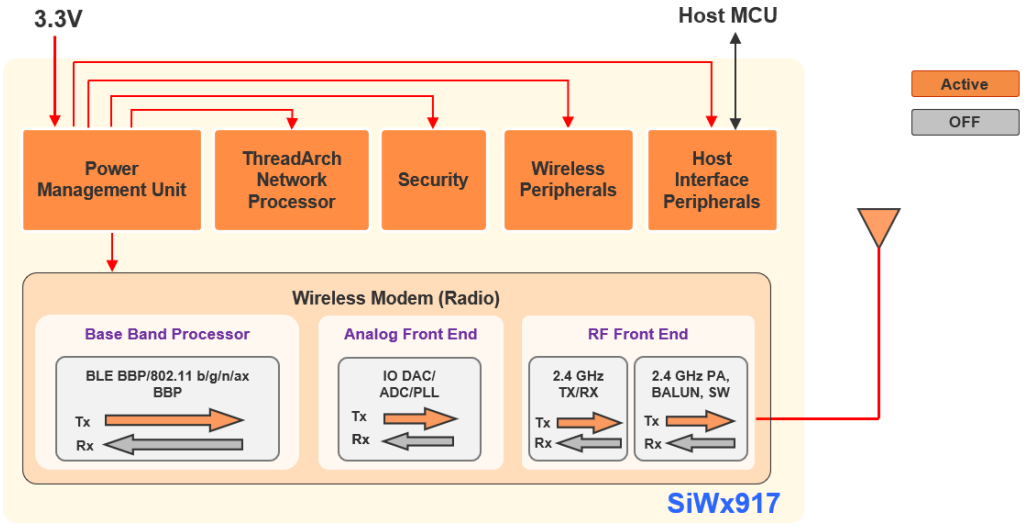

Figure 3.1. Transmit Mode in Active State

3.2 Receive Mode

In the Receive Mode, the transmit sections of the BBP, AFE, and RFFE are powered-off.

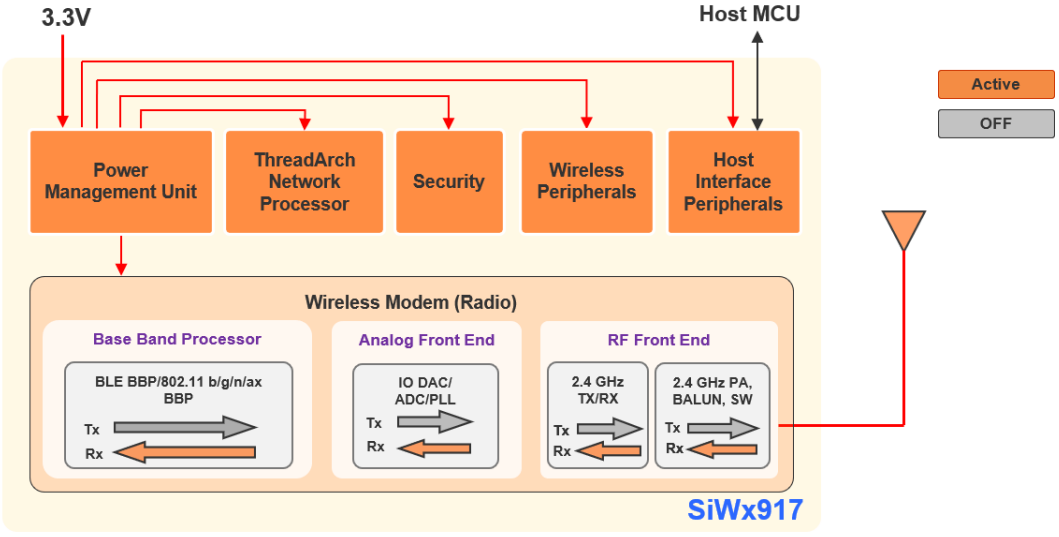

Figure 3.2. Receive Mode in Active State

3.3 Listen Mode

This mode is a subset of the receive mode. In Listen Mode, the transmit sections of BBP, AFE, and RFFE are powered-off. Certain receive portions of the BBP and AFE are powered-off as no packet reception is in progress.

