Overview

The Surface/System Aggregator Module (SAM, SSAM) is an (arguably *the*) embedded controller (EC) on Microsoft Surface devices. It has been originally introduced on 4th generation devices (Surface Pro 4, Surface Book 1), but its responsibilities and feature-set have since been expanded significantly with the following generations.

Features and Integration

Not much is currently known about SAM on 4th generation devices (Surface Pro 4, Surface Book 1), due to the use of a different communication interface between host and EC (as detailed below). On 5th (Surface Pro 2017, Surface Book 2, Surface Laptop 1) and later generation devices, SAM is responsible for providing battery information (both current status and static values, such as maximum capacity etc.), as well as an assortment of temperature sensors (e.g. skin temperature) and cooling/performance-mode setting to the host. On the Surface Book 2, specifically, it additionally provides an interface for properly handling clipboard detachment (i.e. separating the display part from the keyboard part of the device), on the Surface Laptop 1 and 2 it is required for keyboard HID input. This HID subsystem has been restructured for 7th generation devices and on those, specifically Surface Laptop 3 and Surface Book 3, is responsible for all major HID input (i.e. keyboard and touchpad).

While features have not changed much on a coarse level since the 5th generation, internal interfaces have undergone some rather large changes. On 5th and 6th generation devices, both battery and temperature information is exposed to ACPI via a shim driver (referred to as Surface ACPI Notify, or SAN), translating ACPI generic serial bus write-/read-accesses to SAM requests. On 7th generation devices, this additional layer is gone and these devices require a driver hooking directly into the SAM interface. Equally, on newer generations, less devices are declared in ACPI, making them a bit harder to discover and requiring us to hard-code a sort of device registry. Due to this, a SSAM bus and subsystem with client devices (:c:type:`struct ssam_device <ssam_device>`) has been implemented.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\(linux-master)(Documentation)(driver-api)(surface_aggregator)overview.rst`, **line 32**); *backlink*
>
> Unknown interpreted text role "c:type".

Communication

The type of communication interface between host and EC depends on the generation of the Surface device. On 4th generation devices, host and EC communicate via HID, specifically using a HID-over-I2C device, whereas on 5th and later generations, communication takes place via a USART serial device. In accordance to the drivers found on other operating systems, we refer to the serial device and its driver as Surface Serial Hub (SSH). When needed, we differentiate between both types of SAM by referring to them as SAM-over-SSH and SAM-over-HID.

Currently, this subsystem only supports SAM-over-SSH. The SSH communication interface is described in more detail below. The HID interface has not been reverse engineered yet and it is, at the moment, unclear how many (and which) concepts of the SSH interface detailed below can be transferred to it.

Surface Serial Hub

As already elaborated above, the Surface Serial Hub (SSH) is the communication interface for SAM on 5th- and all later-generation Surface devices. On the highest level, communication can be separated into two main types: Requests, messages sent from host to EC that may trigger a direct response from the EC (explicitly associated with the request), and events (sometimes also referred to as notifications), sent from EC to host without being a direct response to a previous request. We may also refer to requests without response as commands. In general, events need to be enabled via one of multiple dedicated requests before they are sent by the EC.

See Documentation/driver-api/surface_aggregator/ssh.rst for a more technical protocol documentation and Documentation/driver-api/surface_aggregator/internal.rst for an overview of the internal driver architecture.