Mikrovezérlő programozás

ARM® mbed™ környezetben

13. USB soros és HID kommunikáció

Az USB-ről röviden

USB = Univerzális soros busz (**U**niversal **S**erial **B**us)

Szabvány leírása: www.usb.org

- USB 1.0 - 1995 1,5 Mbit/s
- **USB 2.0 - 2000 1,5/12/480 Mbit/s** (Full speed = 12 Mbit/s)
- USB 3.0 - 2009 super speed: 5 Gbps (további 2 érpár felhasználásával)

Minden eszköznek jeleznie kell, hogy milyen sebességű átvitelre képes:

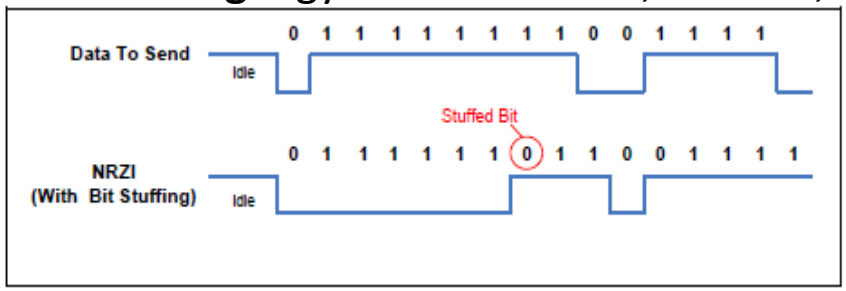
- Az alacsony sebességű eszközök a D- vonalat húzzák fel.
- A teljes sebességű eszközök a D+ vonalat húzzák fel 3,3 V-ra egy 1,5 k Ohm-os ellenállással.
- A nagysebességű eszközök induláskor teljes sebességű eszközként azonosítják magukat, s később, a host-tal történő egyeztetés után kapcsolnak nagysebességű üzemmódba.

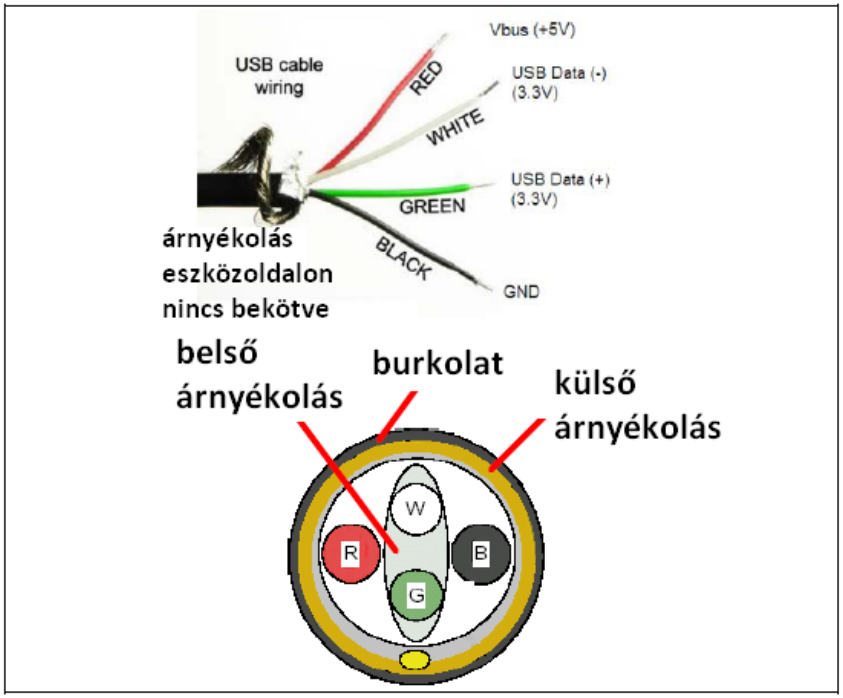
Az USB fizikai megvalósítása

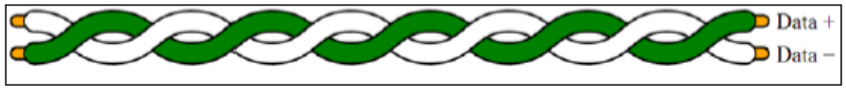
NRZI (Non Return to Zero Inverted) – a jel polaritást vált, ha az adatbit 0.

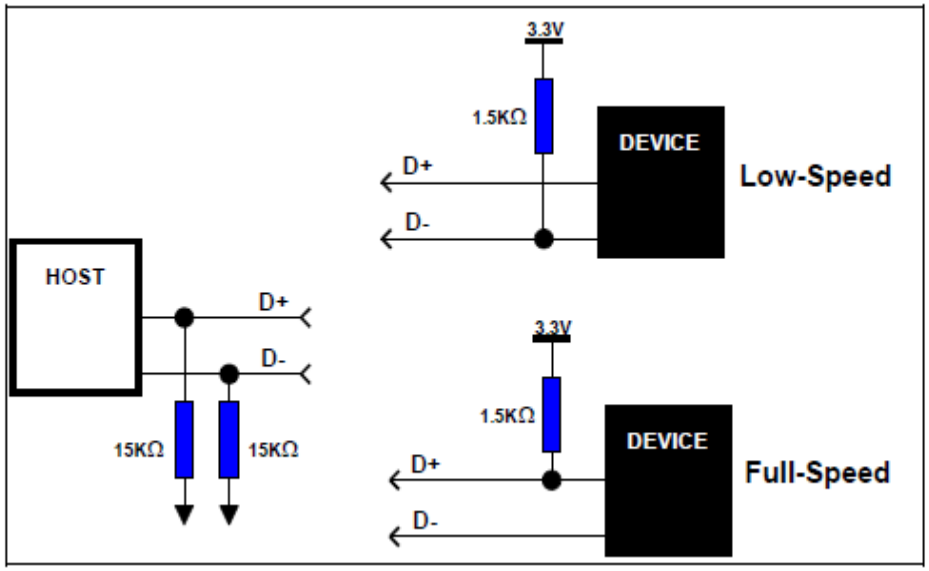
Bit stuffing: Egy 0 bit beszúrása, ha több, mint 6 db 1-es jön egymás után.