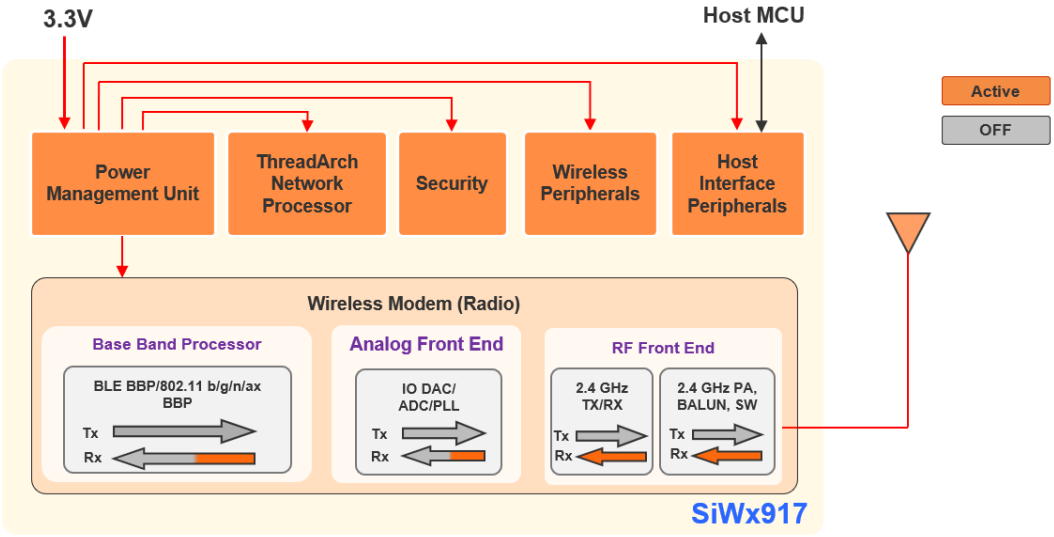

Figure 3.3. Listen Mode in Active State

4. Sleep State Operational Mode

The SiWx917's Sleep state operational mode is called Ultra-low Power Mode with RAM Retention.

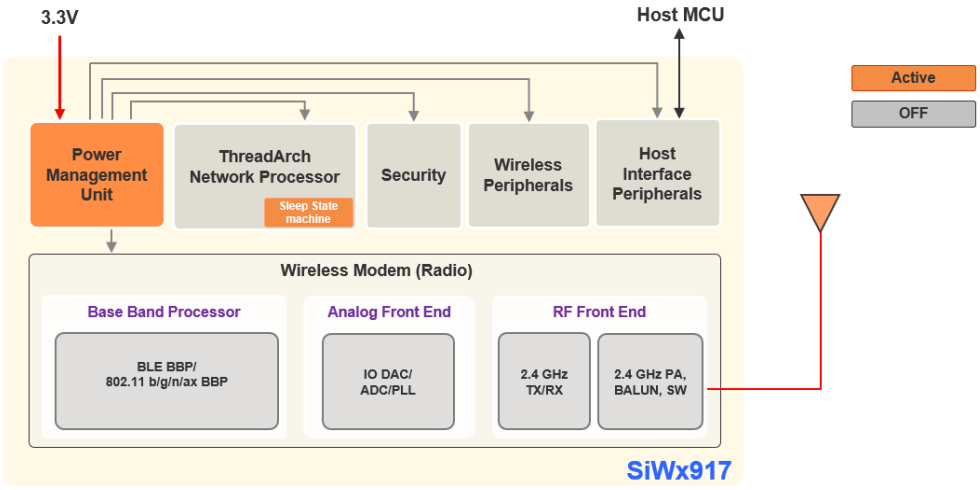

Figure 4.1. Ultra-Low Power Mode with RAM Retention

- In the Ultra-low Power mode with RAM Retention, the Wireless Modem, NWP, Security, and Host Interface power domains are powered-off. The SRAM contents and its current state are retained.
- The PMU has control over the other sections of the chip.
- The always-on logic domain operates on a lowered supply and a 32 kHz low-frequency clock to reduce power consumption.
- The SiWx917 RCP's Host Interface is inactive and thereby the interface is activated each time SiWx917 wakes up.

