STM32 mikrovezérlők programozása STM32CubeIDE környezetben

10. Virtuális soros port – USB CDC eszközosztály

Felhasznált és ajánlott irodalom

- STMicroelectronics: STM32 USB training (27 video + letölthető anyagok)
  - Youtube lejátszási lista
  - Letölthető anyagok (ZIP fájl)
- Al Williams: Hints for using the CDC USB Serial
- Shawn Hymel: Getting Started with STM32 Nucleo USB

USB - Univerzális Soros Busz

- Az **USB** az 1990-es évek derekán kidolgozott ipari szabvány, amely az alábbiakat definiálja:
  - a busz architektúrája
  - vezetékek, csatlakozók, elektromos jelszintek
  - kommunikációs protokollok

USB Implementers Forum (USB-IF): https://usb.org/
- Specifikációk kidolgozása
- Megfelelőségi tesztek kidolgozása
- PID/VID azonosítók kiosztása

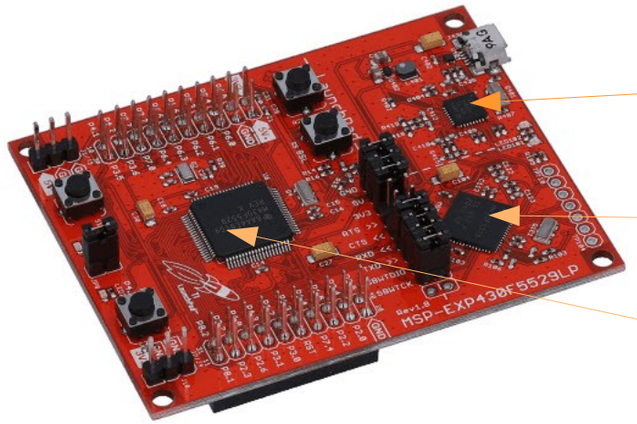
- Az USB szabványt a számítógép perifériák (billentyűzet, egér, egyéb mutatóeszköz, digitális kamerák, nyomtatók, hordozható médiaeszközök, háttértárolók, hálózati csatolók) egységes csatlakoztatására dolgozták ki
- Az USB lett a közös csatlakozófelülete más eszközöknek is (mobiltelefonok, táblagépek, játékkonzolok)
- Az USB gyakorlatilag leváltotta a korábbi csatlakozásokat (printer port, soros port)

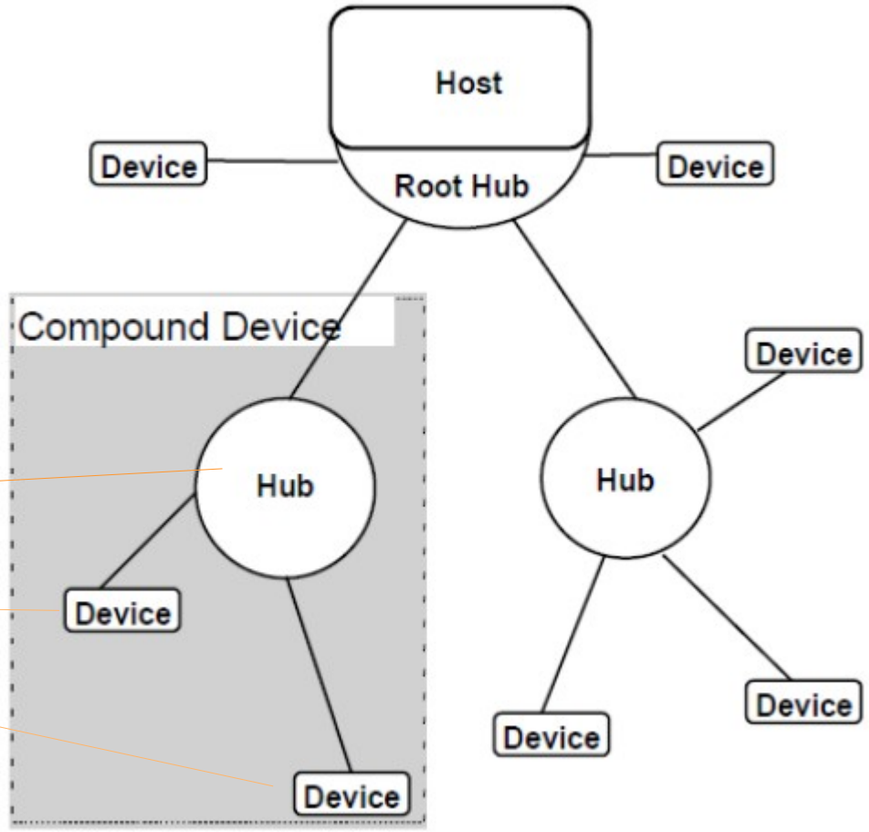
Az USB történetének mérföldkövei

- Az **USB 1.0** specifikációt **1996 januárjában** vezették be, amely **Low Speed** (1.5 Mbit/s) és **Full Speed** (12 Mbit/s) árviteli sebességet definiált
  - Az első széleskörűen alkalmazott verzió az **1998. szeptemberében** kiadott **USB 1.1**
  - A Low speed módot az USB perifériával nem rendelkező MCU-kra szánták (software emulation)
- **USB 2.0** specifikáció **2000. áprilisában**
  - Magasabb átviteli sebességet definiál elérve a 480 Mbit/s-ot, ami 40-szeres növekedés az USB 1.1-hez képest
- **USB 3.0** specifikáció **2008. novemberében**
  - Növekedés az adatátviteli sebességben (5 Gbit/s-ig)
  - kompatibilis az USB 2.0-vel: egy új, *SuperSpeed* nevű buszt valósít meg a 2.0 busszal párhuzamosan
- **USB 3.1** specifikáció **2013. júliusban**
  - Egy még gyorsabb, új átviteli mód: *„SuperSpeed USB 10 Gbps"*
- **USB 3.2** specifikáció **2017. szeptember 25.**
  - Továbbfejlesztett SuperSpeed busz 20 Gbps-ig
  - Két sávos működés az USB-C kábelen

USB busz topológia

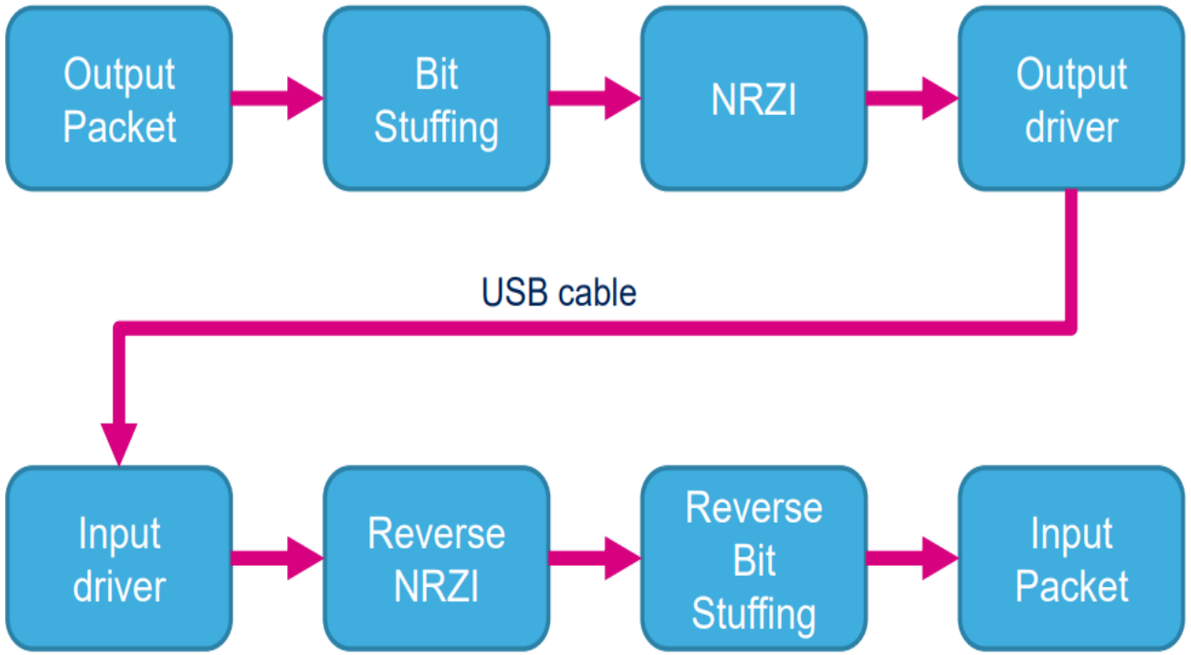
- Többszintű csillag topológia, amely az USB **eszközöket** a **host**-hoz kapcsolja
- A **hub** eszköz a csillagpontok elágazását kezeli
- Minden hálózati szegmens (vezeték) **pont-pont** kapcsolatot valósít meg
- Legfeljebb 127 eszköz kapcsolódhat a buszra
  - Minden eszközhöz egy 7-bites cím tartozik
  - A 0 cím fenntartott a még nem számbavett eszköz számára
- Legfeljebb 5 **hub** köthető sorba
- A maximális kábelhossz legfeljebb 5 m lehet
- Példa az összetett eszközre:
  Texas Instrument
  MSP-EXP430F5529LP
  USB LaunchPad kártya



