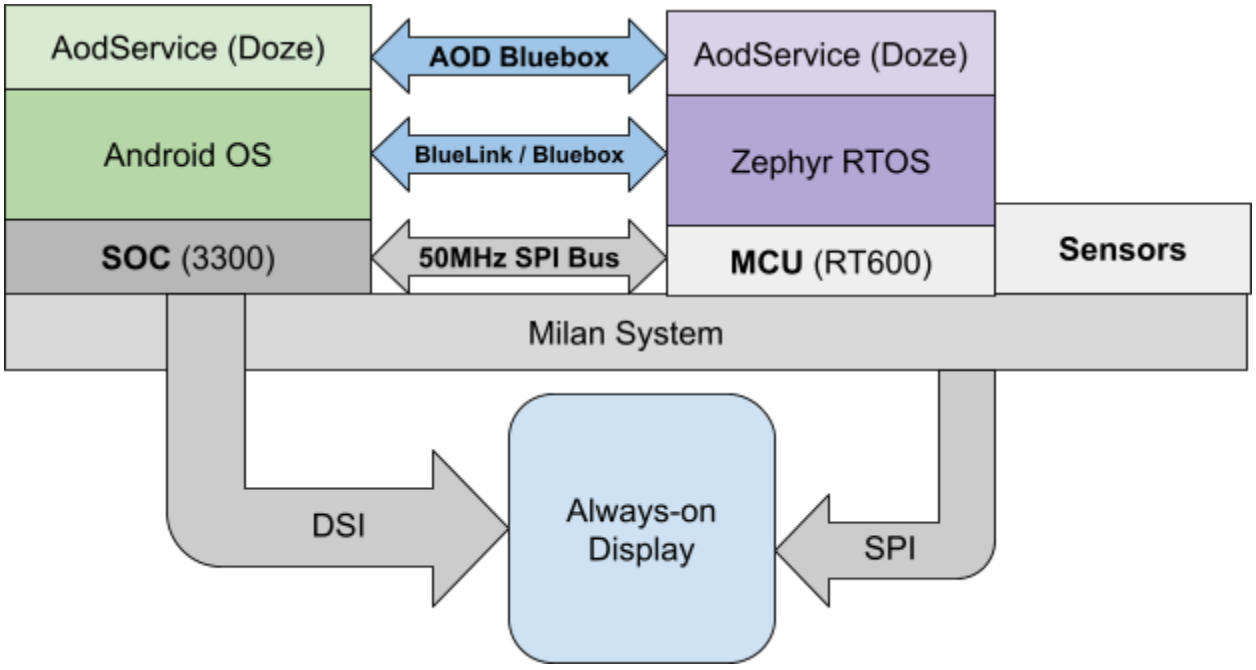
Milan Always-on Display Handover

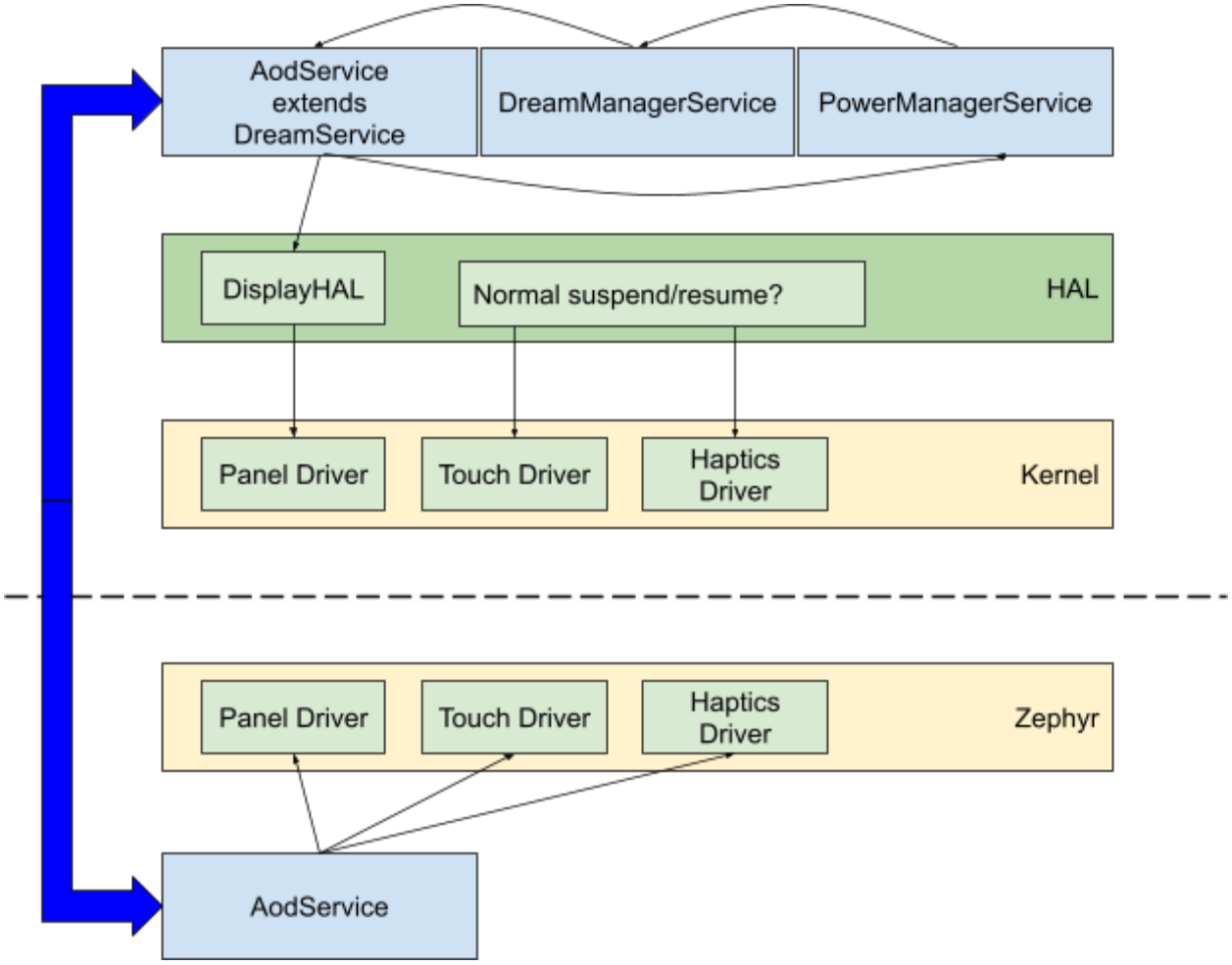
Status: { Draft | RFC | Final | POR | Inactive }
Author: Christian Flowers, Dalton Flanagan
Last Update: 06/18/2020

This document complements the Milan Always-on Display Architecture document, and focuses on the low level handover of the display, touch, and haptics between the SOC and MCU. This includes device power up along with the handover, and any configuration changes needed to optimize for power or responsiveness.



System Design (Hardware Handoff)

In API 21, Android added the concept of Doze mode as an interim display state. Their implementation supports both an SOC run Activity that activates on user gestures, and the framework for handing over to/from an external component when the screen logically (but not physically) turns on and off.



*This needs lots of detail about how Doze works, interaction with PowerManagerService

```
<bool name="config_powerDecoupleInteractiveModeFromDisplay">true</bool>
<bool name="config_powerDecoupleAutoSuspendModeFromDisplay">true</bool>
```

And what these need to be set to, investigation needed.

HAL changes are necessary for the DisplayHAL
Kernel changes necessary for resume/suspend of display at minimum.
Do we keep touch up, or let it suspend and resume?