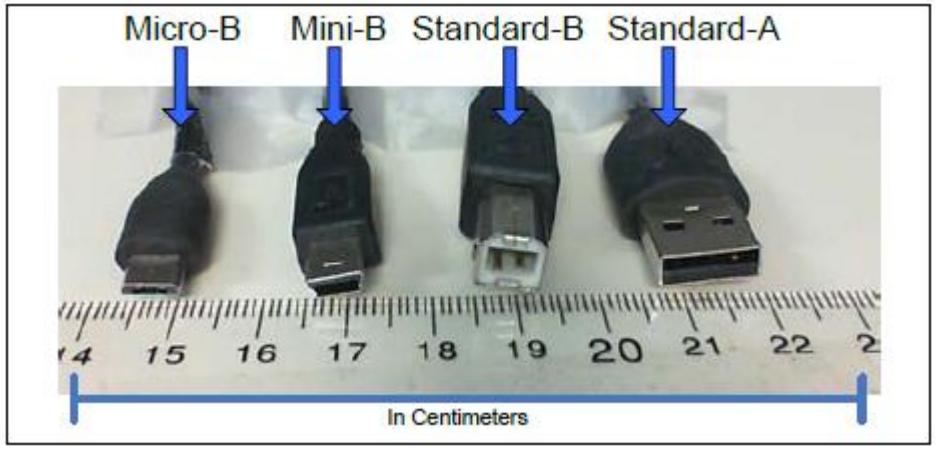
Forrás: Cypress AN57294



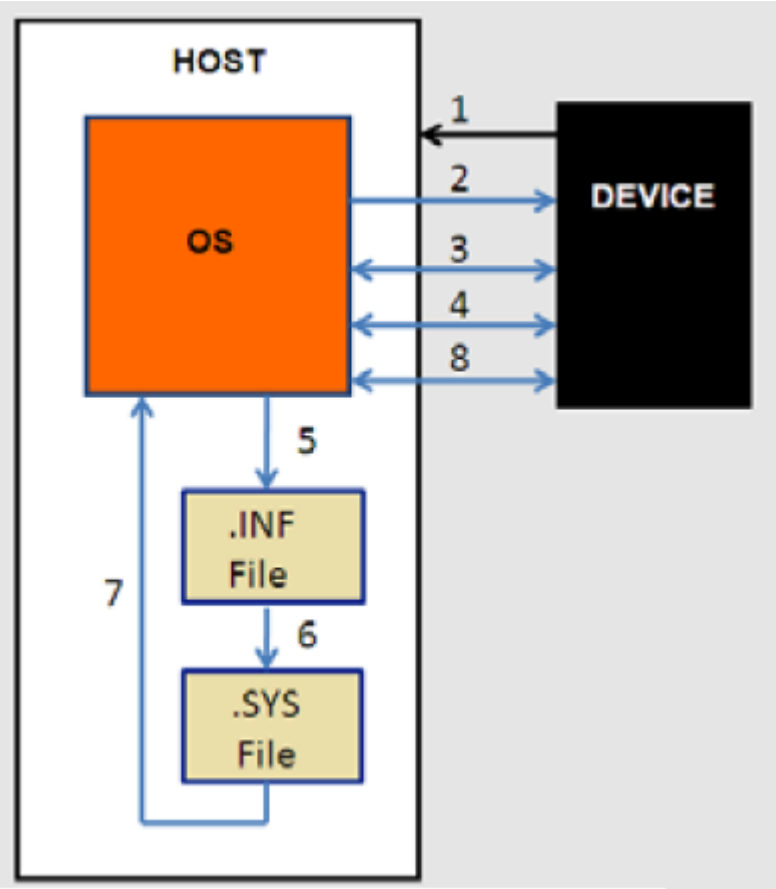








USB számbavétel (enumeration)

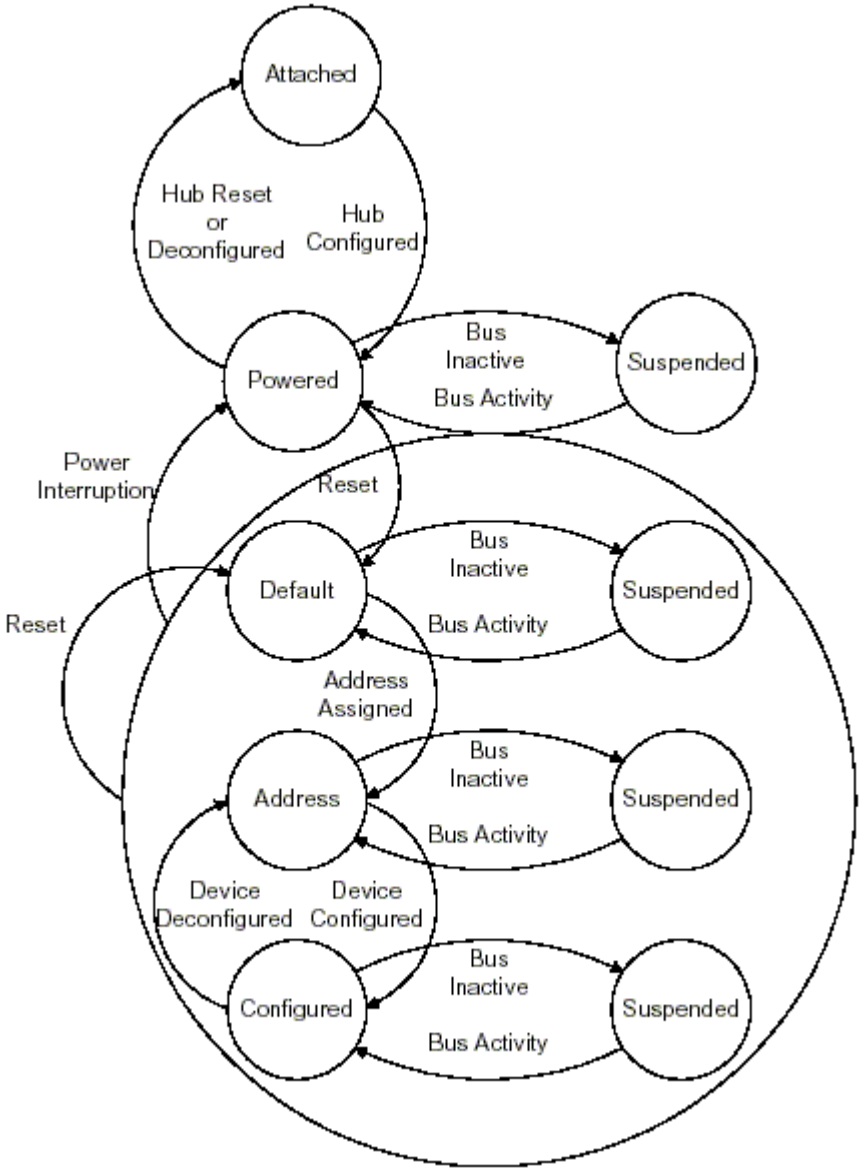


1. Az eszköz csatlakozik a gazdagéphez
2. A gazdagép reset parancsot küld az eszköznek
3. Az eszköz válaszol a kérésre és a gazdagép új címet állít be az eszkönek.
4. A gazdagép eszközleírót kér az új címmel azonosított eszköztől. Az eszköz válaszol
5. A gazdagép megkeresi és olvassa az .INF állományt
6. Az .INF specifikálja az eszköz meghajtóját
7. Az eszközmeghajtó betölése
8. A gazdagépkiválasztja az eszköz konfigurációját.