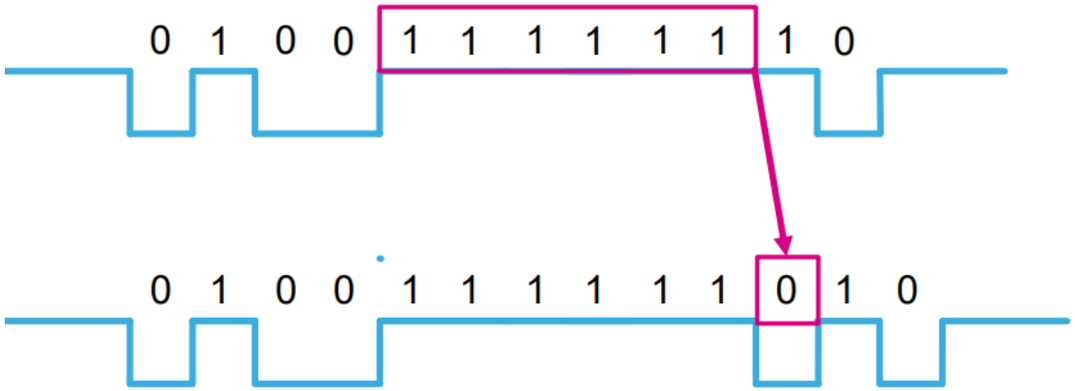


Fizikai réteg

- Az üzenetcsomagok **NRZI** *(non return to zero, inverted)* kódolással és szükség esetén bit beszúrással **(Bit Stuffing)** kerülnek kiküldésre
- Az adatfogalom differenciális adatvonalon folyik

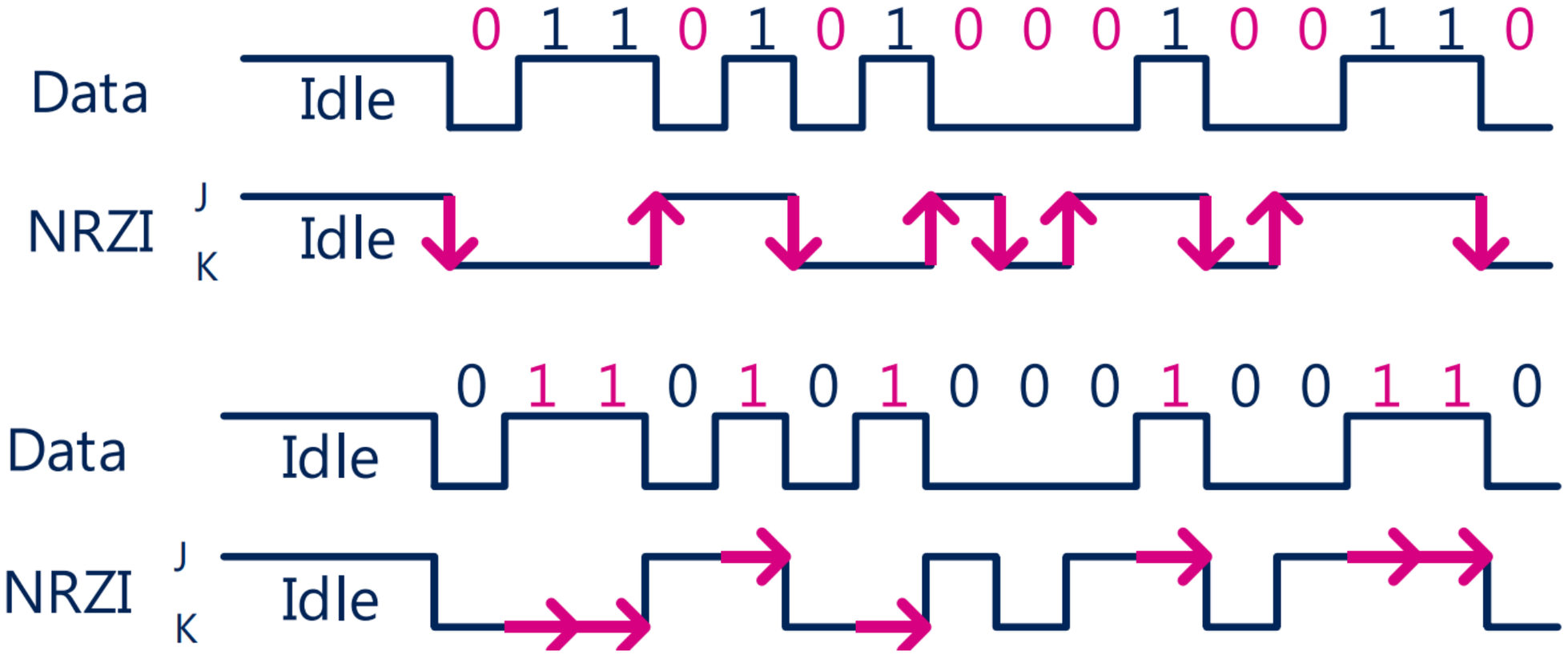


Bit beszúrás (Bit Stuffing): hat egymás utáni egyes után beszúrunk egy nullát, hogy legyen átmenet elegendő



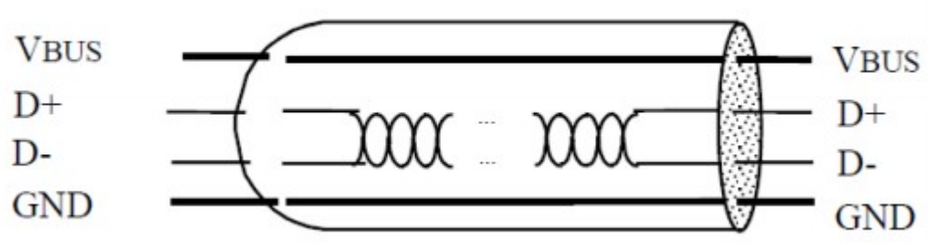
NRZI kódolás

- **NRZI** (non return to zero inverted)
  - Jelszintváltás ha az adatbit 0
  - Jelszint tartása, ha az adat 1



