USB support in Zephyr OS

Zephyr Project Developer Summit 2021

Johann Fischer
johann.fischer@nordicsemi.no

Virtual / June 8, 2021

Contents I

Brief overview of the USB support in Zephyr OS

- USB Device support only
- Follows USB 2.0 Specification
- Support for well known USB classes
- Composite configuration support
- Combinations of user and built-in classes possible
- Full (FS) and high speed (HS) device drivers
- Poor high speed support
- Runtime reconfiguration not possible
- Only one device controller instance is possible

History

- Device stack is not written for Zephyr from scratch
- Import based on LPCUSB stack
- Originally without composite support
- Stack and classes revised several times
- Design has many weaknesses that are anything but easy to fix

Supported USB classes

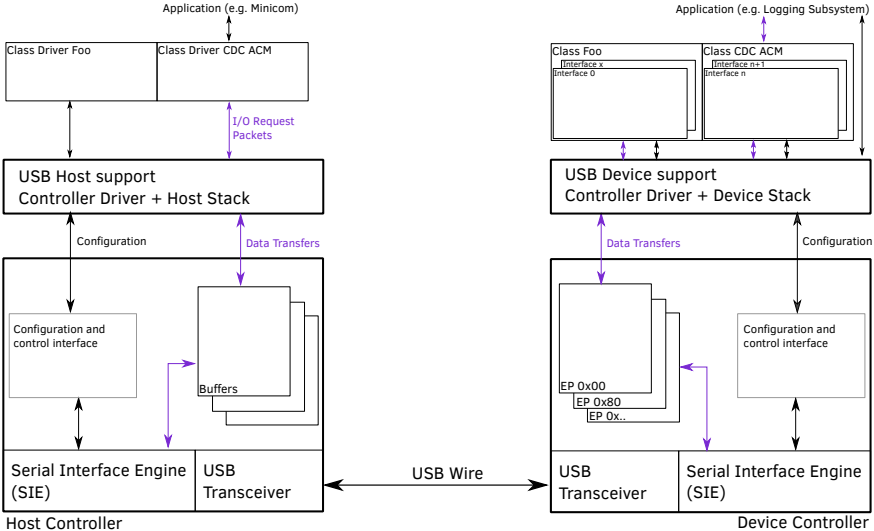
- Audio (still experimental)
- CDC ACM
- CDC ECM, EEM, RNDIS
- USB DFU (depends on DFU)
- USB HID
- Bluetooth HCI over USB
- Mass Storage Class (MSC)

USB classes can be combined with each other (composite configuration). It is only limited by the number of endpoints and host driver capabilities.

Relevant USB samples

- Multiple CDC ACM
- Shell over CDC ACM
- Console over CDC ACM
- Basic USB HID
- USB HID mouse + CDC ACM
- Accelerometer as HID mouse
- MSC with ramdisk, flash, SDMMC
- Audio
- USB DFU + MCUboot
- zperf with ECM, EEM, RNDIS
- Bluetooth HCI over USB
- USB 802.15.4 radio adapter
- WebUSB
- testusb

USB support overview



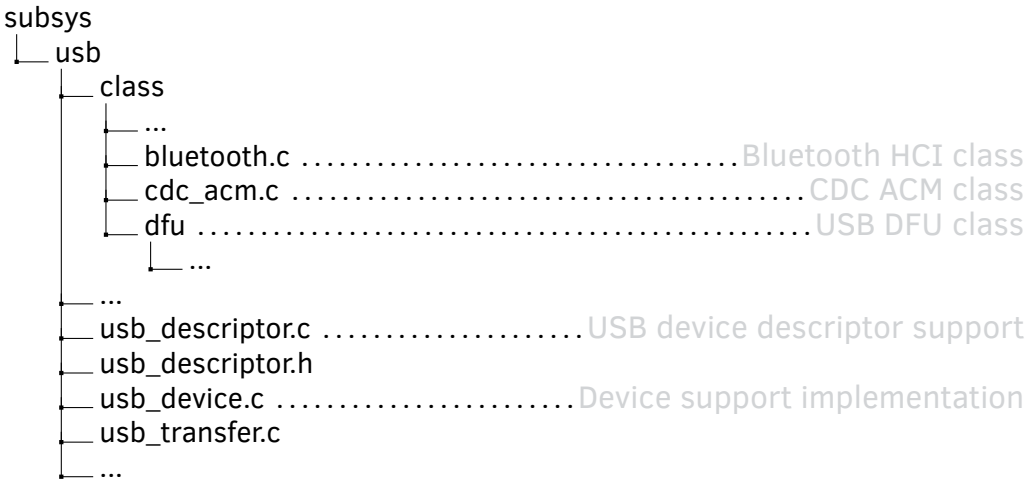
USB device controller driver organization

```
include
└── drivers
    └── usb
        └── usb_dc.h ..... USB device controller driver API
```

```
drivers
└── usb
    └── CMakeLists.txt
        └── device
            ├── CMakeLists.txt
            ├── Kconfig
            ├── usb_dc_dw.c ..... device controller driver
            ├── usb_dc_sam0.c ..... another driver
            ├── ...
            └── usb_dw_registers.h ..... driver header file
```