Az eszköz konfigurált és kész a használatra

Az USB termékek azonosítására szolgáló VID, PID számpárost az eszköz gyártója adja meg.
VID = gyártó azonosítója (az USB-IF-től vásárolható), PID = termékazonosító (gyártó adja)

USB eszköz állapotdiagramja

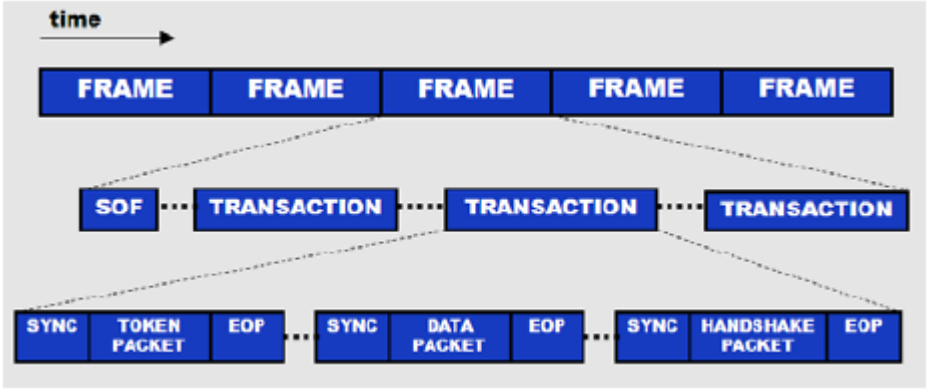


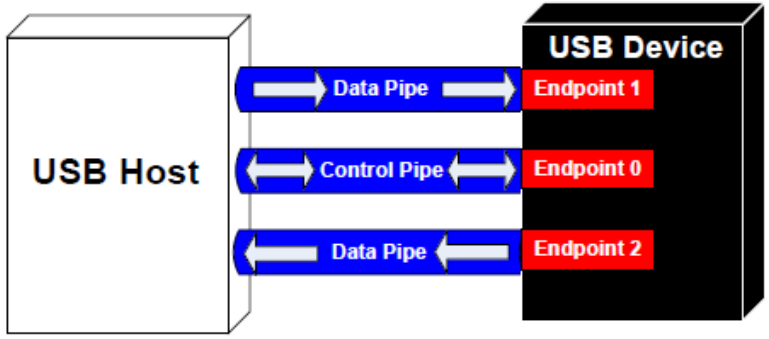
Attached: csatlakoztatott
Powered: tápellátással rendelkező
Default: alapértelmezett
Addressed: megcímzett
Configured: konfigurált
Suspended: felfüggesztett

Csak a „Configured” állapotban levő eszköz áll készen az adatküldésre, adatfogadásra.

USB kommunikációs protokol

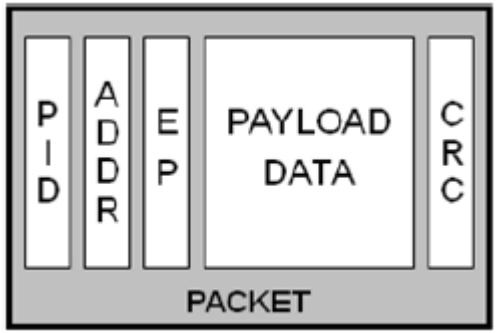
A kommunikáció keretekre van osztva (1 keret = 1 ms), a kereten belül tranzakciók zajlanak.
A kommunikáció a host és az általa megcímzett végpont között történik.
Minden eszközben kell egy vezérlő végpontnak is lennie (ez a nulladik sorszámú).





Adatcsomag felépítése:

- Packet ID (4 bit) lehet pl. IN, OUT, SET, SOF
- Eszköz cím (7 bit)
- Végpont cím (4 bit: max. 16 IN és max. 16 OUT)
- Adatblokk (max. 64/1024 bájt)
- CRC ellenőrző kód (5/16 bit)



Végpont/kommunikációs osztályok

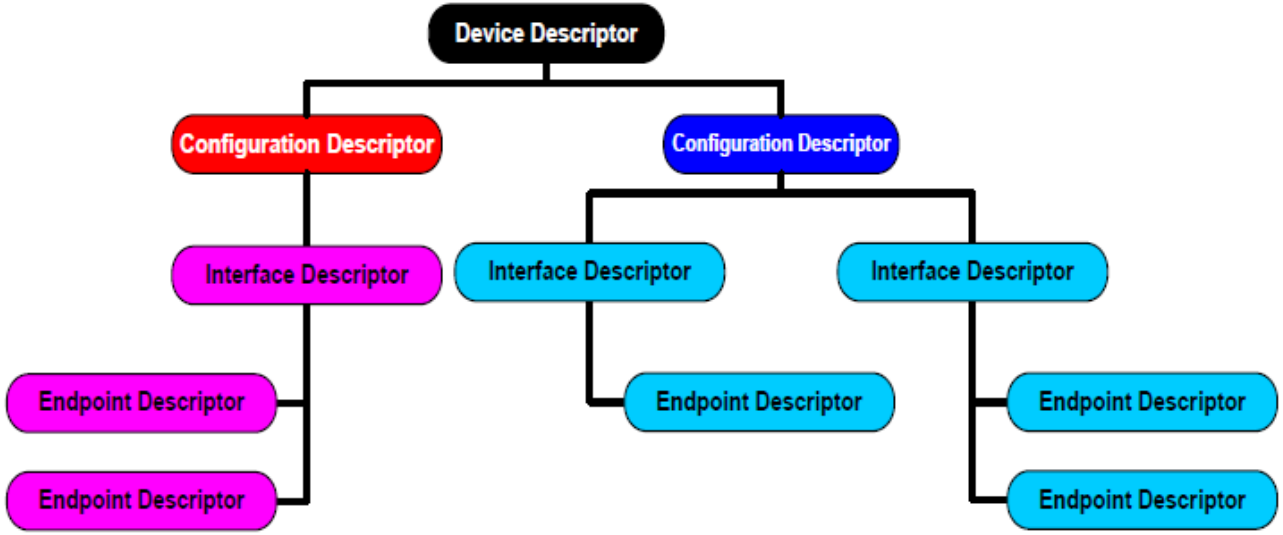
Az USB négy alapvető adatátviteli típust támogat:

- **Vezérlő (control):** minden eszköznek rendelkeznie kell egy vezérlő típusú végponttal. Ezen keresztül zajlik az USB enumeráció (enumeration) folyamata, amiről később esik szó.
- **Ömlesztett (bulk):** nagy mennyiségű, de nem időkritikus adat mozgatására szolgál. Ilyen pl. a **CDC** eszközosztály (Communication Device Class).
- **Megszakításos (interrupt):** kis késleltetésű átvitelt garantál. Pl. szabályos időnként küldendő adatot szokás mozgatni vele (**HID** = Human Interface Devices, pl. billentyűzet, egér).
- **Valós idejű (isochronus):** időkritikus folyamatos átvitelhez, mint pl. a hang-, és videó átvitel.

A 13. fejezet mintapéldáinál a CDC és a HID eszközosztályokkal fogunk találkozni.

Eszközleírók (descriptors)

Az eszköz-, konfiguráció-, interfész-, végpont- és egyéb leíró táblák az enumerációhoz szükséges adatokat tartalmazzák. Egy eszköz több konfigurációval és interfésszel is rendelkezhet.



Az USBDevice programkönyvtár

Az mbed kompatibilis mikrovezérlő, mint USB eszköz

USBMouse – egér, amely kurzort mozgat, kattintást és görgetést kezel

USBKeyboard – billentyűzetként kezeli a mikrovezérlőt

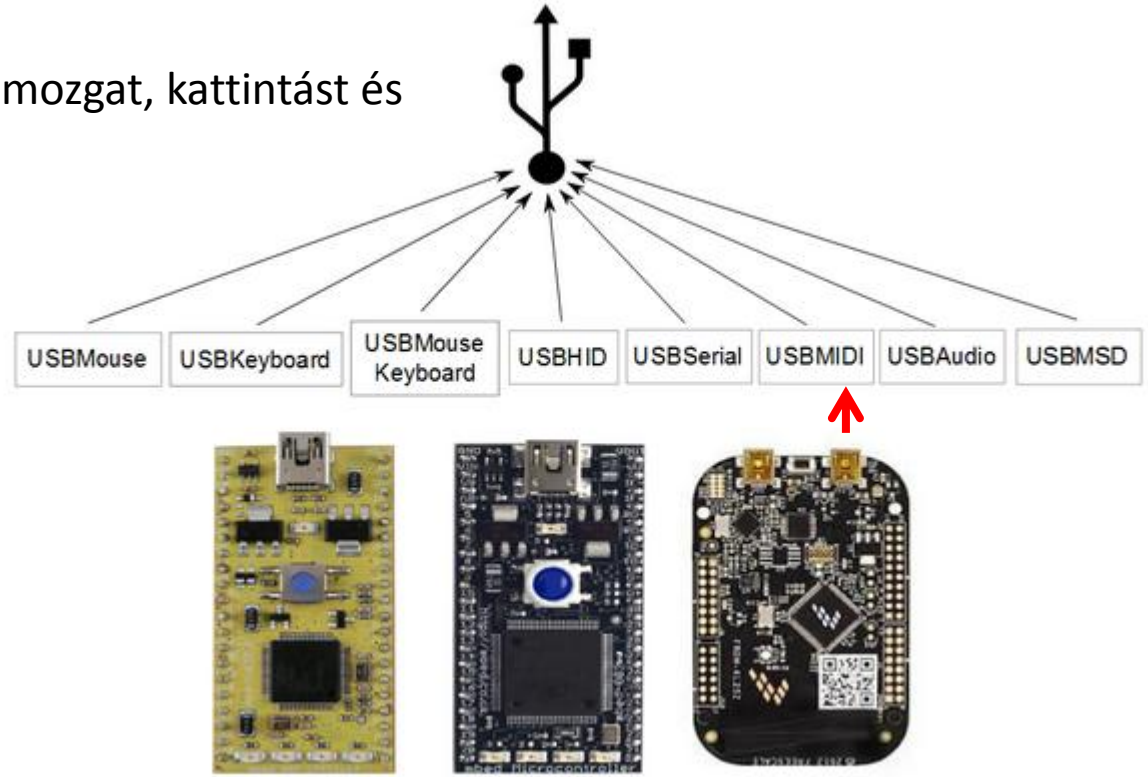
USBHID – adatküldés és fogadás nyers adatokkal.