5. Power Save Mode

The SiWx9117 modules broadly support Ultra-low Power mode as outlined below:

- Ultra-low Power (ULP) Mode: A majority of the module is powered off except for a small section which has a timer and interrupts logic for waking up the module. The module cannot respond to the Host processor's commands/requests unless and until it gets wake up because of timeout or because of an interrupt asserted by Host processor. The sleep entry/exit procedures in this mode are indicated to the Host processor through a packet-based handshake.
- In Wi-Fi, only the Client (Station) mode supports power save. By default, the module will be in a power save disable state. The user has to enable it explicitly.

5.1 Device Sleep Mode

For each of the above power save modes, the module supports the sleep modes outlined below:

- **Unconnected Sleep:** In Unconnected Sleep, the module is not connected to an access point (AP). This is defined by the <deep_sleep_wakeup_period> parameter of the PS command.
- **Connected Sleep:** In the connected state, the module can operate in Max PSP. These profiles are used by the module to decide when to enter and exit from power save modes on the fly. They have to be selected based on the performance and power consumption requirements of the end product.

5.2 Wakeup Procedures and Data Retrieval

When in power save mode, the module wakes up at periodic intervals or due to certain events (like pending transmit packets from the Host). At every wakeup, the module has to poll the access point and check whether there are any pending RX packets destined for the module. The module uses different protocols to retrieve data from the access point based on the protocol supported by the access point. These data retrieval methods (protocol-based) are used to further classify the power save profiles described in the previous section into Max PSP.

- **Maximum Power Save Polling (Max PSP)**: In this mode, the module wakes up at the end of the sleep period (Listen or DTIM interval) and retrieves pending RX packets from the sccess point by sending a PS-POLL packet. It also transmits any packets received from the Host processor and then goes back to sleep. The parameters listed below are used by the module to decide the period of sleep during power save, in the same order of priority:
  1. <listen_interval_duration>
  2. <dtim_interval_duration>
  3. <num_beacons_per_listen_interval>
  4. <num_dtims_per_sleep>

Max PSP:

- Whenever the AP receives data frames that are destined for a station (here, SiWx917), it buffers the frames.
- The AP informs the station by setting the corresponding station's AID in the TIM element of the next immediate beacon.
- On the next wakeup (based on DTIM or listen interval), the station receives the beacon and checks for the TIM.
- 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 a data frame.
- 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.
- After receiving a data frame, the SiWx917 station sends it to the host.
- The station sends a PS-Poll frame to retrieve each data frame from the AP.
- While sending the last data frame to the station, the AP shall set the "More data" field to "0".
- After receiving the last data frame, the station goes back to sleep state.

The Max PSP saves more power but produces lower throughputs. If receiving a bulk amount of data, sending a PS-Poll frame to retrieve each frame affects the throughput and induces a considerable amount of delay.

- **Fast Power Save Polling (Fast PSP):** This profile is a variant of the Traffic-based PSP, which exits power save mode even for a single packet and enters the power save mode if no packet is transferred for the <monitor_interval> amount of duration. This profile is enabled independently for the Transmit and Receive directions if the <tx_threshold> and <rx_threshold> parameters are assigned zero, respectively, while assigning a non-zero value to the <monitor_interval> parameter.

Fast PSP:

- Whenever the AP receives data frames that are destined for a station (here, SiWx917), it buffers the frames.
- The AP informs the station by setting the corresponding station's AID in the TIM element of the next immediate DTIM beacon.
- On the next wakeup (based on DTIM or listen interval), the station receives the beacon and checks for the TIM.
- 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 packets from AP.
- After receiving a data frame, the SiWx917 station sends it to the host.
- After receiving the last data frame, the station waits for the monitor interval time configured and checks for data frames from AP.
- If no data is to be received, the station goes back to sleep state by sending a Null data frame with the PWR MGT bit set to "1" to the AP.

The Fast PSP saves less power and produces better throughput when compared to the Max PSP. If both throughput and power save are important for your application, use Fast PSP.

5.3 Configuring ULP Device Power Save

Install the driver as follows to enable ULP power save.

```
# insmod rsi_91x.ko rsi_zone_enabled=1 dev_oper_mode=<value> ps_sleep_type=2 ulp_handshake_mode=2

# insmod rsi_sdio.ko
```

5.4 Enabling Power Save

The following are the commands used in the power save configuration.

