USB Device Support in Zephyr RTOS From the Application Perspective

EOSS 2023

Johann Fischer
johann.fischer@nordicsemi.no

Prague, Czech Republic / June 28, 2023

Contents I

Devicetree

```
zephyr_udc0: &usbd {
    status = "okay";
};
```

Kconfig options and configuration files

```
# prj.conf
CONFIG_USB_DEVICE_STACK=y

CONFIG_USB_DEVICE_VID=0x2FE3
CONFIG_USB_DEVICE_PID=0x1234

CONFIG_USB_DEVICE_MANUFACTURER="Nice company"
CONFIG_USB_DEVICE_PRODUCT="Good product"
CONFIG_USB_DEVICE_SN="0123456789ABCDEF"

CONFIG_USB_SELF_POWERED=n
CONFIG_USB_MAX_POWER=123
...
```

Runtime code for configuration and activation

```c
/* main.c */

#include <zephyr/kernel.h>
#include <zephyr/sys/printk.h>
#include <zephyr/usb/usb_device.h>

static bool configured;

static void status_cb(enum usb_dc_status_code status,
                      const uint8_t *param)
{
    if (status == USB_DC_RESET) {
        configured = false;
    }

    if (status == USB_DC_CONFIGURED && !configured) {
        configured = true;
    }
}

int main(void)
{
    return usb_enable(status_cb);
}
```

Useful Kconfig options

CONFIG_USB_COMPOSITE_DEVICE=n

Function → Interface
"Use class code info from Interface Descriptors"[3]

bDeviceClass = 0		bInterfaceClass, e.g USB_BCC_MASS_STORAGE
bDeviceSubClass = 0	→	bInterfaceSubClass, e.g. SCSI_TRANSPARENT_SUBCLASS
bDeviceProtocol = 0		bInterfaceProtocol, e.g. BULK_ONLY_TRANSPORT_PROTOCOL

CONFIG_USB_COMPOSITE_DEVICE=y

"Use Interface Association Descriptor code triple"[1]
"IAD Descriptor is used to describe that two or more interfaces are associated to the same function."[2]

bDeviceClass = 0xEF		bInterfaceClass or IAD bFunctionClass
bDeviceSubClass = 0x02	→	bInterfaceSubClass or IAD bFunctionSubClass
bDeviceProtocol = 0x01		bInterfaceProtocol or IAD bFunctionProtocol

Useful Kconfig options

CONFIG_USB_DEVICE_INITIALIZE_AT_BOOT

Intended for use with CDC-ACM Console (Snippet).

CONFIG_USB_DEVICE_BOS

Enable BOS descriptors handling, used by the WebUSB sampe.
bcdUSB → 0210

CONFIG_USB_DEVICE_OS_DESC

Enable OS descriptors handling, used by the RNDIS network function.

- How to enable new USB device support in Zephyr RTOS?

```
# usbd_next_prj.conf
CONFIG_USB_DEVICE_STACK_NEXT=y
...
```

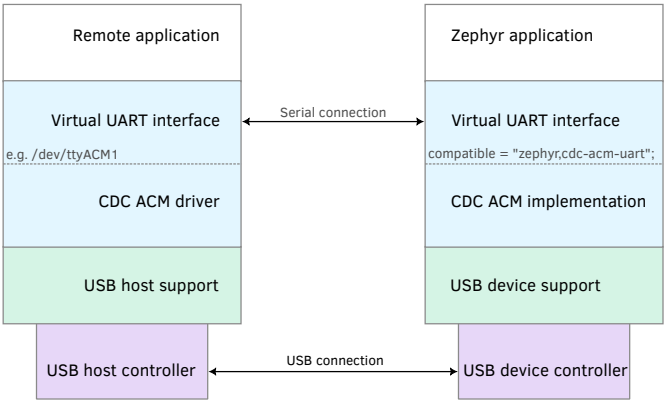
```c
/* main.c */

#include <zephyr/usb/usbd.h>

/* Use helper macro to define at least one configuration */
USBD_CONFIGURATION_DEFINE(foo_config,
                          USB_SCD_SELF_POWERED,
                          200);

/* Use helper macros to define string descriptors */
USBD_DESC_LANG_DEFINE(foo_lang);
USBD_DESC_MANUFACTURER_DEFINE(foo_mfr, "Nice company");
USBD_DESC_PRODUCT_DEFINE(foo_product, "Good product");
USBD_DESC_SERIAL_NUMBER_DEFINE(foo_sn, "0123456789ABCDEF");

/* Use helper macro to define device instance and
    descriptor */
USBD_DEVICE_DEFINE(foo_usbd,
                   DEVICE_DT_GET(DT_NODELABEL(zephyr_udc0)),
                   0x2fe3, 0xffff);
```

```c
int main(void)
{
    usbd_add_descriptor(&foo_usbd, &foo_lang);
    usbd_add_descriptor(&foo_usbd, &foo_mfr);
    usbd_add_descriptor(&foo_usbd, &foo_product);
    usbd_add_descriptor(&foo_usbd, &foo_sn);

    usbd_add_configuration(&foo_usbd, &foo_config);

    usbd_register_class(&foo_usbd, "foo_0", 1);

    usbd_init(&foo_usbd);

    return usbd_enable(&foo_usbd);
}
```