USBSerial – virtuális soros port

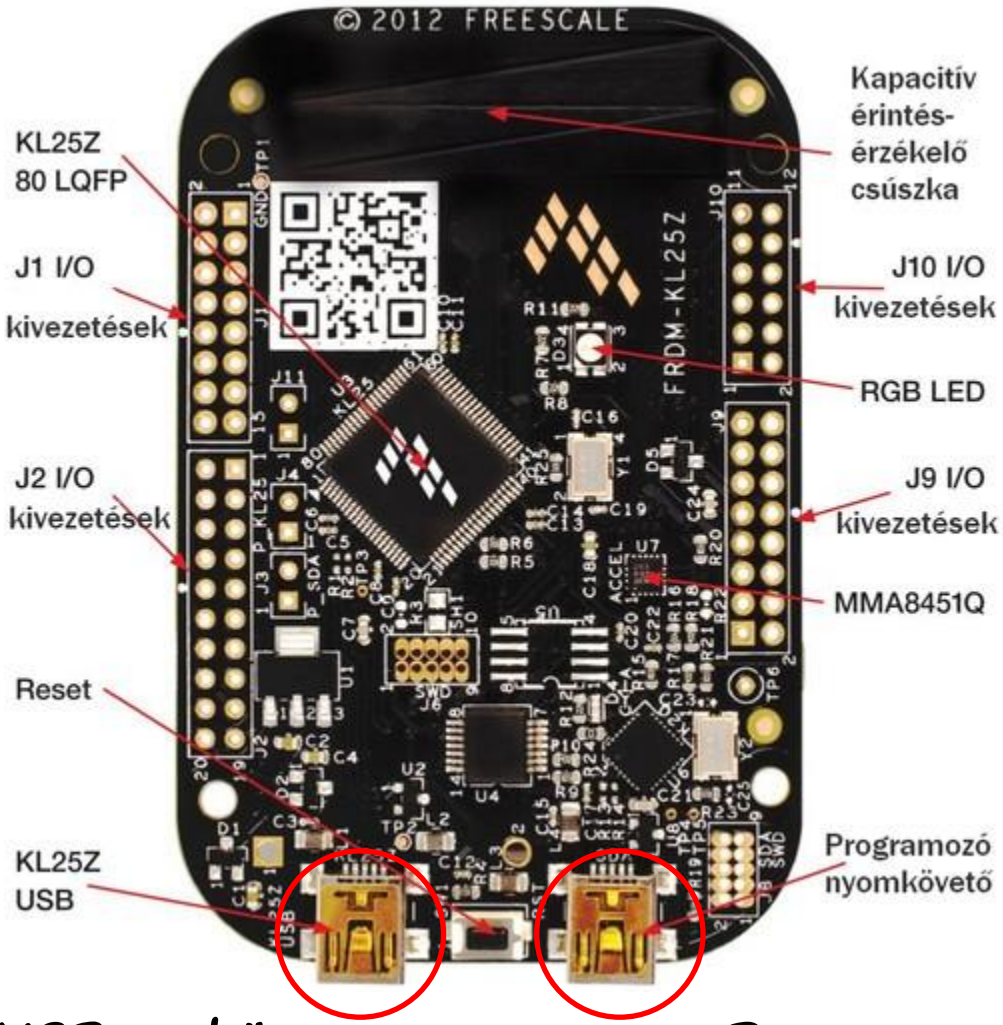
USBMIDI – MIDI eszközként használja a mikrovezérlőt

USBAudio – hangeszközként (forrás vagy lejátszó) kezeli a mikrovezérlőt

USBMSD – tömegtároló eszközosztály (mint pl. a Pendrive)



Az USB csatlakozók használata



USB eszköz
Közvetlen csatlakozás
a mikrovezérlőhöz

Programozó eszköz
Soros porton kommunikál a
mikrovezérlővel *(Serial class)*

USBSerial objektumosztály

USB "bulk transfer" kommunikáció, az **CDC (Character Device Class)** eszközosztályba tartozik. A **PC felől virtuális soros portnak látszik,** ezért a kapcsolat bármelyik olyan alkalmazással kezelhető, amely a soros kommunikációs portokat (**COM** portok) kezelni tudja.
Az **USBSerial** objektumosztályt az **USBDevice** progamkönyvtár része. Tagfüggvényei:

Függvénynév	Rövid leírás
USBSerial név(vid=0x1f00, pid=0x2012, release=0x0100, blocking=true)	Konstruktor függvény. Példányosítja és inicializálja az USBSerial objektumosztályt, s az opcionális paraméterek szerint módosítja a konfigurációt
putc(int c)	Egy karakter küldése a host felé, blokkoló típusú várakozással. (további lehetőségek: **puts()** vagy **printf()**
getc()	Egy karakter fogadása a host felől, blokkoló típusú várakozással.
writeBlock(*buf, size)	Adatblokk kiírása (max 64 karakter)
puts(s)	Szövegkonstans karakterfüzér kiíratása
gets(s,n)	*n* db karakter fogadása és eltárolása karakterfüzérként az *s* tömbbe
printf()	Formázott kiíratás
scanf()	Formázott szöveg beolvasása
available()	Fogadott karakterek száma
readable()	Megvizsgálja, hogy van-e beérkezett karakter
writeable()	Megvizsgálja, hogy van-e szabad hely a kimeneti tárban

A virtuális soros port felismeréséhez az erről a címről letölthető eszközmeghajtó is kell!

Kiíratás virtuális soros portra

Az alábbi, "helló világ" szintű programban az USB porton keresztül íratunk ki másodpercenként egy rövid szöveget.

Hardver követelmények:

- **FRDM-KL25z** kártya
- A **KL25Z USB** jelzésű aljzat csatlakoztatása a számítógéphez

A 13_USBSerial_HelloWorld/main.cpp program listája

```
#include "mbed.h"
#include "USBSerial.h"

USBSerial serial;       //Virtual serial port over USB
int main(void) {
    while(1) {
        serial.printf("I am a virtual serial port\r\n");
        wait(1);
    }
}
```

Az **USBSerial** felhasználásával végzett kommunikációnál a PC oldali terminál programban **nem kell beállítani az adatsebességet,** mert a virtuális soros port esetében ennek nincs szerepe. Az adatátvitel nem karakterenként, hanem keretezett USB üzenetcsomagonként történik (bulk transfer).

Az USB HID eszközosztály

A HID talán a legjobban támogatott eszközosztály a szabványosított USB eszközosztályok közül. Ide tartozik az USB egér, billentyűzet, botkormány, távirányító és sok más eszköz.

Tulajdonságok:

- Minden tranzakció vezérlő vagy megszakítás típusú átvitelt használ.
- Tranzakciónként legfeljebb 64 bájt vihető át.
- A maximális átviteli sebesség 1 tranzakció/ms, ami átszámítva legfeljebb 64 KB/s.
- Csak egy kiviteli és egy beviteli végpont használható (a vezérlő csatorna mellett).
- A gazdagép periodikusan kérdezi le a HID eszközt.

Általános (generic) HID eszközök

A szabványos HID eszközöknél az ún. HID report descriptor (HID jelentés leíró tábla) megmondja, hogy az átvitt adatcsomag egyes bájtjai (vagy bitjei) mit jelentenek.

Az általános (generic) HID eszközöknél a HID report descriptor csak az átvitt adatok mennyiségét mondja meg, azok értelmezése a mikrovezérlőbe töltött firmware-re és a PC-n futó alkalmazásra van bízva. Ebben az esetben tehát csak az átvitel mikéntje szabványos, az adatok felhasználása azonban gyártóspecifikus.