Elektromos jellemzők

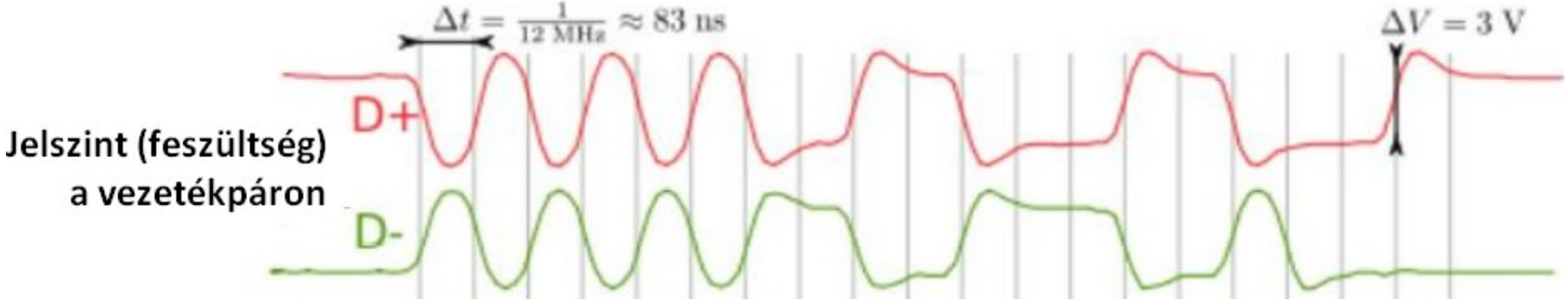
- Az **USB** soros busz, az USB 2.0 verzió négy árnyékolt vezetéket használ:
  - Két vezeték a tápellátásra (VBUS és GND)
  - Két adatvezeték (D3 és D-)



USB 1.x/2.0 standard pinout

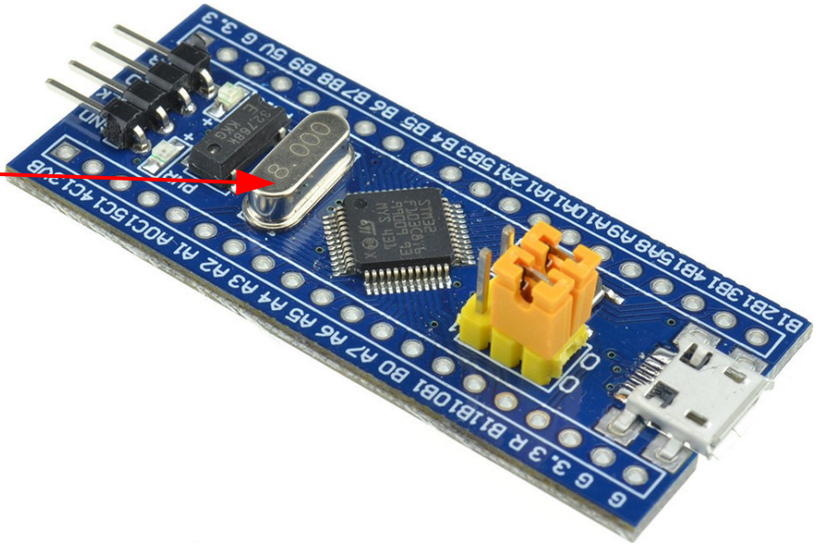
Pin	Name	Wire color	Description
1	V_{BUS}	Red (or Orange)	+5 V
2	D−	White (or Gold)	Data−
3	D+	Green	Data+
4	GND	Black (or Blue)	Ground

- Full speed példa:



Időzítés

- A host, a hubok és a nagy sebességű (high speed) adatküldésre képes USB eszközök esetében az adatküldési sebesség megkövetelt pontossága ± 0.05% (500 ppm)
- A full-speed eszközöknél az adatküldési sebesség 12 Mbit/s ± 0.25% (2500 ppm)
  - Bővebben lásd az USB specifikáció 7.1.11 – Data Signaling Rate fejezetében
- Fentiekből is látható, hogy az USB eszközöknél alapvetően fontos az órajel pontossága
- Amelyik mikrovezérlő nem rendelkezik a fejlett órajel-újraszinkronizáló képességgel, azoknál a HSE (külső kristály rezonátorral stabilizált frekvenciájú) nagyfrekvenciás oszcillátor használata kötelező
- Az **STM32F103C8** (Blue pill) kártyánál a külső 8 MHz-es kvarc biztosítja a kellően pontos órajelet
- A **Nucleo-F446RE** kártyánál az átkötések gyári beállítását használva a kártyára épített **ST-Link-V1** szolgáltat „kvarc pontosságú” 8 MHz-es külső órajelet a HSE számára, az USB használatához ezt kell konfigurálni (HSE bypass mód)



USB busz állapotok

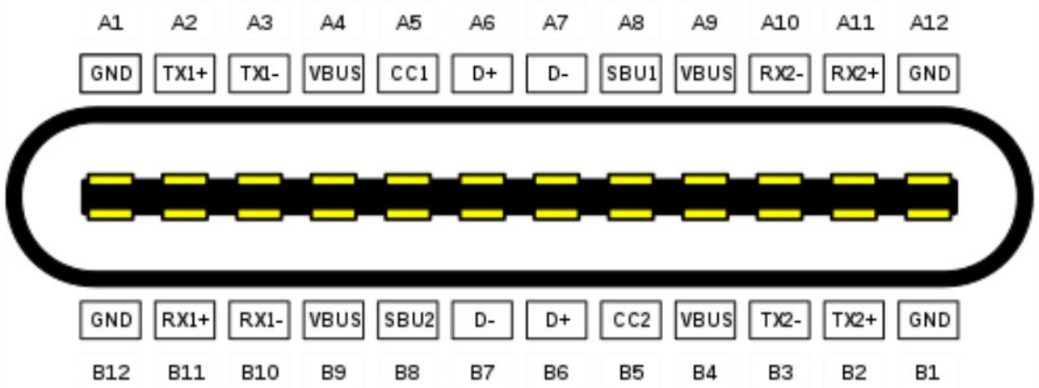
- Az alábbi táblázatban a jelszintek és a hozzájuk tartozó busz állapotok láthatók
- A táblázatban a **J** állapotnak nevezett a csatlakoztatott eszköz **Idle** (tétlen) állapota

Bus State	Condition
Differential 1	D+ high, D- low
Differential 0	D- high, D+ low
Single Ended Zero (SE0)	D+ and D- low
Single Ended One (SE1)	D+ and D- high
Data J	Low Speed: Differential 0 Full Speed: Differential 1
Data K	Low Speed: Differential 1 Full Speed: Differential 0

Bus State	Condition
Start of Packet (SOP)	Switching from Idle to K state
End of Packet	SE0 for 2 bit times followed by J state for 1 bit time
Disconnect	SE0 for more than 2 us
Connect	Idle for more than 2.5 us
Reset	SE0 for more than 2.5 us acceptable (required >= 10 us)

USB-C és a teljesítmény-átvitel

- **USB-C: új típusú csatlakozó, mind a host mint a végponti eszközök számára**
  - Megfordítva is csatlakoztatható (180° elforgatás)
  - Más illesztőket is támogat (HDMI, Display port, Ethernet, audio, azonosítás)
  - Speciális egyeztetést és hibadetektálást igényel (pl. host-host összekapcsolás)
  - **USB-C ≠ USB 3.x**
- **Lehetővé teszi komolyabb teljesítmény átvitelét**
  - Nagyobb feszültség és áram – jó kábellel
  - Az átvitel iránya nem függ host / device szereptől
  - Külön csatornaként működhet (adatátviteltől függetlenül)
  - Kommunikáció a CC vonalakon
- **Példa: Power bank laptophoz kötve USB-C-n keresztül**
  - Amikor a laptop nem csatlakozik hálózati töltőre, akkor a power bankból táplálható
  - Amikor a laptop hálózati töltőre csatlakozik, akkor tölti a power bankot