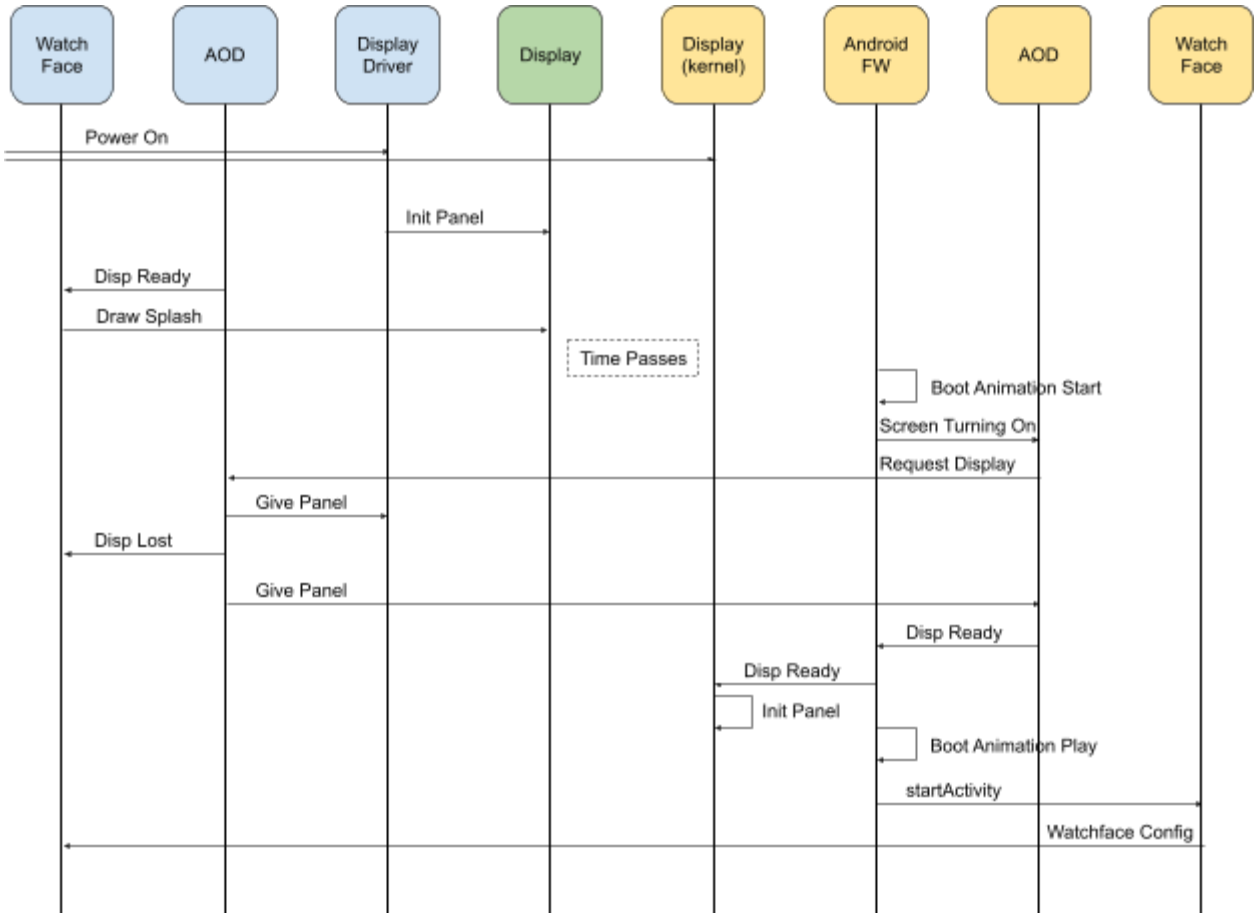
https://drive.google.com/file/d/1VYC7KVrkCEV5XAOjT0pDkZ0jsSvEF6Vx

Use Cases and Sequences

The use cases are identical, but these are sequences from a V1 perspective where the Watchface is expected to handle the entire UX. In general, the SOC must request access to the display from the MCU before attempting to write to it.