CDC ACM UART

- UART driver API used by various subsystems for communication
- There are three types of UART API: polling, interrupt driven and ASYNC.
- CDC ACM implementation provides a virtual UART interface
- It is desirable for the emulation to behave like the real interface
- CDC ACM implementation supports polling and interrupt driven APIs

Overview


Remote application
Zephyr application
Virtual UART interface
e.g. /dev/ttyACM1
Serial connection
Virtual UART interface
compatible = "zephyr,cdc-acm-uart";
CDC ACM driver
CDC ACM implementation
USB host support
USB device support
USB host controller
USB connection
USB device controller

Instantiation in devicetree

```
/ {
        chosen {
                zephyr,console = &cdc_acm_uart0;
        };
};

&zephyr_udc0 {
        cdc_acm_uart0: cdc_acm_uart0 {
                compatible = "zephyr,cdc-acm-uart";
        };
};
```

Kconfig options

```
CONFIG_SERIAL=y
CONFIG_UART_LINE_CTRL=y
CONFIG_CONSOLE=y
CONFIG_UART_CONSOLE=y
```

Application code

```
int main(void)
{
        const struct device *const dev = DEVICE_DT_GET(DT_CHOSEN(zephyr_console));
        uint32_t dtr = 0;

        if (usb_enable(NULL)) {
                return 0;
        }

        /* Wait for Data Terminal Ready signal */
        while (!dtr) {
                uart_line_ctrl_get(dev, UART_LINE_CTRL_DTR, &dtr);
                k_sleep(K_MSEC(100));
        }

        printk("Hello World!\n");

        return 0;
}
```

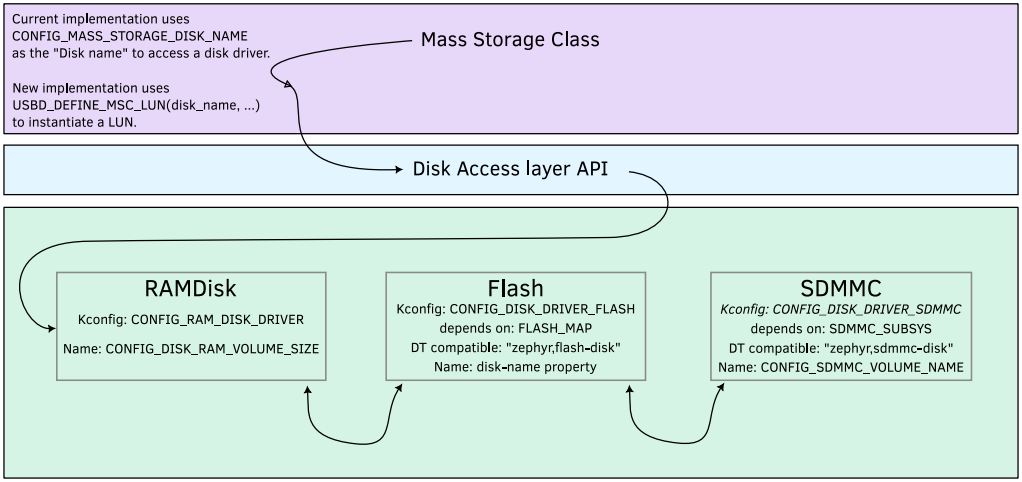
CDC ACM UART

- The communication is completely determined by the host
- No application accesses the host side, no IN/OUT tokens
- If the USB device is not configured, data passed to uart_poll_in() or uart_fifo_fill() will be discarded
- If in configured state, uart_poll_in() discards character from the tail
- Design application for the UART API, but not for the CDC ACM UART implementation

Mass Storage Class (MSC)

- Implementations for current and new USB device support
- CONFIG_USB_MASS_STORAGE | CONFIG_USBD_MSC_CLASS
- New implementation supports multiple LUN instances
- Both implementations use the Disk Access subsystem