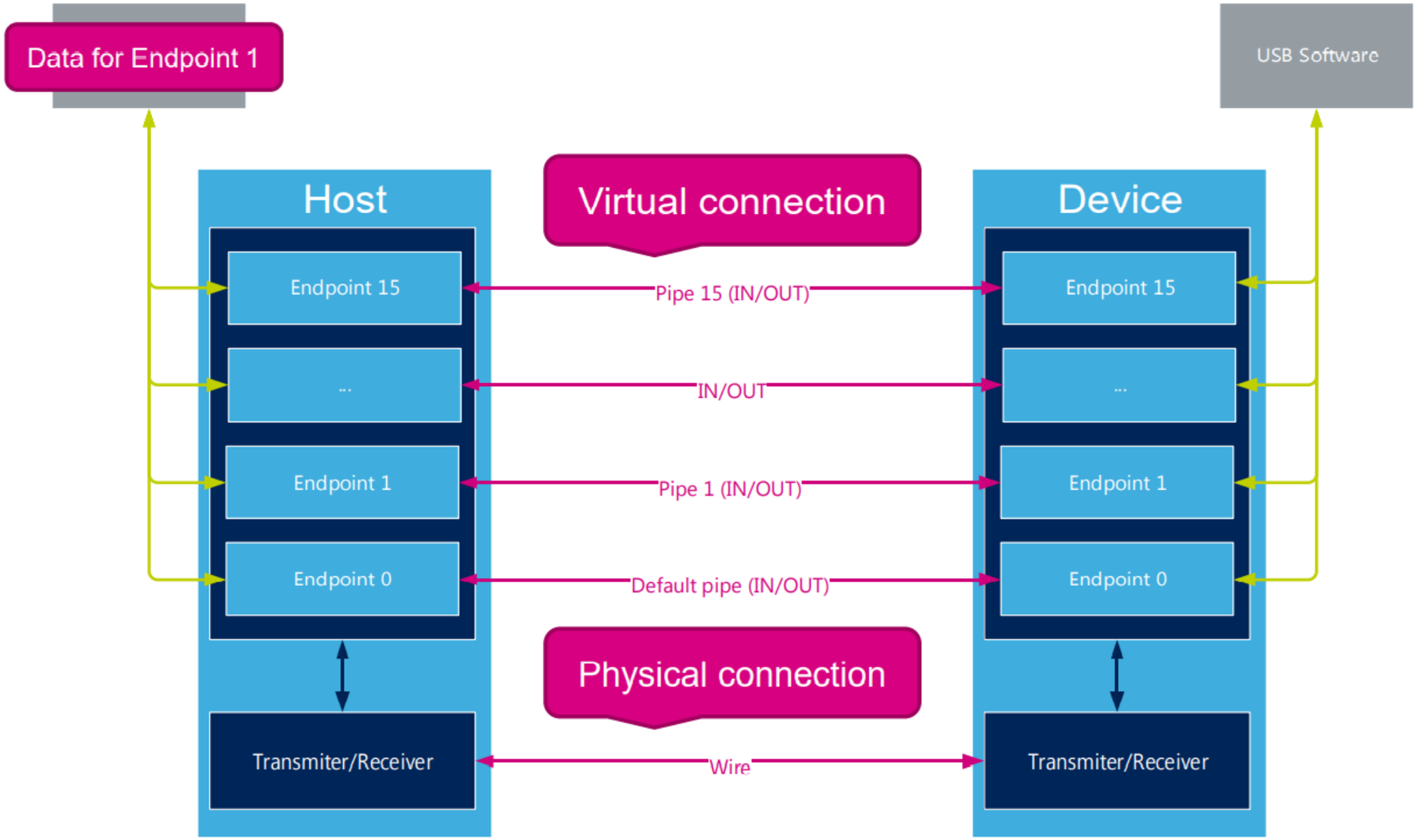


USB On-the-Go (OTG)

- A **2001. végén** kidolgozott **USB on-the-go** specifikáció lehetővé teszi, hogy olyan eszközök, mint pl. táblagépek, okostelefonok hostként viselkedjenek, s egeret, billentyűzetet, USB sticket, vagy digitális kamerát csatlakoztassunk hozzájuk
- A host, vagy device funkciót a kábelen keresztül az **ID** jel választja ki
- **Micro AB** csatlakozó: A és B típusú csatlakozót is fogad
- Példa: mobiltelefon
  - Számítógéphez csatlakoztatva eszközként kapcsolódik (B)
  - Eszközt (pl pendrive) „OTG” kábelen csatlakoztatva (A) hostként viselkedik
- Az OTG funciót a HAL programkönyvtár nem támogatja, habár az **STM32F446RE** mikrovezérlő hardvere képes lenne rá

Csővezetékek és végpontok

- Fizikailag csak egy összeköttetés van, virtuálisan pedig max. 16 db



Csővezetékek és végpontok

- Minden eszköz több végponttal (endpoint) rendelkezik
- **A végpont** virtuális csatorna host és az eszköz között
- Minden végpont különböző átviteli tranzakciókat végezhet, amelyk az alábbiak lehetnek:
  - **SETUP** (beállítás)
  - **BULK** (tömeges átvitel)
  - **Interrupt** (megszakításos átvitel)
  - **Isochronous** (valósidejű átvitel)
- **Endpoint 0** fenntartott az USB eszköz számbavételére (enumeration) és konfigurálására
- Legfeljebb 16 végpont használható (mindkét irányba), a hardver kiépítettségétől függően
- **Az USB csővonal** (pipe) kapcsolat a host és egy kiválasztott eszköz kiválasztott végpontja között
  - Egy eszközcímből és egy végpont sorszám határozza meg
  - Host nézőpontból használjuk

USB standard osztályok áttekintése

- Az eszközosztályokat az USB-IF definiálja
- Az eszközosztályokat az operációs rendszerek (többnyire) támogatják
  - Nem kell hozzájuk külön meghajtóprogramot írni
- Kompatibilitást biztosítanak a különböző host-ok és eszközök között
- Különböző alkalmazások széles területét lefedik (nem soroltunk fel minden osztályt)
  - **MSC** _ Mass Storage Class (nyers adatokat kezelő tömegtároló osztály), pl. USB stick
  - **MTP** – Multimedia Transfer Protocol - saját fájlrendszer kezeléssel rendelkező eszköz (médialejátszó, fényképezőgép, okostelefon stb)
  - **CDC** – Communication Device Class (virtuális soros port, általában bulk transfer)
  - **HID** – Human Interface Device (pl. egér, billentyűzet, játékvezérlő – interrupt transfer)
  - **DFU** – Device firmware update (pl. STM32F4xx USB bootloader)
  - **Audio** class – valós idejű hangátvitel (mikrofon, hangszóró – isochronous transfer)
- **Kompozit eszközök:** többféle eszközosztályt megvalósító eszköz (például az **ST-Link V1** egyidejűleg programozó/hibavadász + tömegtároló + virtuális soros port

Támogatott osztályok az STM32 USB programkönyvtárban

USB Class	Device support	Host support	Windows driver support
MSC	Yes	Yes	Yes
MTP	No	Yes	Yes
CDC	Yes	Yes	From Windows 10 / with ST drivers
HID	Yes	Yes	Yes
AUDIO	Yes	Yes	Yes
DFU	Yes	No	With ST drivers