1. Enable the power save:

   # iw dev <interface_name> set power_save on

2. Disable power save:

   # iw dev < interface_name> set power_save off

3. Check the power save status:

   # iw dev <interface_name> get power_save

Here, the interface_name will vary from host to host. We can get the interface name with the following command:

iw dev

For BT coexistence with Wi-Fi, give BT the power save command.

hcitool -I hci0 cmd <vendor command> <power save related> <power save ON/OFF> <ULP(0x02)/LP(0x01) power save> <sleep duration>

Example:

hcitool -i hci0 cmd 0x3f 0x0003 0x01 0x02 0xff

5.5 Configure Power Save Parameters/ Profiles Through Debugfs Dynamically

The driver supports dynamic configuration of power save type and profile parameters using debugfs as explained below.

To update the power save parameter, use the following command:

```
# echo <sleep_type> <tx_threshold> <rx_threhold> <tx_hysteresis> < rx_hysteresis> <monitor_interval> <listen_interval_duration>
<num_beacons_per_listen_interval> <dtim_interval_duration> <num_dtims_per_sleep> <deep_sleep_wakeup_period>
<uapsd_wakeup_period> >/sys/kernel/debug/phy<X>/ ps_params
```

The input parameters of the power save command are explained below.

- **<sleep_type>**: This parameter is used to select the sleep mode between LP (1) and ULP (2) modes.
- **<tx_threshold>**: If a non-zero value is assigned, this parameter is used to set a threshold for the Transmit throughput computed during the <monitor_interval> period so that the module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and supported TX threshold is 0 to 10 Mbps.
- **<rx_threshold>**: If a non-zero value is assigned, this parameter is used to set a threshold for the Receive throughput computed during the <monitor_interval> period so that the module can decide to enter (throughput ≤ threshold) or exit (throughput > threshold) the power save mode. The value is in Mbps and supported RX threshold is 0 to 10 Mbps.
- **<tx_hysteresis>**: The decision to enter or exit power save mode based on the Transmit throughput alone can result in frequent switching between the power save and non-power save modes. If this is not beneficial, the <tx_hysteresis> parameter can be used to make the module re-enter the power save mode only when the throughput falls below the difference between the <tx_threshold> and <tx_hysteresis> values. The value is in Mbps and the minimum value is 0 Mbps. This parameter should be assigned a value which is less than the value assigned to the <tx_threshold> parameter.
- **<rx_hysteresis>**: The decision to enter or exit power save mode based on the Receive throughput which alone can result in frequent switching between the power save and non-power save modes. If this is not beneficial, the <rx_hysteresis> parameter can be used to make the module re-enter the power save mode only when the throughput falls below the difference between the <rx_threshold> and <rx_hysteresis> values. The value is in Mbps and the minimum value is 0 Mbps. This parameter should be assigned a value which is less than the value assigned to the <rx_threshold> parameter.
- **<monitor_interval>**: This parameter specifies the duration (in milliseconds) over which the Transmit and Receive throughput's are computed to compare with the <tx_threshold>, <rx_threshold>, <tx_hysteresis>, and <rx_hysteresis> values. The maximum value of this parameter is 30000 ms (30 seconds).
- **<listen_interval_duration>**: This parameter specifies the duration (in milliseconds) for which the module sleeps in the connected state power save modes. If a non-zero value is assigned to this parameter, it takes precedence over the other sleep duration parameters that follow (<num_beacons_per_listen_interval>, <dtim_interval_duration>, <num_dtims_per_sleep>). The maximum duration for which the device supports sleep is 4095 times the duration of the beacon interval considering the listen interval parameters of the access point. The maximum value for this parameter can be 65535, but the duration should be the deciding factor in the beacon interval of the access point. This parameter is considered only after the module is connected to the access point. For example, if the beacon interval of the AP is 100 ms and the listen interval of AP is 8 beacons, then the maximum time the device can sleep without any data loss is 800 ms (8 * 100). Hence, the listen_interval_duration can be up to 800 ms.
- **<num_beacons_per_listen_interval>**: This parameter specifies the number of beacon intervals for which the module sleeps in the connected state power save modes. Here, the device will wake up for the nth beacon, where n is the listen interval value programmed by the user. If a non-zero value is assigned to this parameter, it takes precedence over the other sleep duration parameters that follow (<dtim_interval_duration>, <num_dtims_per_sleep>). This parameter is used only when the above parameter is assigned to 0. The maximum value for this parameter is 4095. The value for this parameter also has to be chosen according to the listen interval of the access point. This parameter is considered only after the module is connected to the access point.
- **<dtim_interval_duration>**: This parameter specifies the duration (in milliseconds) for which the module sleeps in the connected state power save modes. The device will wake up for the nearest DTIM beacon after the time which the user has programmed expires. This parameter can be used when DTIM information is not available. If a non-zero value is assigned to this parameter, then it takes precedence over the other sleep duration parameter that follows (<num_dtims_per_sleep>). This parameter is used only when the above parameters are assigned 0. The maximum value for this parameter can be 10000 ms. This parameter is considered only after the module is connected to the access point.
- **<num_dtims_per_sleep>**: This parameter specifies the number of DTIM intervals for which the module sleeps in the connected state power save modes. This parameter has least priority compared to the ones above and is used only if the above parameters are assigned to 0. The maximum value for this parameter is 10. This parameter is considered only after the module is connected to the access point.
- **<deep_sleep_wakeup_period>**: This parameter specifies the duration (in milliseconds) for which the module sleeps in the Deep Sleep mode. For LP mode, a value of 0 for the <sleep_duration> parameter programs the module to be in Deep Sleep mode indefinitely until it is woken up by the Host processor via the host interface. The value of 0 is invalid for ULP mode and should not be used. The maximum value for this parameter can be 65535.

