# Manage Power Down state for Azure Sphere devices

- 02/25/2020
- 3 minutes to read

A high-level Azure Sphere application can use the power management API to put the device into the Power Down state. The Power Down state is the lowest possible power state a device can be in other than being fully powered off. The device can be woken up by either of two events:

1. Arrival of an input signal triggering the WAKEUP pin.
2. Passage of a specified amount of time.

To use this feature, you must:

1. Configure your hardware.
    - See the RTC power requirements and Power Down considerations sections of the MT3620 Hardware Notes for general hardware design considerations.
    - See the Power supply and Power Down mode sections of the MT3620 RDB user guide to see how to configure the RDB hardware for Power Down mode.
2. Declare the ForcePowerDown value for the **PowerControls** capability in the application manifest.
3. Use **PowerManagement_ForceSystemPowerDown** from the Power Management API.

## Power Down state

The Power Down state has the following characteristics:

- Everything is powered off except the real-time clock (RTC). This means all connectivity, RAM, flash, CPU cores, and so forth, are all powered off.
- There is no state preservation. Waking up from Power Down is equivalent to a cold boot.
- Wake from Power Down occurs when the real-time clock alarm fires (time-based wakeup) or when the WAKEUP pin is pulled low (event-based wakeup), whichever comes first.

## MT3620 specifics

MediaTek MT3620 Power Down considerations are provided in MT3620 hardware notes.

## Force Power Down and updates

**Warning**

Failure to follow the guidance in this section could result in your device being unable to fetch application or OS updates and requiring recovery. Please read carefully before using ForcePowerDown.

Because both ForcePowerDown and ForceReboot allow an application to power down the device at any time, it becomes the responsibility of that application to ensure the device is still able to periodically check for updates when using ForcePowerDown or ForceReboot. To make it easier to check for updates in this scenario, we have introduced update-related system event notifications to give applications information about the update process so your apps can make an informed decision about when to power down the device. The relevant available system event notifications are:

- SysEvent_Events_NoUpdateAvailable: The update check has finished and no OS or application updates are available.
- SysEvent_Events_UpdateStarted: An OS or application update has started downloading. This event will be followed by the **SysEvent_Events_UpdateReadyForInstall** event when the update is fully downloaded and ready for installation. If no update is available, **SysEvent_Events_NoUpdateAvailable** will be sent instead of this event.
- SysEvent_Events_UpdateReadyForInstall: An update has finished downloading and is ready to be applied on reboot.

Applications that use ForcePowerDown need to be mindful of the update state of the device. Applications that use ForcePowerDown should always register for these events and heed them to ensure the app does not cause an update to be postponed indefinitely.

We provide a sample application that demonstrates how to properly build update checks into an application that uses ForcePowerDown. It is highly recommended that you begin with this sample when developing apps with ForcePowerDown.

# Application termination

After a Power Down request is issued, a SIGTERM signal is sent to your app. If your app handles the signal, it has up to 2 seconds to do cleanup work. Otherwise, the app will be terminated immediately. For more information, including how to properly handle the signal, see app termination for update.

# Sample application

The Power Down sample application demonstrates how to properly make use of ForcePowerDown to reduce power consumption while still ensuring the device will periodically stay awake to check for OS and app updates.

This sample blinks an LED red, representing work or 'business logic' that an app may need to do while the device is awake, then powers down the device for a specified amount of time. Every Nth Power Down/wake cycle, the app will keep the device awake longer to check for updates

instead of immediately powering down following completion of the business logic (the blinking red LED in this case). To ensure updates have completed before powering down, the sample app makes use of three system event notifications (SysEvent_Events_NoUpdateAvailable, SysEvent_Events_UpdateStarted, and SysEvent_Events_UpdateReadyForInstall) that inform the app about the status of the update check/download. The sample app also shows how to measure the current consumption of the RDB to validate that the device is entering Power Down.