API for USB Type-C Alternate Mode drivers

Introduction

Alternate modes require communication with the partner using Vendor Defined Messages (VDM) as defined in USB Type-C and USB Power Delivery Specifications. The communication is SVID (Standard or Vendor ID) specific, i.e. specific for every alternate mode, so every alternate mode will need a custom driver.

USB Type-C bus allows binding a driver to the discovered partner alternate modes by using the SVID and the mode number.

:ref:`USB Type-C Connector Class <typec>` provides a device for every alternate mode a port supports, and separate device for every alternate mode the partner supports. The drivers for the alternate modes are bound to the partner alternate mode devices, and the port alternate mode devices must be handled by the port drivers.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 16); *backlink***
>
> Unknown interpreted text role "ref".

When a new partner alternate mode device is registered, it is linked to the alternate mode device of the port that the partner is attached to, that has matching SVID and mode. Communication between the port driver and alternate mode driver will happen using the same API.

The port alternate mode devices are used as a proxy between the partner and the alternate mode drivers, so the port drivers are only expected to pass the SVID specific commands from the alternate mode drivers to the partner, and from the partners to the alternate mode drivers. No direct SVID specific communication is needed from the port drivers, but the port drivers need to provide the operation callbacks for the port alternate mode devices, just like the alternate mode drivers need to provide them for the partner alternate mode devices.

Usage:

General

By default, the alternate mode drivers are responsible for entering the mode. It is also possible to leave the decision about entering the mode to the user space (See Documentation/ABI/testing/sysfs-class-typec). Port drivers should not enter any modes on their own.

`->vdm` is the most important callback in the operation callbacks vector. It will be used to deliver all the SVID specific commands from the partner to the alternate mode driver, and vice versa in case of port drivers. The drivers send the SVID specific commands to each other using :c:func:`typec_altmode_vdm()`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 46); *backlink***
>
> Unknown interpreted text role "c:func".

If the communication with the partner using the SVID specific commands results in need to reconfigure the pins on the connector, the alternate mode driver needs to notify the bus using :c:func:`typec_altmode_notify()`. The driver passes the negotiated SVID specific pin configuration value to the function as parameter. The bus driver will then configure the mux behind the connector using that value as the state value for the mux.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 51); *backlink***
>
> Unknown interpreted text role "c:func".

NOTE: The SVID specific pin configuration values must always start from `TYPEC_STATE_MODAL`. USB Type-C specification defines two default states for the connector: `TYPEC_STATE_USB` and `TYPEC_STATE_SAFE`. These values are reserved by the bus as the first possible values for the state. When the alternate mode is entered, the bus will put the connector into `TYPEC_STATE_SAFE` before sending Enter or Exit Mode command as defined in USB Type-C Specification, and also put the connector back to `TYPEC_STATE_USB` after the mode has been exited.

An example of working definitions for SVID specific pin configurations would look like this:

```
enum {
    ALTMODEX_CONF_A = TYPEC_STATE_MODAL,
    ALTMODEX_CONF_B,
    ...
};
```

Helper macro `TYPEC_MODAL_STATE()` can also be used:

```
#define ALTMODEX_CONF_A = TYPEC_MODAL_STATE(0);
#define ALTMODEX_CONF_B = TYPEC_MODAL_STATE(1);
```

Cable plug alternate modes

The alternate mode drivers are not bound to cable plug alternate mode devices, only to the partner alternate mode devices. If the

alternate mode supports, or requires, a cable that responds to SOP Prime, and optionally SOP Double Prime messages, the driver for that alternate mode must request handle to the cable plug alternate modes using :c:func:`typec_altmode_get_plug()`, and take over their control.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 84); *backlink*

Unknown interpreted text role "c:func".

Driver API

Alternate mode structs

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 97)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/usb/typec_altmode.h
   :functions: typec_altmode_driver typec_altmode_ops
```

Alternate mode driver registering/unregistering

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 103)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/usb/typec_altmode.h
   :functions: typec_altmode_register_driver typec_altmode_unregister_driver
```

Alternate mode driver operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 109)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/usb/typec/bus.c
   :functions: typec_altmode_enter typec_altmode_exit typec_altmode_attention typec_altmode_vdm typec_altmo
```

API for the port drivers

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 115)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/usb/typec/bus.c
   :functions: typec_match_altmode
```

Cable Plug operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\usb\[linux-master][Documentation][driver-api][usb]typec_bus.rst, line 121)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/usb/typec/bus.c
   :functions: typec_altmode_get_plug typec_altmode_put_plug
```