Suspend mode

SoC power states

The power states of the system on a chip (SoC) are: on, idle, and suspend. “On” is when the SoC is running. “Idle” is a medium power mode where the SoC is powered but doesn't perform any tasks. “Suspend” is a low-power mode where the SoC is not powered. The power consumption of the device in this mode is usually 100 times less than in the “On” mode.

Non-wake-up sensors

Non-wake-up sensors are sensors that do not prevent the SoC from going into suspend mode and do not wake the SoC up to report data. In particular, the drivers are not allowed to hold wake-locks. It is the responsibility of applications to keep a partial wake lock should they wish to receive events from non-wake-up sensors while the screen is off. While the SoC is in suspend mode, the sensors must continue to function and generate events, which are put in a hardware FIFO. (See Batching (/devices/sensors/batching.html) for more details.) The events in the FIFO are delivered to the applications when the SoC wakes up. If the FIFO is too small to store all events, the older events are lost; the oldest data is dropped to accommodate the latest data. In the extreme case where the FIFO is nonexistent, all events generated while the SoC is in suspend mode are lost. One exception is the latest event from each on-change sensor: the last event must be saved (/devices/sensors/batching.html#precautions_to_take_when_batching_non-wake-up_on-change_sensors)
outside of the FIFO so it cannot be lost.

As soon as the SoC gets out of suspend mode, all events from the FIFO are reported and operations resume as normal.

Applications using non-wake-up sensors should either hold a wake lock to ensure the system doesn't go to suspend, unregister from the sensors when they do not need them, or

expect to lose events while the SoC is in suspend mode.

Wake-up sensors

In opposition to non-wake-up sensors, wake-up sensors ensure that their data is delivered

independently of the state of the SoC. While the SoC is awake, the wake-up sensors behave like non-wake-up-sensors. When the SoC is asleep, wake-up sensors must wake up the SoC to deliver events. They must still let the SoC go into suspend mode, but must also wake it up when an event needs to be reported. That is, the sensor must wake the SoC up and deliver the events before the maximum reporting latency has elapsed or the hardware FIFO gets full. See Batching (/devices/sensors/batching.html) for more details.

To ensure the applications have the time to receive the event before the SoC goes back to sleep, the driver must hold a "timeout wake lock" for 200 milliseconds each time an event is being reported. *That is, the SoC should not be allowed to go back to sleep in the 200 milliseconds following a wake-up interrupt.* This requirement will disappear in a future Android release, and we need this timeout wake lock until then.

How to define wake-up and non-wake-up sensors?

Up to KitKat, whether a sensor was a wake-up or a non-wake-up sensor was dictated by the sensor type: most were non-wake-up sensors, with the exception of the proximity (/devices/sensors/sensor-types.html#proximity) sensor and the significant motion detector (/devices/sensors/sensor-types.html#significant_motion).

Starting in L, whether a given sensor is a wake-up sensor or not is specified by a flag in the sensor definition. Most sensors can be defined by pairs of wake-up and non-wake-up variants of the same sensor, in which case they must behave as two independent sensors, not interacting with one another. See Interaction (/devices/sensors/interaction.html) for more details.

Unless specified otherwise in the sensor type definition, it is recommended to implement one wake-up sensor and one non-wake-up sensor for each sensor type listed in Sensor types (/devices/sensors/sensor-types.html). In each sensor type definition, see what sensor (wake-up or non-wake-up) will be returned by `SensorManager.getDefaultSensor(sensorType)`. It is the sensor that most applications will use.

Content and code samples on this page are subject to the licenses described in the Content License (/license). Java is a registered trademark of Oracle and/or its affiliates.