USB device support organization

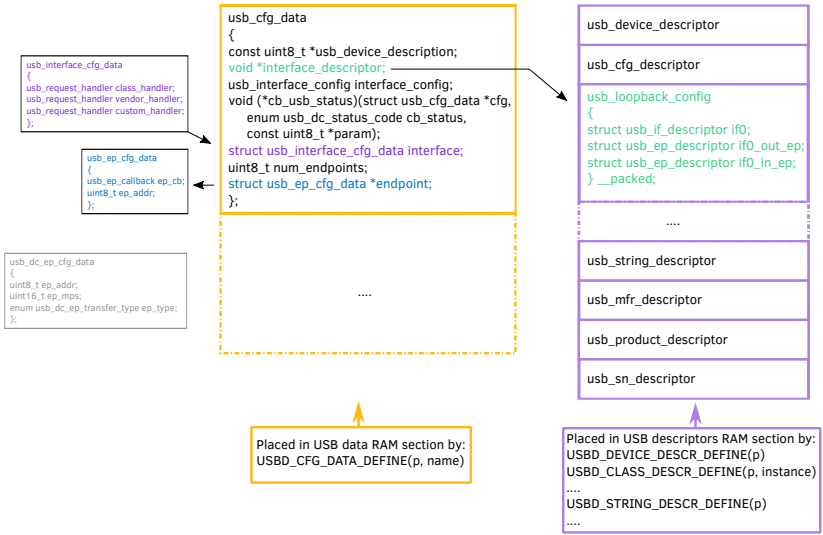
USB device support organization



Important USB device attributes

- Endpoints
  - Device endpoint is source or sink of data
  - Endpoint address consists of direction bit (IN,OUT) + number
  - Control endpoint are endpoints with the endpoint number 0
    - OUT - 0x00, IN - 0x80
- Configuration and Interfaces
  - At least one interface, with zero or more endpoints, within a configuration
  - At least one device configuration
- Descriptors
  - Device, Configuration, Interface, Endpoint
  - Completely describe a device

USB device support configuration overview



```
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x2fe3 NordicSemiconductor
  idProduct          0x0009
  bcdDevice            2.06
  iManufacturer           1 ZEPHYR
  iProduct                2 Zephyr testusb sample
  iSerial                 3 86FE679A598AC47A
  bNumConfigurations      1
  Configuration Descriptor:
    bLength              9
    bDescriptorType      2
    wTotalLength    0x0020
    bNumInterfaces       1
    bConfigurationValue  1
    iConfiguration       0
    bmAttributes      0xc0
      Self Powered
    MaxPower           100mA
    Interface Descriptor:
      bLength            9
      bDescriptorType    4
      bInterfaceNumber   0
      bAlternateSetting  0
      bNumEndpoints      2
      bInterfaceClass  255 Vendor Specific Class
      bInterfaceSubClass 0
      bInterfaceProtocol 0
      iInterface         0
      Endpoint Descriptor:
        bLength          7
        bDescriptorType  5
        bEndpointAddress 0x01  EP 1 OUT
        bmAttributes     2
          Transfer Type      Bulk
          Synch Type         None
          Usage Type         Data
        wMaxPacketSize   0x0040  1x 64 bytes
        bInterval        0
      Endpoint Descriptor:
        bLength          7
        bDescriptorType  5
        bEndpointAddress 0x81  EP 1 IN
        bmAttributes     2
          Transfer Type      Bulk
          Synch Type         None
          Usage Type         Data
        wMaxPacketSize   0x0040  1x 64 bytes
        bInterval        0
```