STM32 MCU USB perifériák áttekintése

	Device family	Core	Host/OTG	HS interface	Crystal-less
General purpose	STM32F0x0 (Value line)	Cortex-M0	No	No	No
	STM32F0x2/0x8	Cortex-M0	No	No	Yes
	STM32F102/103	Cortex-M3	No	No	No
	STM32F105/107	Cortex-M3	Yes	No	No
	STM32F3	Cortex-M4	No	No	No
High performance	STM32F2	Cortex-M3	Yes (2x)	ULPI (1x)	No
	STM32F4/F7	Cortex-M4	Yes (2x)	ULPI (1x)	No
	STM32F7x3	Cortex-M7	Yes (2x)	**On-chip PHY**	No
	STM32H7	Cortex-M7	Yes (2x)	ULPI (1x)	Yes
Low power	STM32L0	Cortex-M0+	No	No	Yes
	STM32L1	Cortex-M3	No	No	No
	STM32L4x2/4x3	Cortex-M4	No	No	Yes
	STM32L4x5/4x6	Cortex-M4	Yes	No	From LSE
	STM32L4+	Cortex-M4	Yes	No	Yes

life.augmented

USB device-only perifériák

- Csak a Full Speed mód támogatott. Max. 8 végpont, a kétirányú végpont minkét irányban azonos típusú (pl. bulk) legyen! Sok típusnál a CAN perifériával együtt nem használható

Device	Buffer (bytes)	Buffer (with CAN enabled)	LPM	BCD
STM32F0	1024	768	Yes	Yes
STM32F103/105	512	0 (forbidden)	No	No
STM32F303xB/C STM32F358xC STM32F302xB/C	512	512	No	No
STM32F303xD/E STM32F302x6/8	1024	768	Yes	No
STM32L0	1024	1024 (no CAN)	Yes	Yes
STM32L1	512	512 (no CAN)	No	No
STM32L4x2/4x3	1024	1024 (no CAN)	Yes	Yes

STM32 USB OTG perifériák

- High-speed támogatott átvitel néhány eszköznél, de külső PHY IC kell hozzá
  - USB_OTG_HS full-speed módban is működhet, a belső PHY használatával

Device	USB_OTG_FS			USB_OTG_HS		
	Endpoints *	Pipes (Host)	FIFO size	Endpoints *	Pipes (Host)	FIFO size
STM32F1 STM32F401 STM32F411	4 IN + 4 OUT	8	1.25 K	-		-
STM32F2 STM32F4 (other)	4 IN + 4 OUT	8	1.25 K	6 IN + 6 OUT	12	4 K
STM32F412 STM32F413	6 IN + 6 OUT	12	1.25 K	-		-
STM32F446/469/479 STM32F7	6 IN + 6 OUT	12	1.25 K	8 IN + 8 OUT	18	4 K

Virtuális soros port

- Kommunikációs protokoll, ami COM portot emulál
- Elterjedt kommunikációs módszer a PC és a beágyazott rendszerek között
- Széleskörűen támogatott: sok terminál emulátor szoftver létezik, szoftveresen könnyen kezelhető
- Hátrányok:
  - További végpont, további réteg (COM port enumeráció)
  - Nincs natív Plug and Play támogatás

Példaprogramok

Blue pill kártya:

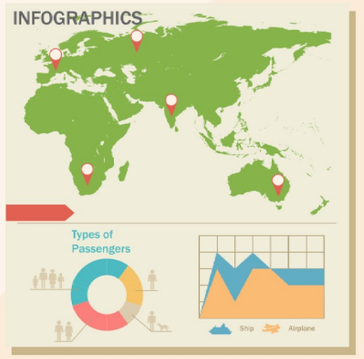
F103_USB_proba – egyszerű kiíratás USB CDC kapcsolaton

F103_USB_CDC – kétirányú USB CDC kommunikáció

Nucleo-F446RE kártya:

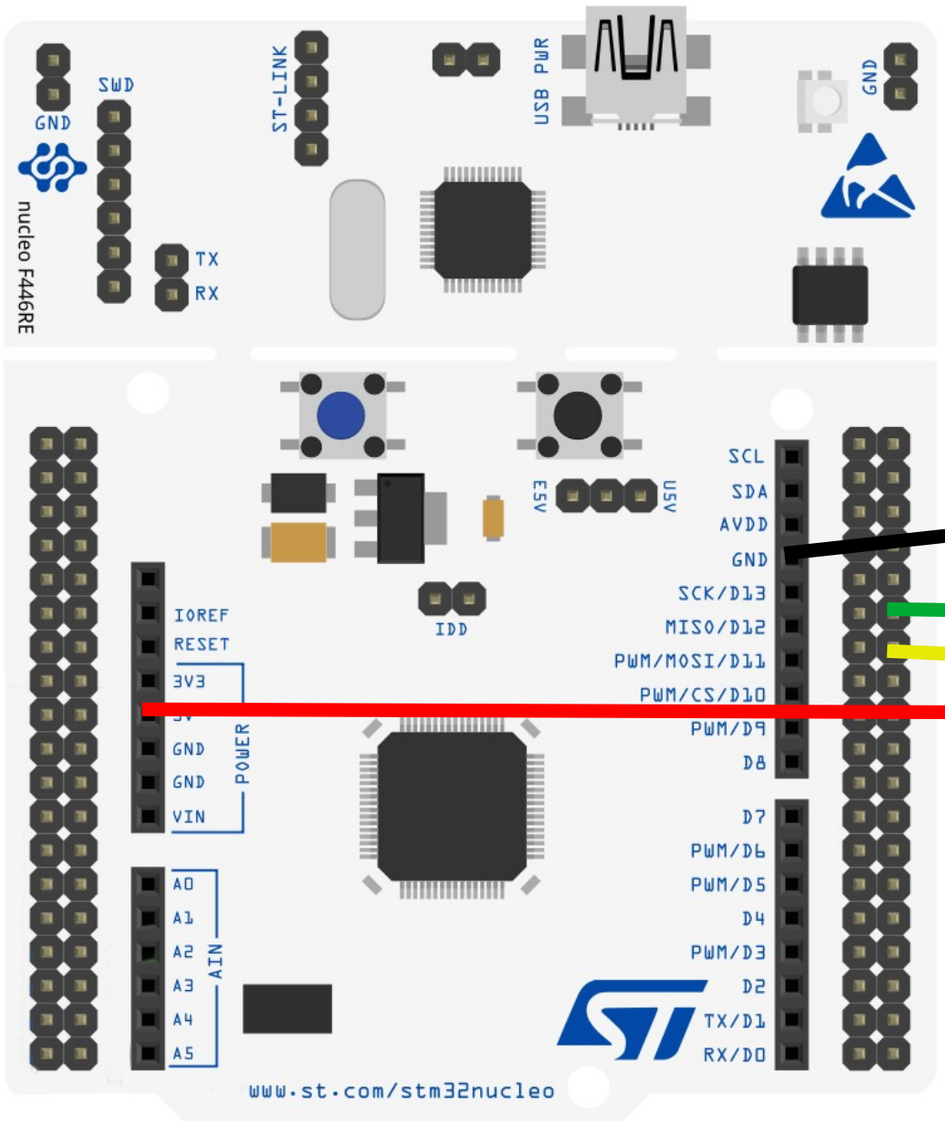
F446RE_USB_proba – egyszerű kiíratás USB CDC kapcsolaton