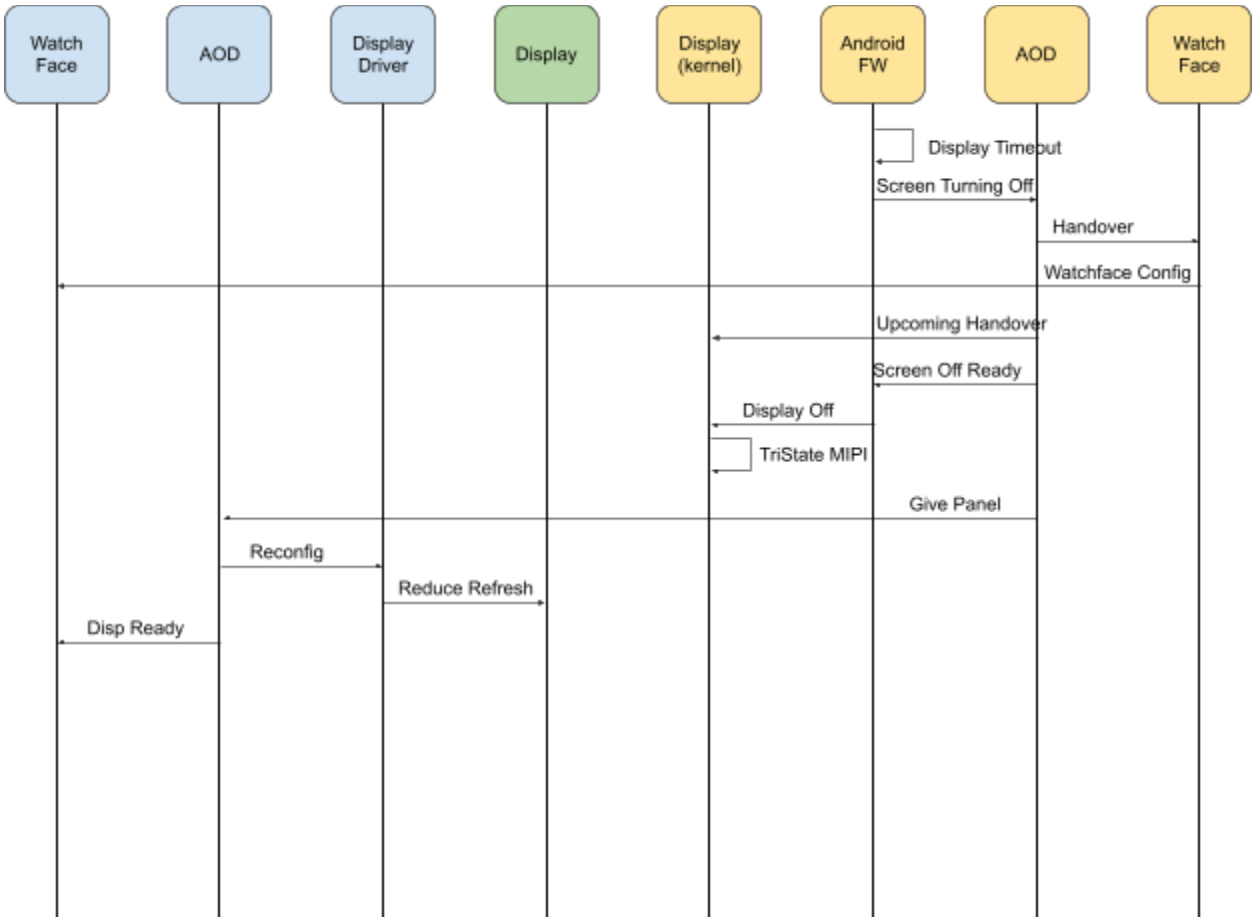
Power On/Boot Up

In the power on case, both the MCU and SOC are booted in parallel. Due to its faster boot times, the MCU could show an initial splash screen, followed by animation once the SOC is up. After the standard display timeout, control would revert to the MCU.