Token Packets

- SOF, SETUP, OUT, IN
- Only the host is allowed to send token packets
- SOF consist of PID, frame number and CRC
- SOF nominal rate is 1 ms for FS and 125 µs for HS
- SETUP, OUT, IN consist of PID, address + endpoint fields, CRC
- OUT - data transaction to a function (device)
- IN - data transaction to the host
- SETUP signals control transfers
- SETUP - function (device) must accept the Data and respond ACK

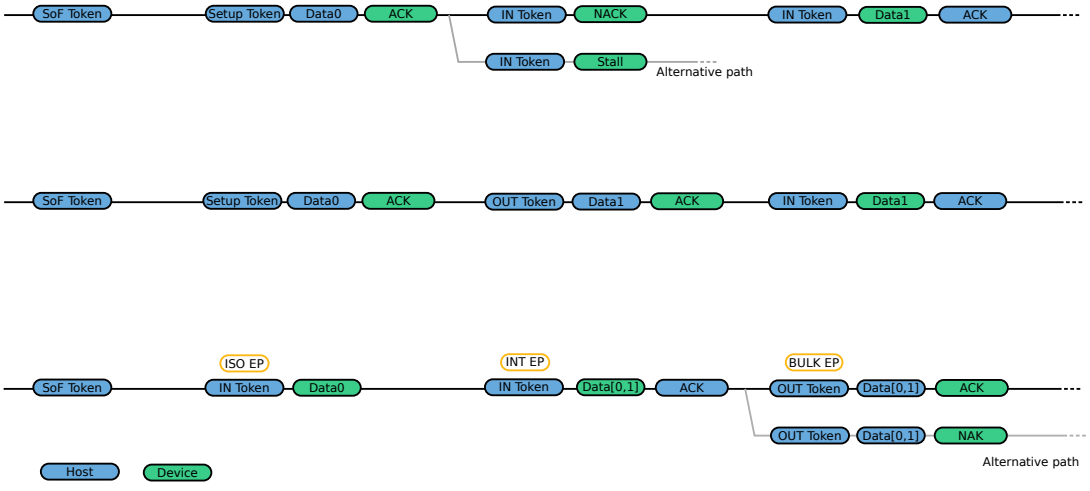
Data Packets

- DATA0, DATA1, (DATA2)
- consists of a PID, data field and CRC

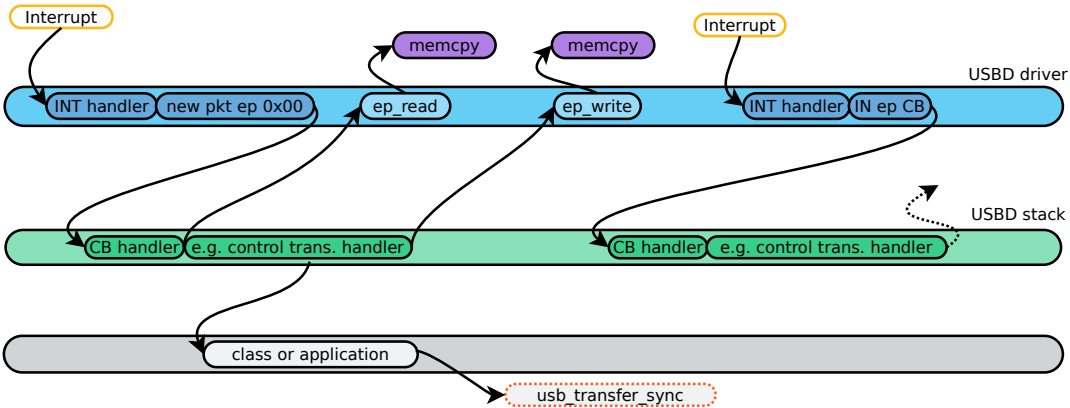
Handshake Packets

- ACK, NAK, STALL, (NYET)
- Host can never issue NAK
- NAK handshake for OUT - unable to accept data packet
- NAK handshake for IN - no data to transmit
- STALL - endpoint is halted or request is not supported
- Only the function (device) is allowed to send STALL