Beacon Interval

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

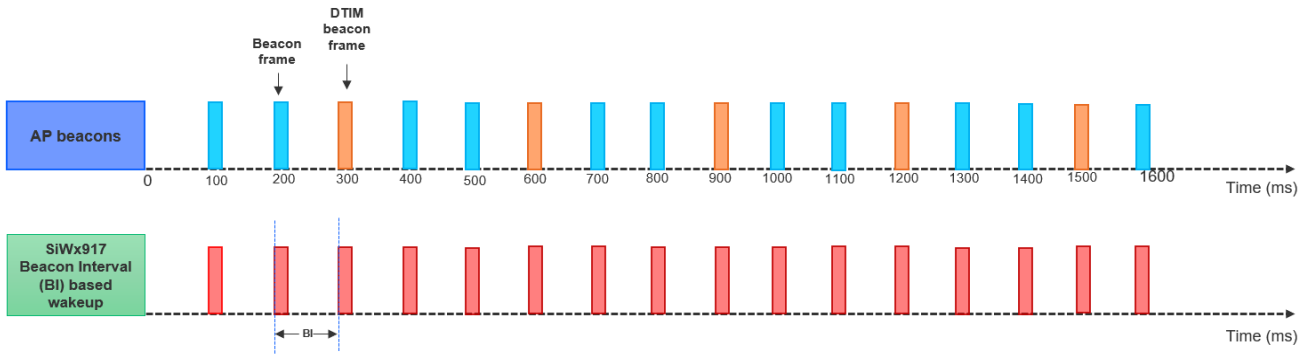

Figure 5.1. Beacon Interval-based Wakeup

5.5.1 DTIM Interval

The SiWx917 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 BI-based wakeup of SiWx917 for AP's BI = 100 ms and DTIM period = 3. In this case, the SiWx917's wake interval = 300 ms.