Mass Storage Class and Disk Access



```
/*
 * Block device on a flash partition,
 * devicetree overlay
 */

&mx25r64 {
    partitions {
        compatible = "fixed-partitions";
        #address-cells = <1>;
        #size-cells = <1>;

        storage_partition: partition@0 {
            label = "storage";
            reg = <0x00000000 0x00020000>;
        };
    };
};

/ {
    msc_disk0 {
        compatible = "zephyr,flash-disk";
        partition = <&storage_partition>;
        disk-name = "NAND";
        cache-size = <4096>;
    };
};
```

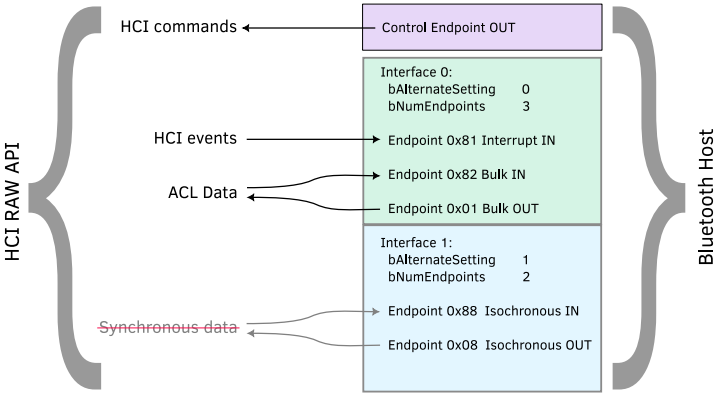
```
#
# Block device on a flash partition,
# configuration file

CONFIG_USB_DEVICE_STACK=y
CONFIG_USB_DEVICE_PRODUCT="MSC example"
CONFIG_USB_DEVICE_INITIALIZE_AT_BOOT=n

CONFIG_FLASH=y
CONFIG_FLASH_MAP=y

CONFIG_DISK_DRIVER_FLASH=y

CONFIG_USB_MASS_STORAGE=y
CONFIG_MASS_STORAGE_DISK_NAME="NAND"
```

```
/*
 * MSC with SDMMC disk driver,
 * devicetree overlay.
 * There is magic in the code.
 */

&spi42 {
    cs-gpios = <&gpio0 17 GPIO_ACTIVE_LOW>;

    sdhc13: sdhc@0 {
        compatible = "zephyr,sdhc-spi-slot";
        reg = <0>;
        status = "okay";
        spi-max-frequency = <24000000>;
        mmc {
            compatible = "zephyr,sdmmc-disk";
            status = "okay";
        };
    };
};
```

```
#
# MSC with SDMMC disk driver,
# configuration file

CONFIG_USB_DEVICE_STACK=y
CONFIG_USB_DEVICE_PRODUCT="MSC example"
CONFIG_USB_DEVICE_INITIALIZE_AT_BOOT=n


CONFIG_USB_MASS_STORAGE=y
CONFIG_MASS_STORAGE_DISK_NAME="NAND"
```

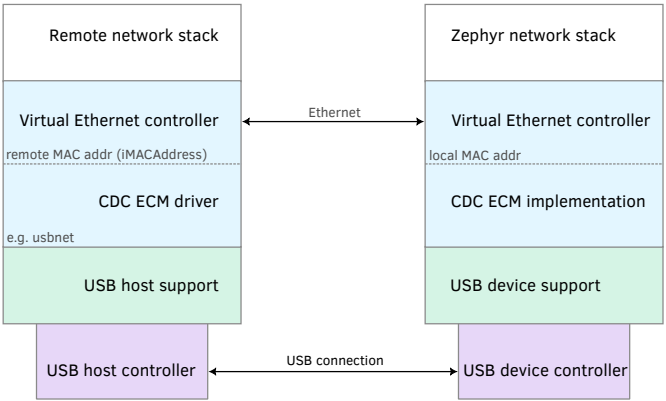
- Zephyr supports Bluetooth Low Energy Controller (LE Link Layer)
- No support for BR/EDR SCO
- HCI RAW API exposes HCI interface to the remote
- Bluetooth HCI USB transport layer uses HCI RAW API
- Implementations for current and new USB device support
- Current implementation cannot be combined with other classes

Overview


HCI RAW API
HCI commands
Control Endpoint OUT
Interface 0:
bAlternateSetting 0
bNumEndpoints 3
HCI events
Endpoint 0x81 Interrupt IN
ACL Data
Endpoint 0x82 Bulk IN
Endpoint 0x01 Bulk OUT
Interface 1:
bAlternateSetting 1
bNumEndpoints 2
Synchronous data
Endpoint 0x88 Isochronous IN
Endpoint 0x08 Isochronous OUT
Bluetooth Host

- CDC ECM, CDC EMM, and RNDIS support
- New USB device support has only CDC ECM implementation yet
- Virtual Ethernet connection between remote and Zephyr network support

Overview


Remote network stack
Zephyr network stack
Virtual Ethernet controller
remote MAC addr (iMACAddress)
Ethernet
Virtual Ethernet controller
local MAC addr
CDC ECM driver
e.g. usbnet
CDC ECM implementation
USB host support
USB device support
USB host controller
USB connection
USB device controller

Configuration

CDC ECM

Hardcoded local MAC address, remote MAC address can be set using the CONFIG_USB_DEVICE_NETWORK_ECM_MAC option.