F446RE_USB_CDC – kétirányú USB CDC kommunikáció

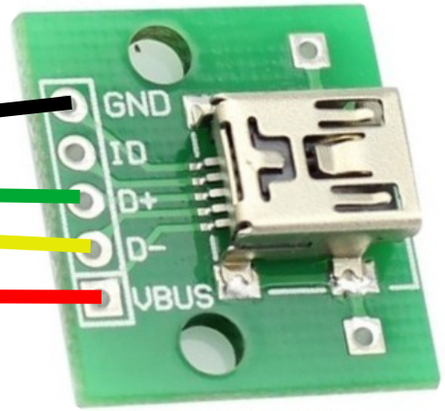


freepik.com

Bekötési vázlat

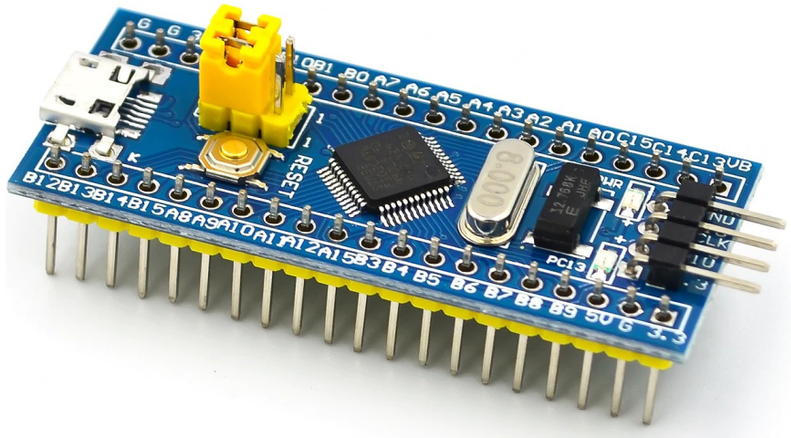


PA12 USB D+
PA11 USB D-



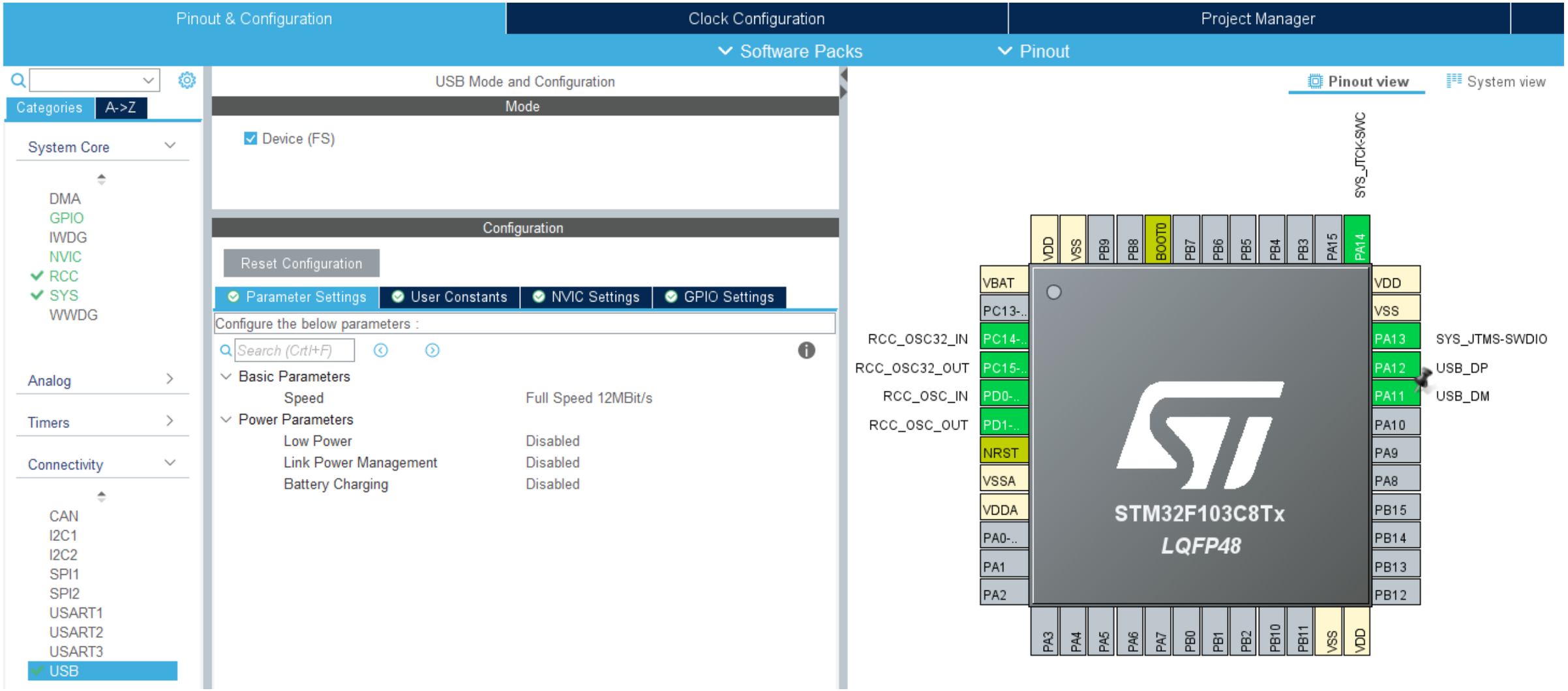
STM32F4xx: a **D+** vonal külső felhúzására nincs szükség!

STM32F103C8: a **D+** vonalat 1,5 kΩ-mal 3.3 V-ra fel kell húzni!



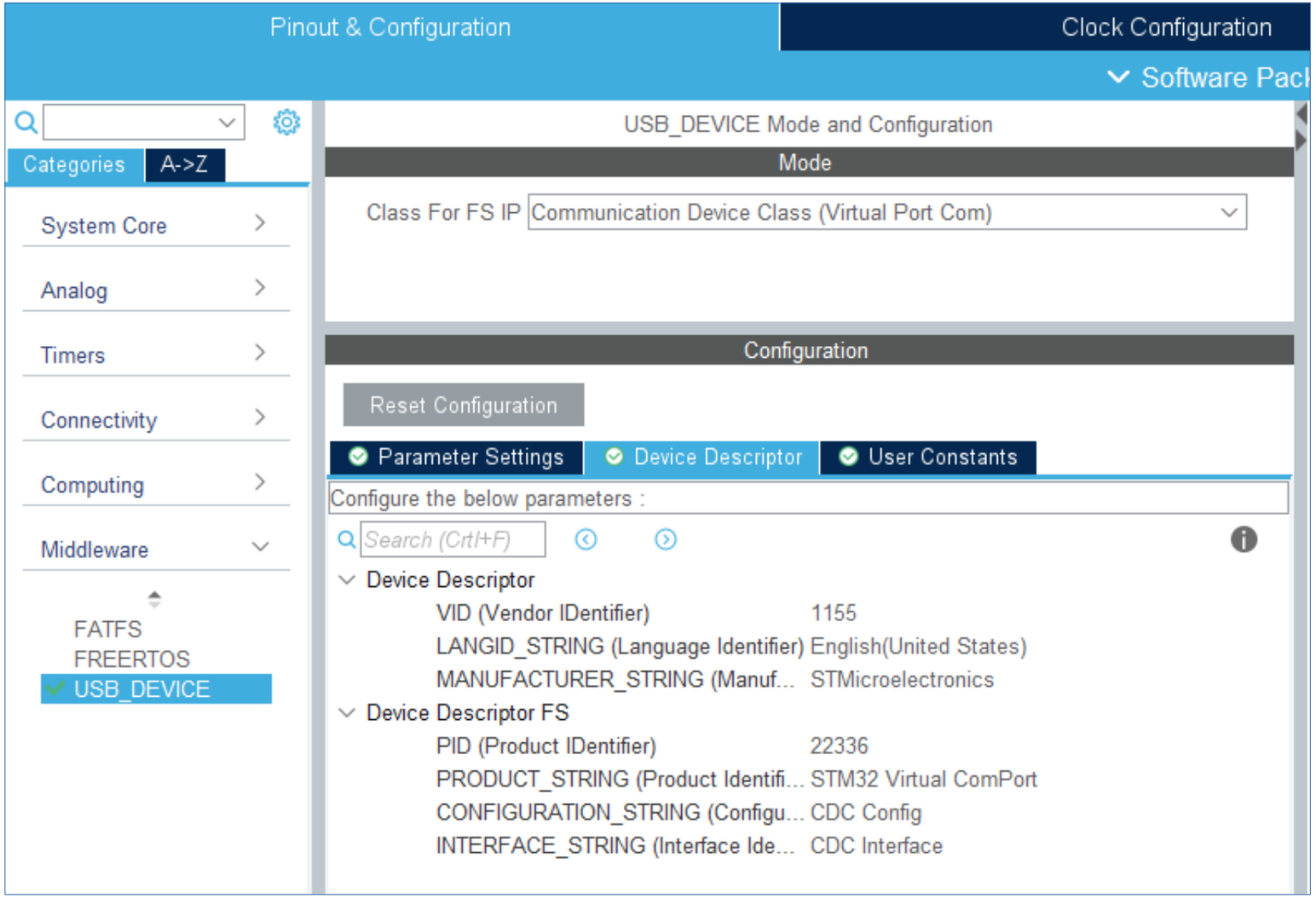
F103_USB_proba: az USB periféria konfigurálás

- Az USB kivezetések **PA12** (D+) és **PA11** (D-), **PA12** 1.5 kΩ-mal 3.3 V-ra legyen felhúzva!