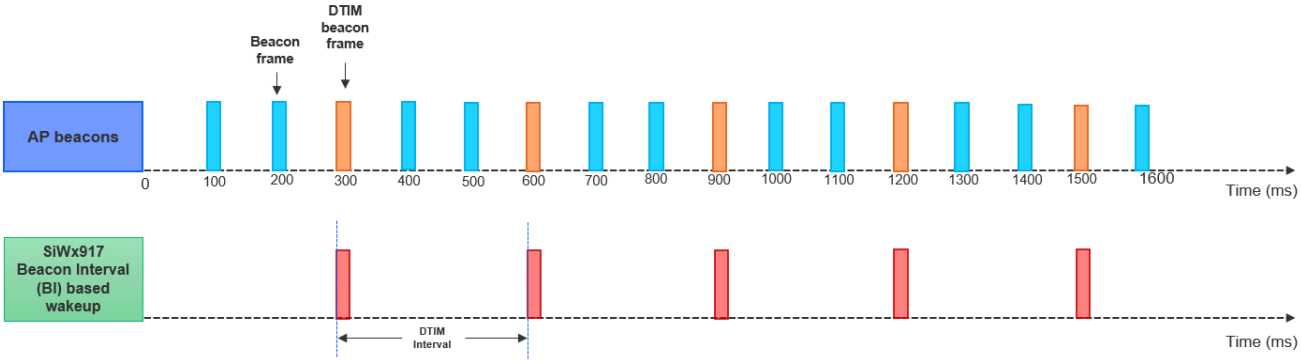

Figure 5.2. DTIM Interval-based Wakeup

5.6 Target Wake Time (TWT)

Target Wake Time (TWT) is a feature introduced in Wi-Fi 6 (802.11ax) designed to improve the efficiency of wireless communication and a power-saving feature in Wi-Fi that allows devices to negotiate when and how frequently they wake up to transmit or receive data, improving battery life and overall network efficiency. It allows the access point to schedule wake timings for its connected stations, ensuring that no two Wi-Fi stations wake at the same time. This method helps avoid packet collisions, thus reducing retransmissions and, in turn, reducing the station's current consumption.

Here are some key benefits of TWT:

- **Reduced Power Consumption**: By adopting sleep times, TWT significantly extends battery life for devices.
- **Minimized Contention**: Scheduled uplink access reduces congestion among devices, ensuring smoother communication.
- **Optimized Performance**: TWT helps in maintaining efficient operation even in crowded network environments.

5.7 TWT Setup/Teardown

Follow the steps below for TWT Setup/Teardown in an open source driver.

1. Enable CONFIG_TWT_SUPPORT in both driver Makefile(rsi folder) and apps Makefile(apps/Makefile).
2. Compile the driver as follows.

   #make clean;make

3. Insert the driver.

   # insmod rsi_91x.ko driver_mode_value=1 rsi_zone_enabled=<val> ...

   # insmod rsi_sdio.ko

4. Connect to third AP by following the command below.

   # wpa_supplicant -i <interface_name> -D nl80211 -c <sta_settings.conf> -ddddt > log1 &

5. Go to the release folder.

To start or stop the TWT session, give the twt_config command command, which will trigger TWT Setup/Teardown frame:

./onebox_util rpine0 twt_config <wake_duration> <wake_duration_tolerance> <wake_int_exp> <wake_int_exp_tolerance> <wake_int_mantissa> <wake_int_mantissa_tolerance> <implicit_twt> <unannounced_twt> <triggered_twt> <negotiation_type> <twt_channel> <twt_protection> <twt_flow_id> <restrict_twt_outside_tsp> <twt_retry_limit> <twt_retry_interval> <twt_req_type> <twt_enable> <wake_duration_unit>

Example:

./onebox_util rpine0 twt_config 255 255 15 15 650 10 1 1 0 0 0 0 1 1 6 10 1 1 0

- **wake_duration**: This is the nominal minimum wake duration of TWT. This is the time for which DUT will be in wake state for Transmission or reception of data. Allowed values range is 0-255.
- **wake_duration_tol**: This is the tolerance allowed for wake duration in case of suggest TWT. Received TWT wake duration from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Allowed values are 0-255.
- **wake_int_exp**: TWT Wake interval exponent. It is exponent to base 2. Allowed values are 0 - 31.
- **wake_int_exp_tol**: This is the allowed tolerance for wake_int_exp in case of suggest TWT request. Received TWT wake interval exponent from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Allowed values are 0 - 31.
- **wake_int_mantissa**: This is the TWT wake interval mantissa. Allowed values are 0-65535.
- **wake_int_mantissa_tol**: This is the tolerance allowed for wake_int_mantissa in case of suggested TWT. Received TWT wake interval mantissa from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Allowed values are 0-65535.
- **implicit_twt**: If enabled (1), the TWT requesting STA calculates the next TWT by adding a fixed value to the current TWT value. Explicit TWT is currently not allowed.
- **un_announced_twt**: If enabled (1), TWT requesting STA does not announce its wake up to AP through PS-POLLs or UAPSD trigger frames.
- **triggered_twt**: If enabled (1), at least one trigger frame is included in the TWT Service Period (TSP).
- **negotiation_type**: If disabled (0), the TWT requesting STA supports individual TWT. Broadcast TWT is currently not supported.
- **twt_channel**: Currently this configuration is not supported. Allowed values are 0-7.
- **twt_protection**: If enabled (1), TSP is protected. This is negotiable with AP. Currently not supported. Only zero is allowed.
- **twt_flow_id**: This is TWT flow ID.
- Range : 0-7 (should be the same for setup and teardown. Otherwise, an error will be triggered).
- 0xff : To teardown all active sessions. This is valid only in Teardown TWT case.
- **restrict_tx_outside_tsp**: If enabled (1), any TX outside the TSP is restricted. Else, TX can happen outside the TSP also.
- **twt_retry_limit**: This is the maximum number of retries allowed if the TWT response frame is not recieved for the sent TWT request frame. Allowed values are 0 - 15.
- **twt_retry_interval**: The interval, in seconds, between two TWT request retries. Allowed values are 5 - 255.
- **req_type**: This is the TWT request type.
- - 0 - Request TWT
  - 1 - Suggest TWT
  - 2 - Demand TWT
- **twt_enable**: If enabled (1), TWT setup frame is triggered or if disabled (0), TWT Teardown frame is triggered.
- **wake_duration_unit**: This parameter defines unit for wake_duration. Allowed values are 0 (256 µS) and 1 (1024 µS).