New CDC ECM

MAC addresses can be set via remote-mac-address (required) and local-mac-address node properties, ETHERNET_CONFIG_TYPE_MAC_ADDRESS is also supported.

CDC EEM and RNDIS

Nothing to configure, hardcoded remote and local MAC addresses.
Applications using RNDIS support should enable CONFIG_USB_DEVICE_OS_DESC.

HID overview

- Implementation for new USB device support is WIP
- Current HID implementation supports multiple instances
- And abuses Device Driver Model
- No HID device API as such, but interface provided by hid_ops
- Instantiation using LISTIFY macro and CONFIG_USB_HID_DEVICE_COUNT

- Each instance requires a HID report descriptor ...
- and interrupt IN endpoint, OUT endpoint optional

```c
#include <zephyr/usb/usb_device.h>
#include <zephyr/usb/class/usb_hid.h>

#define REPORT_ID                1
static bool configured;
static const struct device *hdev;

static void int_in_ready_cb(const struct device *dev)
{
    static uint8_t report[2] = {REPORT_ID, 0};

    if (hid_int_ep_write(dev, report,
                         sizeof(report), NULL)) {
            LOG_ERR("Failed to submit report");
    } else {
            report[1]++;
    }
}

static const struct hid_ops my_ops = {
        .int_in_ready = int_in_ready_cb,
};
```

```c
static const uint8_t hid_report_desc[] = {
    HID_USAGE_PAGE(HID_USAGE_GEN_DESKTOP),
    HID_USAGE(HID_USAGE_GEN_DESKTOP_UNDEFINED),
    HID_COLLECTION(HID_COLLECTION_APPLICATION),
    HID_LOGICAL_MIN8(0x00),
    HID_LOGICAL_MAX16(0xFF, 0x00),
    HID_REPORT_ID(REPORT_ID),
    HID_REPORT_SIZE(8),
    HID_REPORT_COUNT(1),
    HID_USAGE(HID_USAGE_GEN_DESKTOP_UNDEFINED),
    HID_INPUT(0x02),
    HID_END_COLLECTION,
};
```

```
static void status_cb(enum usb_dc_status_code status,
                      const uint8_t *param)
{
    if (status == USB_DC_RESET) {
        configured = false;
    }
    if (status == USB_DC_CONFIGURED && !configured) {
        int_in_ready_cb(hdev);
        configured = true;
    }
}

int main(void)
{
    int ret;

    hdev = device_get_binding("HID_0");
    if (hdev == NULL) {
        return -ENODEV;
    }

    usb_hid_register_device(hdev, hid_report_desc,
                            sizeof(hid_report_desc), &my_ops);

    ret = usb_hid_init(hdev);
    if (ret) {
        return ret;
    }

    return usb_enable(status_cb);
}
```

Output reports via the OUT interrupt endpoint

Enable CONFIG_ENABLE_HID_INT_OUT_EP and provide int_out_ready callback to receive output reports.

Caution

USB HID Kconfig options apply to all instances!

- Follows USB Audio specification version 1.00
- Supports synchronous synchronisation type only
- Supports multiple instances
- Abuses Device Driver Model
- No USB Audio device API as such, but interface provided by usb_audio_ops
- Instantiation using LISTIFY macro and devicetree

Audio device types

Headphones

compatible = ”usb-audio-hp”;

Microphone

compatible = ”usb-audio-mic”;

Headset

compatible = ”usb-audio-hs”;

- Tightly coupled to Zephyr Device Firmware Upgrade and MCUBoot API

Troubleshooting

Disable CONFIG_BOOTLOADER_MCUBOOT.

Questions?

[1] USB-IF. *USB Association Descriptor ECN code triple*.
https://www.usb.org/defined-class-codes#anchor_BaseClassEFh.

[2] USB-IF. *USB ECN: Interface Association Descriptor*.
https://www.usb.org/document-library/usb-20-specification.

[3] USB-IF. *USB null class code triple*.
https://www.usb.org/defined-class-codes#anchor_BaseClass00h.