A HID átvitel használatának előnyei:

- Nem kell gyártóspecifikus meghajtó (az oprendszer eleve „tudja” kezelni az eszközt)
- Garantált átviteli időzítés

Az USBHID objektumosztály

Az USBHID objektumosztály az előzőekben említett "Általános HID Eszközt" (Generic HID Device) valósítja meg. A tagfüggvények segítségével üzeneteket küldhetünk és kaphatunk az USB buszon. Saját protokollt dolgozhatunk ki és eszerint kommunikálhatunk a mikrovezérlő és a számítógép között.

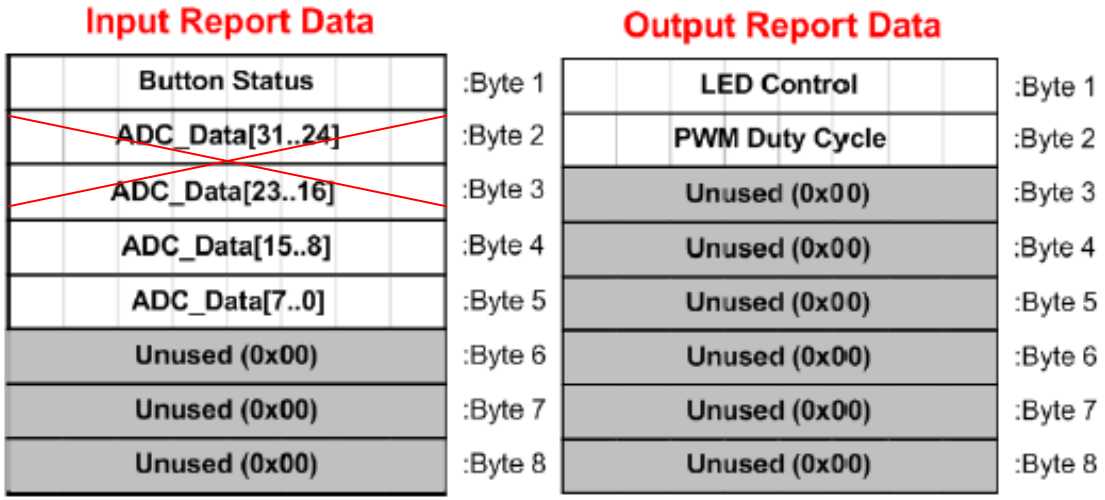
Függvénynév	Rövid leírás
USBHID (uint8_t output_report_length=64, uint8_t input_report_length=64, uint16_t vendor_id=0x1234, uint16_t product_id=0x0006, uint16_t product_release=0x0001, bool connect=true)	Konstruktor függvény. Példányosítja és inicializálja az USBHID objektumosztályt, megadja a kimenő és bejövő adatblokk méretét, a VID/PID azonosítót, s a connect paraméter értékétől függően kapcsolódik (vagy nem)
read(HID_REPORT* report)	Egy adatblokk fogadása blokkoló várakozással
readNB(HID_REPORT* report)	Egy adatblokk fogadása nem blokkoló módon
send(HID_REPORT* report)	Egy adatblokk küldése blokkoló várakozással
sendNB(HID_REPORT* report)	Egy adatblokk küldése nem blokkoló módon

Az **USBHID** használatához **a számítógépen kell olyan alkalmazás,** amely képes kezelni az **USBHID** adatforgalmat. Ez lehet akár egy szkript is, amihez használhatjuk a Python értelmező **pywinusb** kiegészítését. Egy másik lehetőség a **hidapi** könyvtár, amelyet C/C++ konzol alkalmazásokból vagy grafikus alkalmazásokból is használhatunk. Az USBHID bindings és az USBHID C bindigs oldalon találunk mintapéldákat mindkettő használatára.

USB HID mintapélda

Mintapéldánkat a Cypress AN82072 alkalmazási mintapéldájához igazítva készítettük el, hogy annak a letölthető ZIP csomagjában található grafikus PC alkalmazását használni tudjuk. Ennek megfelelően mindkét irányba 8 bájtos csomagokat küldünk.
Az Input/Output irány a gazdagép szempontjából értendő.
Input Report: a mikrovezérlő küldi, első bájtja a **D3** nyomógomb bemenet állapota, a következő 4 bájt az **A0** analóg csatorna ADC konverziójának eredménye.
Az **Output Report:** (a PC küldi) vezérlési funkciót lát el: az első bájt **LED1**-et vezérli (0: ki, 1: be), a második bájt pedig **LED2** fényerősségét vezérli (1 - 100 közötti értéket küldhetünk ki, ami a **PWM** jel százalékos kitöltését adja meg).

A 8 bájtos adatcsomagokat csak a firmware és a PC alkalmazás fogja tudni értelmezni! Az ADC adat nálunk 16 bites…