USB host device communication (simplified)



USB device stack driver interaction (simplified)



```c
/* Works, but it is not right */

static void x_move(const struct device *gpio,
                   struct gpio_callback *cb, uint32_t pins)
{
    ....

    status[MOUSE_X_REPORT_POS] = state;
    k_sem_give(&sem);
}

void main(void)
{
    ....

    ret = usb_enable(status_cb);
    if (ret != 0) {
        return;
    }

    while (true) {
        k_sem_take(&sem, K_FOREVER);

        report[MOUSE_X_REPORT_POS] = status[MOUSE_X_REPORT_POS];
        ret = hid_int_ep_write(hid_dev, report, sizeof(report), NULL);
        if (ret) {
            LOG_ERR("HID write error, %d", ret);
        }
    }
}
```

Summary of the drawbacks

Driver API

- Unreasonable event codes, e.g. USB_DC_CONFIGURED
- No proper support for endpoint configuration
- No support for multiple driver instances
- Confusing usb_dc_ep_read() / usb_dc_ep_write() API
- No support for endpoint requests/buffer management

Summary of the drawbacks

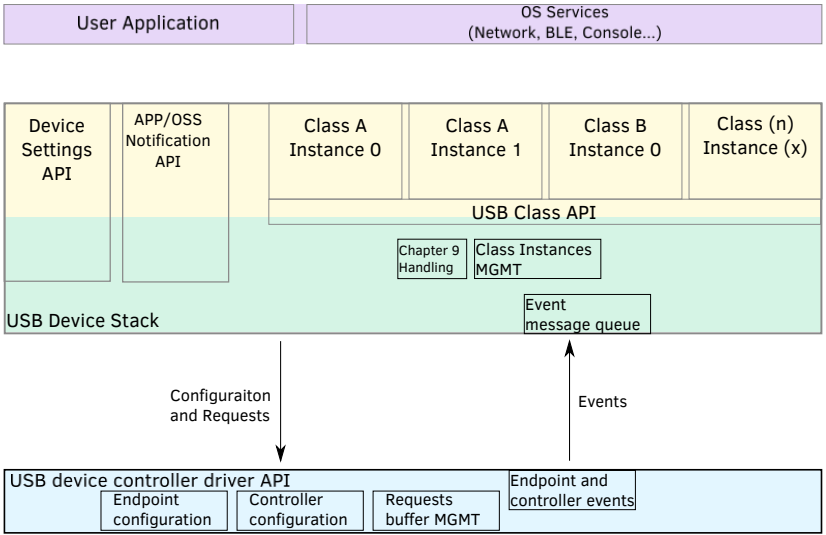
USB device stack

- Messy callback architecture
- No proper management of the class instances
- Lack of the possibility to configure class instances at runtime
- No API to set idVendor, iSerialNumber, bcdDevice ... at runtime
- Not a good notificaton channel to the application

What is going on in USB development?

- We are rewriting USB device controller driver API and USB device stack
- Drafts are finished and will be published soon
- The plan is to have new device support in parallel with the current one for a certain period of time

New USB device support



Other necessary improvements

USB Mass Storage Class

- Imported mbed implementation
- Should be completely rewritten, preferably with a separate SCSI layer

USB DFU Class

- Depends on DFU image manager and partition layout
- Should be more flexible and not depend on specific bootloader
- Rework probably only makes sense for new device stack