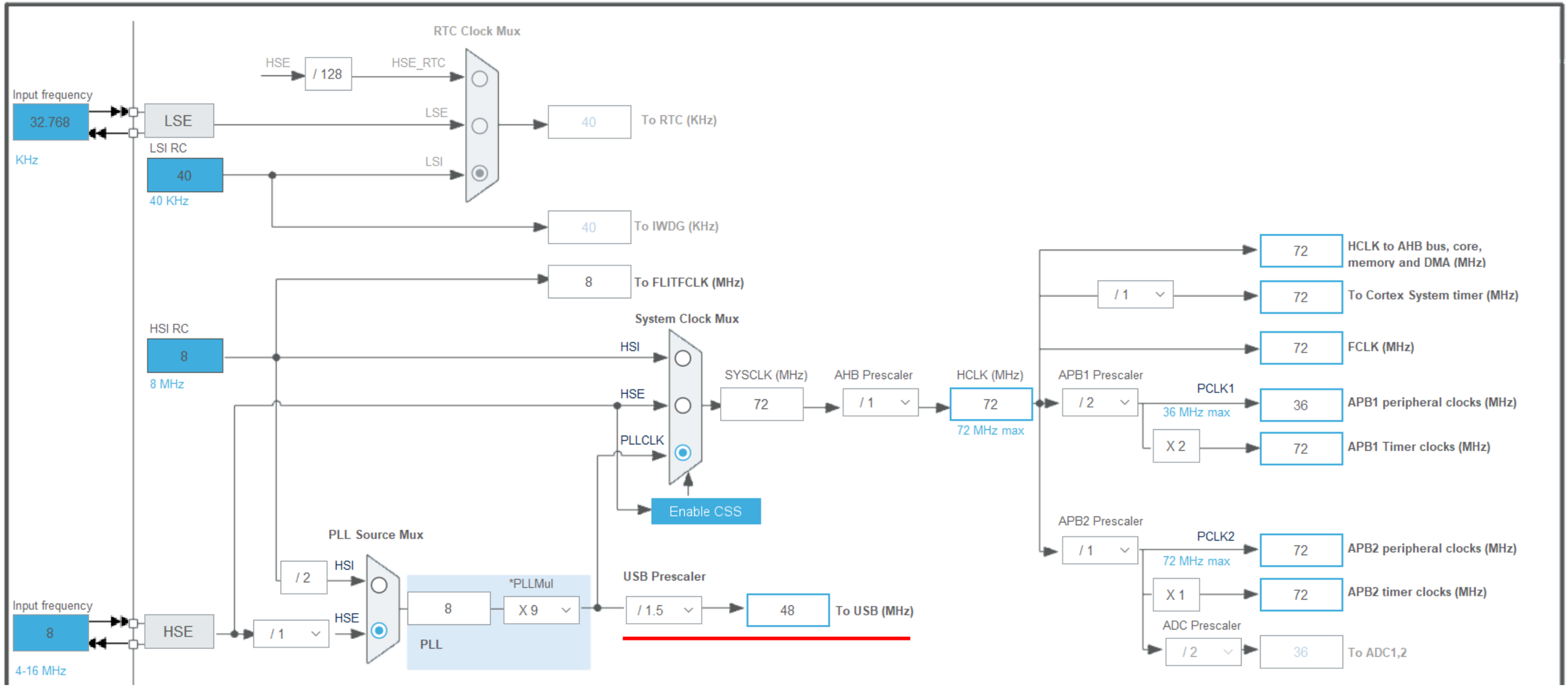
F103_USB_proba: az USB programkönyvtár konfigurálása

- A **Middleware** szekcióból az **USB_DEVICE** csomagot válasszuk ki
- A **Virtuális Port COM** osztály kell nekünk (CDC class)
- A képen látható VID/PID párost ismeri fel és támogatja a gyártó által biztosított szoftver ”illesztő” (csak .inf állomány)



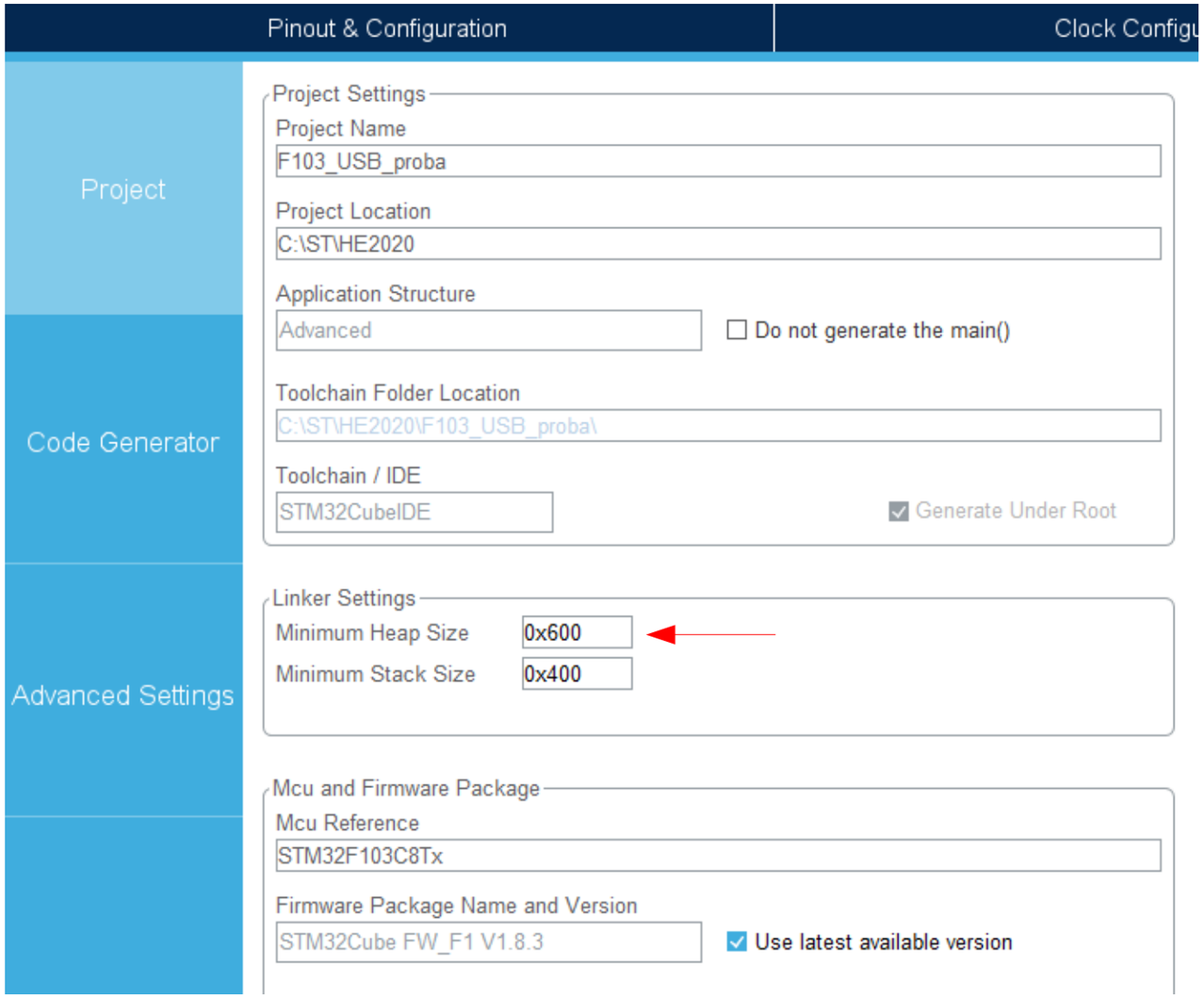
F103_USB_proba: az órajelek konfigurálása