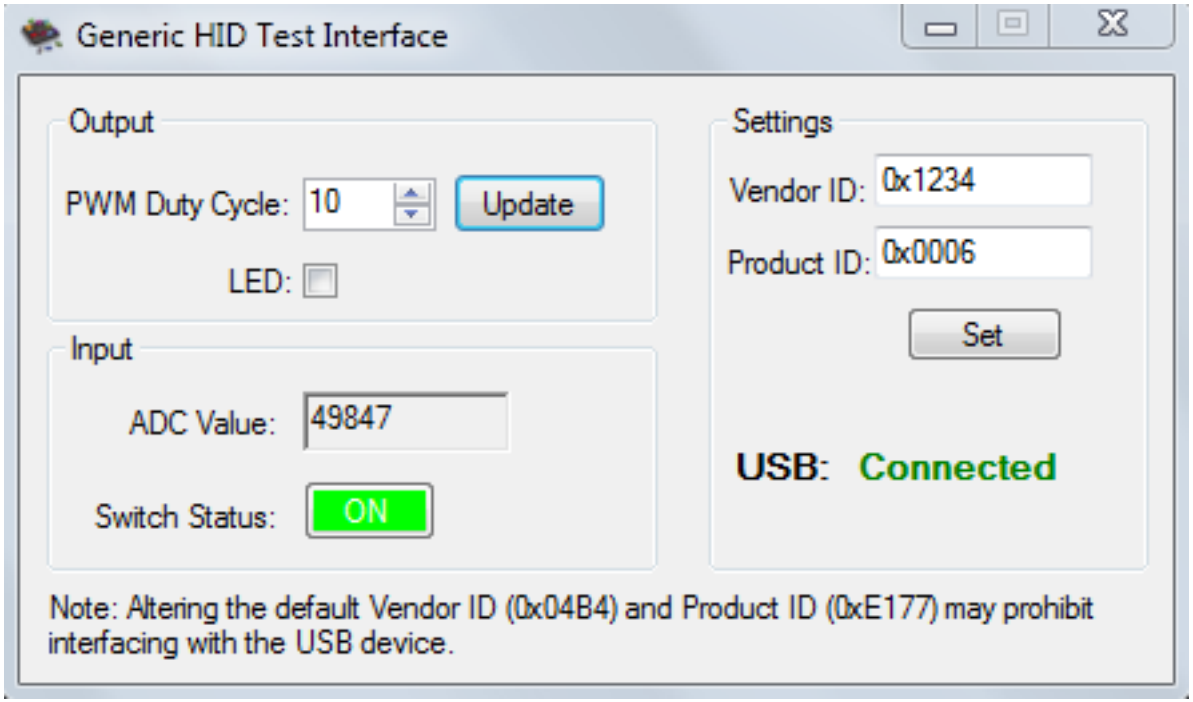
13_USBHID_demo/main.cpp

```
#include "mbed.h"
#include "USBHID.h"

//We declare a USBHID device. By default input and output reports are 8 bytes long.
USBHID hid(8, 8);

HID_REPORT send_report;       //This report will contain data to be sent
HID_REPORT recv_report;       //This report will contain data received

DigitalOut LED_1(LED1);
PwmOut LED_2(LED2);
DigitalIn SW1(D3,PullUp);
AnalogIn adc(A0);

int main(void) {
    send_report.length = 8;
    LED_1 = 1;
    LED_2.period_ms(20);

    while (1) {
        uint16_t raw = adc.read_u16();                  //Read ADC (A0 chan)
        for (int i = 0; i < send_report.length; i++)    //Fill the report
            send_report.data[i] = 0x00;
        send_report.data[0] = SW1.read();
        send_report.data[3] = (raw>>8);
        send_report.data[4] = (raw & 0xff);
        hid.send(&send_report);                         //Send the report

        if(hid.readNB(&recv_report)) {                  //try to read a msg
            LED_1 = !recv_report.data[0];
            LED_2.write(1.0 - recv_report.data[1]*0.01f);
        }
    }
}
```

A programban megmutatjuk, hogy az **USB HID** kapcsolat segítségével hogyan végezhetünk analóg és digitális adatgyűjtést/adatbeolvasást, és hogyan létesíthetünk digitális és (kvázi) analóg vezérlést.

A 13_USBHID_demo futtatása

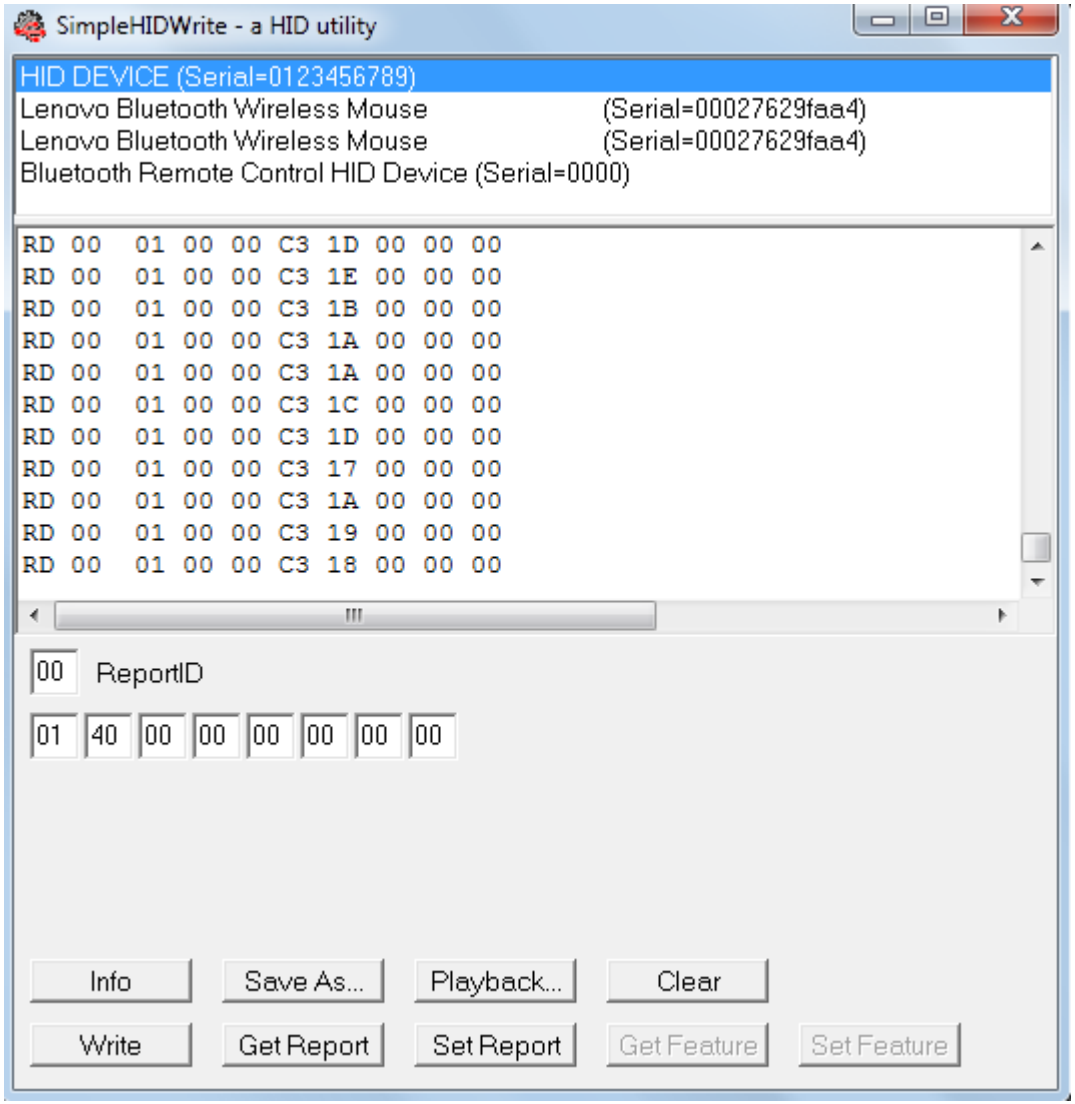
1. Be kell állítanunk a **VID** és **PID** azonosítókat és rá kell kattintanunk a **Set** gombra!
2. A piros LED-et a **LED** felirat melleti jelölőnégyzet kiválasztásával kapcsolható be és ki.
3. A zöld LED fényereje a **PWM Duty Cyle** címke mellett állítható be 1 és 100 közötti értékre, de csak az **Update** gomb kattintásakor lesz kiküldve.
4. A bejövő értékek (az ADC konverzió 16 bites eredménye és a **D3** digitális bemenet állapota) a vezérlőelemek alatt látható.