tests/subsys/usb

- Descriptors and driver API tests
- No functional tests, API test superficial

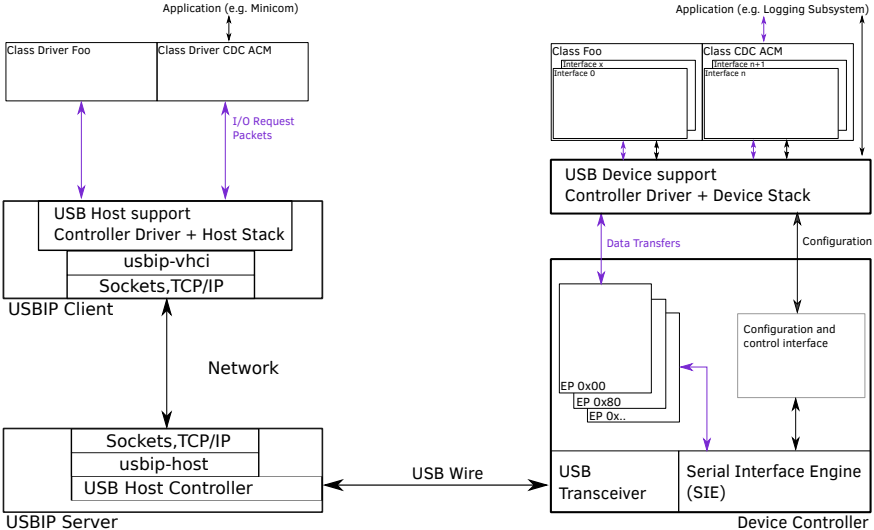
Linux Kernel testusb tool

- samples/subsys/usb/testusb
- Control and Bulk transfers only (limited by the implementation in Zephyr)
- Well suited for testing both driver and stack

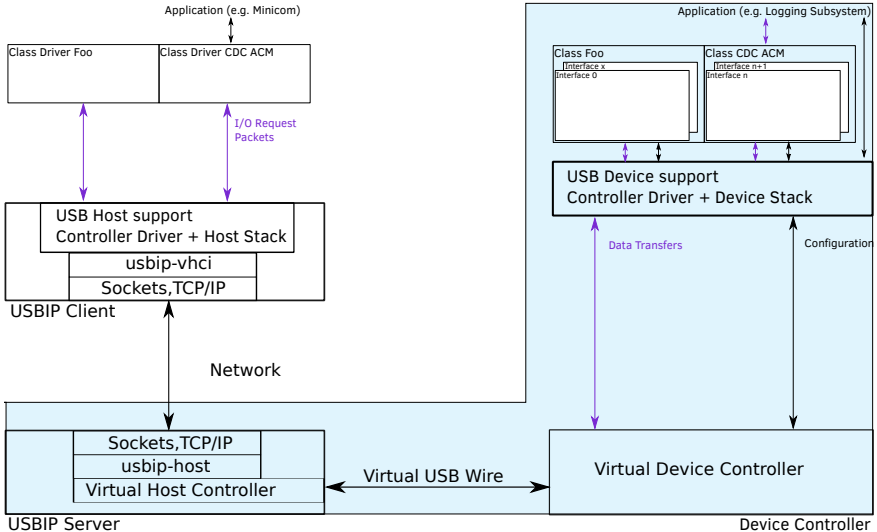
USBIP Protocol

- Passes USB device from server to client over TCP
- Description and implementation available in Linux kernel[2] [3]
- Current implementation in Zephyr can only be built for native_posix platform and is currently being revised
- Can be used to enable USB support for the platforms without USB controller
- Basically it needs USB host controller API for clean implementation
- Intended approach to test USB device controller API and device stack

USBIP Overview



USBIP vudc


Application (e.g. Minicom)
Class Driver Foo
Class Driver CDC ACM
I/O Request Packets
USB Host support
Controller Driver + Host Stack
usbip-vhci
Sockets,TCP/IP
USBIP Client
Network
Sockets,TCP/IP
usbip-host
Virtual Host Controller
USBIP Server
Virtual USB Wire
Application (e.g. Logging Subsystem)
Class Foo
Interface x
Interface 0
Class CDC ACM
Interface n+1
Interface n
USB Device support
Controller Driver + Device Stack
Data Transfers
Configuration
Virtual Device Controller
Device Controller

USB Host support

- Still not supported
- Draft PR (#30361) [1]
- USBIP allows very good test coverage for both Device and Host support
- Regardless of the Host support progress, partial functionalities will come as part of Device support testing

Questions?

[1] https://github.com/zephyrproject-rtos/zephyr/pull/30361.

[2] *USBIP Protocol*. https://elixir.bootlin.com/linux/latest/source/Documentation/usb/usbip_protocol.rst.

[3] *USBIP Tool*. https://elixir.bootlin.com/linux/latest/source/tools/usb/usbip/README.