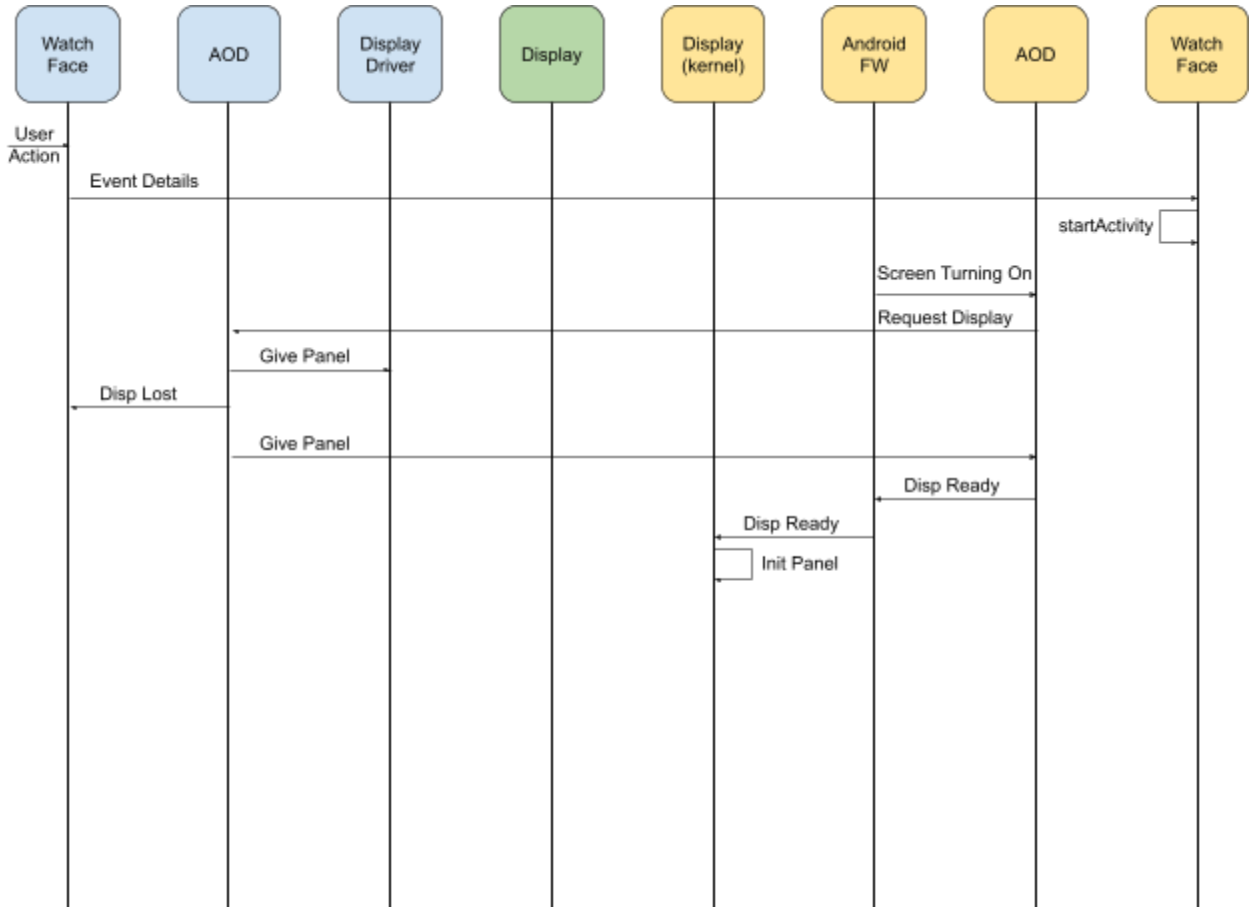
SOC Display Timeout

Transitioning the display from the SOC back to the MCU occurs automatically with the Android display timeout. Milan may also choose to dim the display with the SOC still running prior to screen off, or override the stock Android behavior. Additionally, a gesture like a palm cover could force a handover to the MCU. The SOC will never actually turn the display off.