A SimpleHidWrite alkalmazás

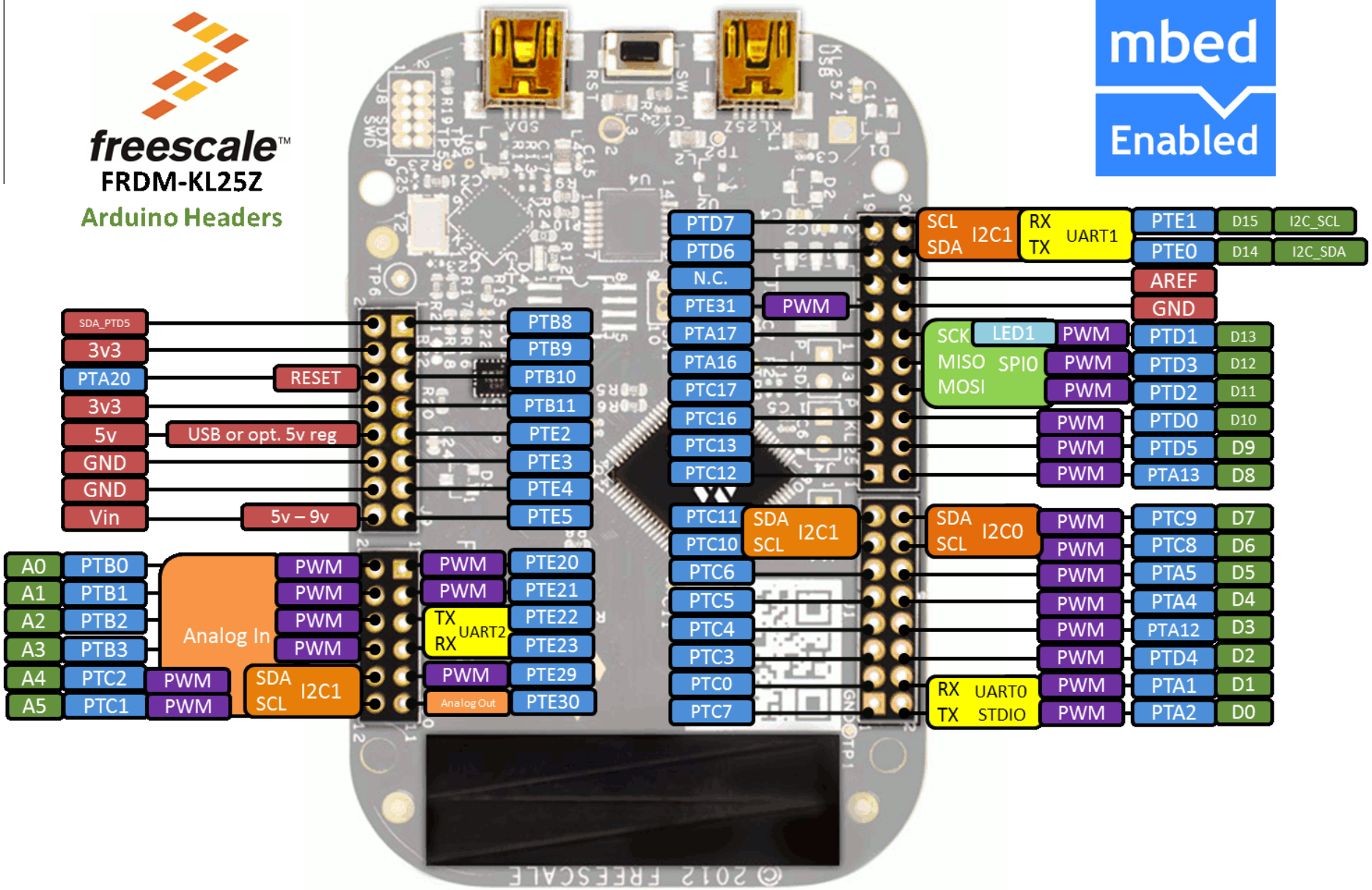
1. Töltsük le és bontsuk ki **SimpleHIDWrite3.zip** csomagot!
2. Indítsuk el a **SimpeHidWrite.exe** alkalmazást!
3. A megnyíló ablak felső részében kattintsunk a felismert HID eszközünk sorára (a **HID DEVICE (Serial=0123456789)** eszközt keressük)! A listázó ablakban ekkor elkezdenek pörögni a bejövő üzenetek. Figyeljünk rá, hogy az RD (olvasást) jelzést követő első szám még nem az üzenetcsomagunk része, hanem a jelentés alapértelmezett azonosítója (00)! Az ez követő bájt jelzi a nyomógombunk állapotát, a következő négy bájt pedig az ADC konverzió eredményét.
4. Állítsunk össze egy adatcsomagot! A Report ID kötelezően 0, az alatta levő 8 bájt pedig a kiküldenő adatok. Csak a nullától különböző mezőket kell kitölteni! Például:
   01 00 00 00 00 00 00 00 bekapcsolja a piros LED-et és leoltja a zöld LED-et,
   00 64 00 00 00 00 00 00 lekapcsolja a piros LED-et és teljes fényerőn (100 %-os kitöltés) kigyújtja a zöld LED-et.
5. Küldjük ki az adatcsomagot a **Write** gombra kattintva!
6. Mentsük el egy napló állományba a beérkezett és kiküldött csomagokat (**Save As...** gomb), majd tanulmányozzuk az adatokat! A kiküldött adatokat a **WD** kezdetű sorok tartalmazzák.

A SimpleHidWrite alkalmazás



freescale™
FRDM-KL25Z
Arduino Headers

mbed Enabled



freescale™

FRDM-KL25Z

Additional Peripherals