The following command provides the status of the ongoing TWT session.

```
# ./onebox_util rpine0 twt_status
```

TWT Command Status

Sr.No	STATUS	DESCRIPTION
1.	TWT_DEVICE_NOT_IN_CONNECTED_STATE	Occurs when the device is not connected to AP.
2.	TWT_SETUP_ERR_SESSION_ACTIVE	Occurs when user tries to give TWT setup command when there is already an active TWT session.
3.	TWT_TEARDOWN_ERR_FLOWID_NOT_MATCHED	Occurs when TWT teardown command is given with a flow ID that does not match existing session flow ID.
4.	TWT_TEARDOWN_ERR_NOACTIVE_SESS	Occurs when teardown command is given while there is no active session.
5.	TWT_SETUP_SESSION_IN_PROGRESS	Occurs when user tries to give TWT setup command when there is a TWT session already in progress.
6.	TWT_SESSION_SUCC	TWT session setup success. TWT session is active.
7.	TWT_UNSOL_SESSION_SUCC	Unsolicited TWT setup response from AP accepted. TWT session is active.
8.	TWT_SETUP_AP_REJECTED	TWT reject frame recieved in response for the sent TWT setup frame.
9.	TWT_SETUP_RSP_OUTOF_TOL	TWT response parameters from AP for when TWT suggest request is not within tolerance set by user.
10.	TWT_SETUP_RSP_NOT_MATCHED	TWT response parameters from AP for when TWT demand request does not match parameters given by user.
11.	TWT_SETUP_UNSUPPORTED_RSP	Unsupported TWT response from AP.
12.	TWT_TEARDOWN_SUCC	TWT session teardown success.
13.	TWT_AP_TEARDOWN_SUCC	TWT session teardown from AP success.
14.	TWT_SETUP_FAIL_MAX_RETRIES_REACHED	TWT setup request retried maximum number of times as configured by user.
15.	TWT_INACTIVE_DUETO_ROAMING	TWT session inactive due to roaming.
16.	TWT_INACTIVE_DUETO_DISCONNECT	TWT session inactive due to disconnect.
17.	TWT_INACTIVE_NO_AP_SUPPORT	TWT session inactive as connected AP does not support TWT.

5.8 Reschedule TWT

Users can suspend and resume the current twt-session dynamically. Follow the steps below to reschedule TWT.

1. Check whether the TWT session is active or not using the command below.

   ```
   # ./onebox_util rpine0 twt_status
   ```

2. If the TWT session is active, use the command below to reschedule TWT.

   ```
   #./onebox_util rpine0 reschedule_twt twt_flow_id twt_action suspend_duration
   ```

3. **twt_flow_id**: Active twt-session flow id.

   twt_action: These are the types for twt_action:

   - 0 - Suspend Indefinitely
   - 1 - Suspend for duration
   - 2 - Resume Immediately

   suspend_duration: Suspend TWT for a given interval. This should be a non-zero value only when twt_action is 1. Otherwise, it should be zero.

4. Check the reschedule_twt session status with the command below.

   ```
   # ./onebox_util rpine0 twt_status
   ```

6. Revision History

Revision 1.0

March, 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