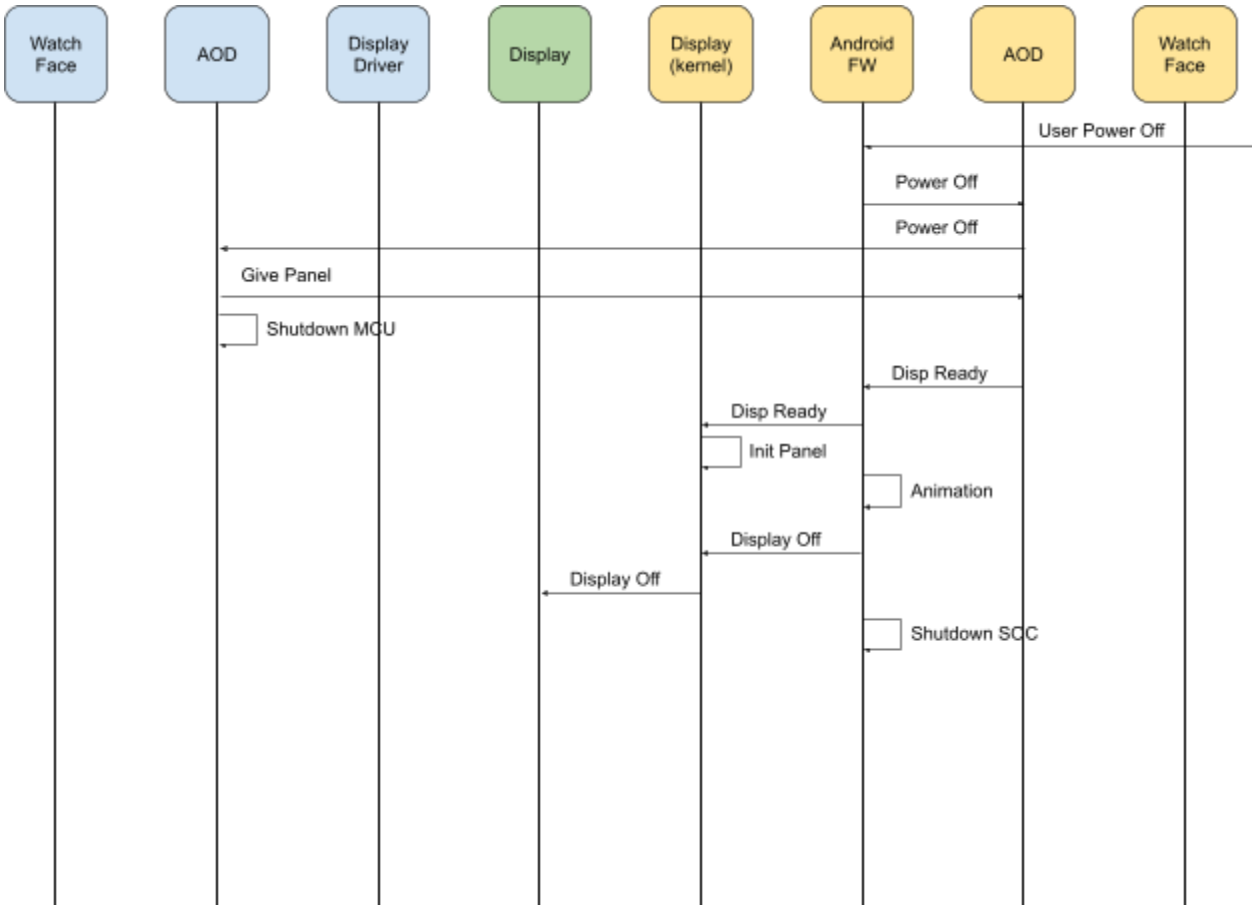
MCU Interaction

This starts with the MCU in control, but based on an explicit user action we wish to wake the SOC. Whether the display is taken over is determined by the Watchface on the SOC side. Note, how to continue a touch action or gesture



Power Off

When turning off the device, the SOC will take the longest and can provide the best experience. As a result, the MCU should relinquish control of the display and shut down immediately. The SOC can then takeover, play any animations, then power down.



Use Cases:
From MCU:
-Gesture to wake or launcher
-Complication press
-App launch
-”Deep link” app transition

From SOC:
-Screen Timeout; User Palm forces “off” to watchface
-”Deep link” app transition
-App launch

Appendix

Android Doze

Config.xml:

<string name="config_dozeComponent">com.android.systemui/com.android.**systemui.doze**.DozeService</string>
config_dozeAfterScreenOffByDefault
config_dozePulsePickup
config_dozeDoubleTapSensorType
config_dozeLongPressSensorType
config_dozeAlwaysOnDisplayAvailable
config_dozeAlwaysOnEnabled
config_displayBlanksAfterDoze (which direction is this)
config_displayBrightnessBucketsInDoze (???)

Need to decide if we're "turning the display off" from the SOC standpoint:
config_powerDecoupleAutoSuspendModeFromDisplay (SOC knows the display is on but suspends anyway??)
config_powerDecoupleInteractiveModeFromDisplay (???)

`frameworks/base/packages/SystemUI/src/com/android/systemui/doze/`

DozeService is an extension of DreamService.

`frameworks/base/core/java/com/android/internal/hardware/AmbientDisplayConfiguration.java`

DOZE
DOZE_SUSPEND

WAFEFULNESS_DREAMING
WAKEFULNESS_DOZING