- Az USB Full Speed eszközöknél az USB perifériának **48 MHz**-es órajelet kell biztosítani

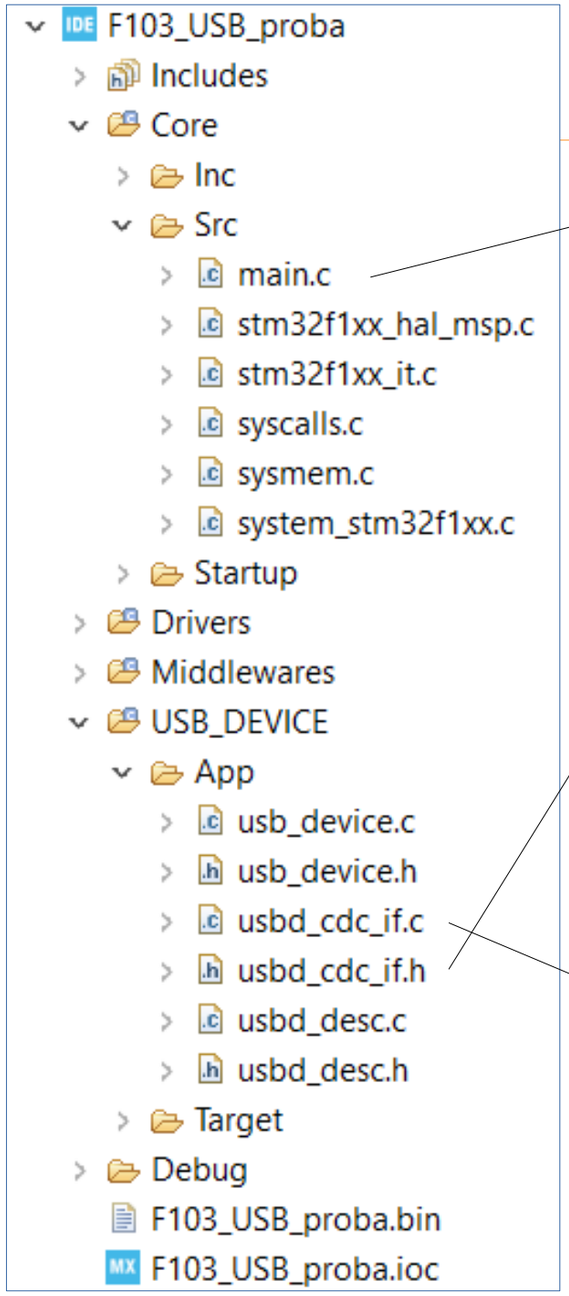


F103_USB_proba: a heap átméretezése

- Az USB működéséhez több helyet kell biztosítani a dinamikus helyfoglaláshoz
- Növeljük meg a **Heap** minimális méretét az eredeti 0x200-ról **0x600**-ra!



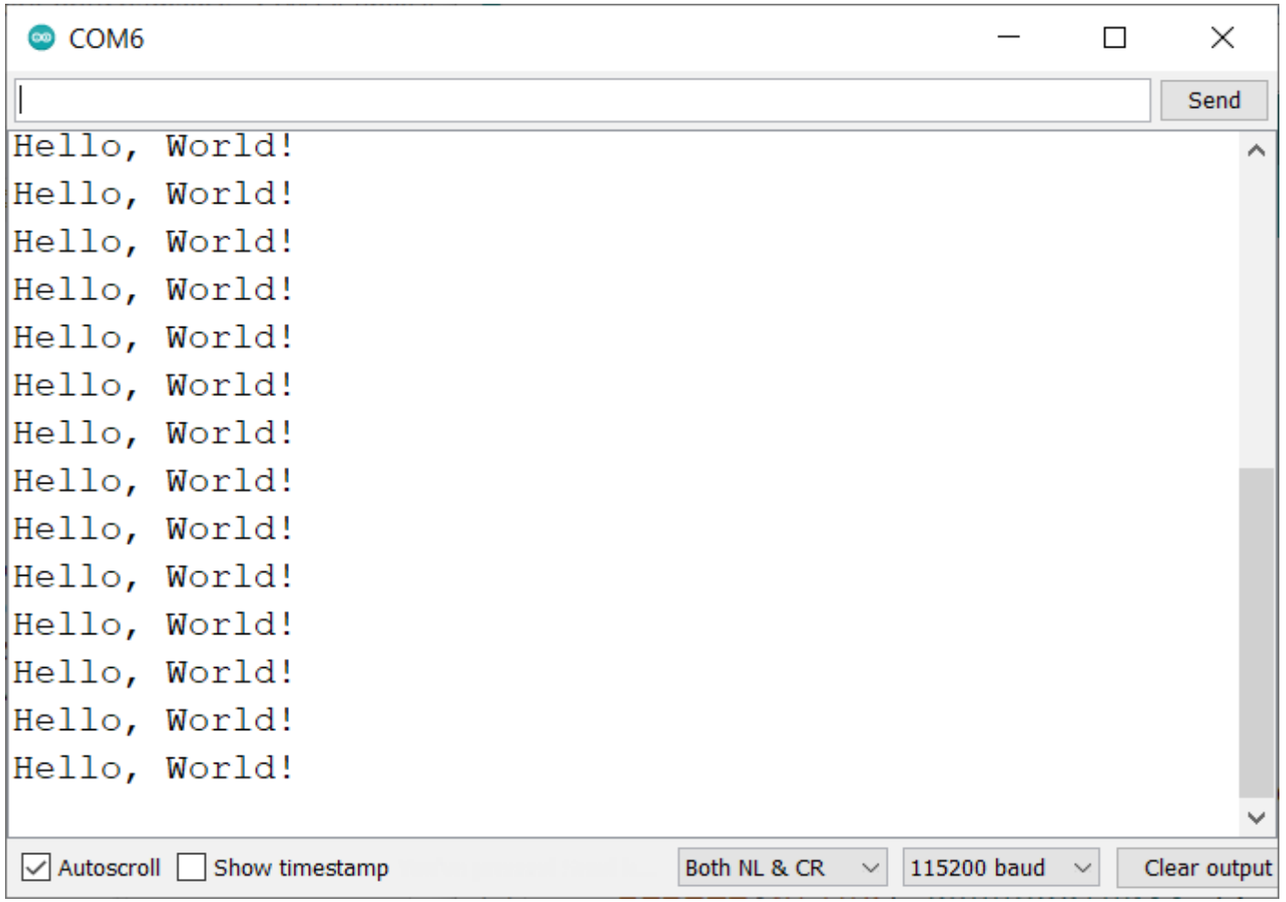
F103_USB_proba: main.c (részlet)




```c
#include "main.h"
#include "usb_device.h"
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"
/* USER CODE END Includes */

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

/* USER CODE BEGIN PV */
uint8_t buffer[] = "Hello, World!\r\n";
/* USER CODE END PV */

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USB_DEVICE_Init();
  while (1) {
    CDC_Transmit_FS(buffer, sizeof(buffer));
    HAL_Delay(1000);
  }
}
```

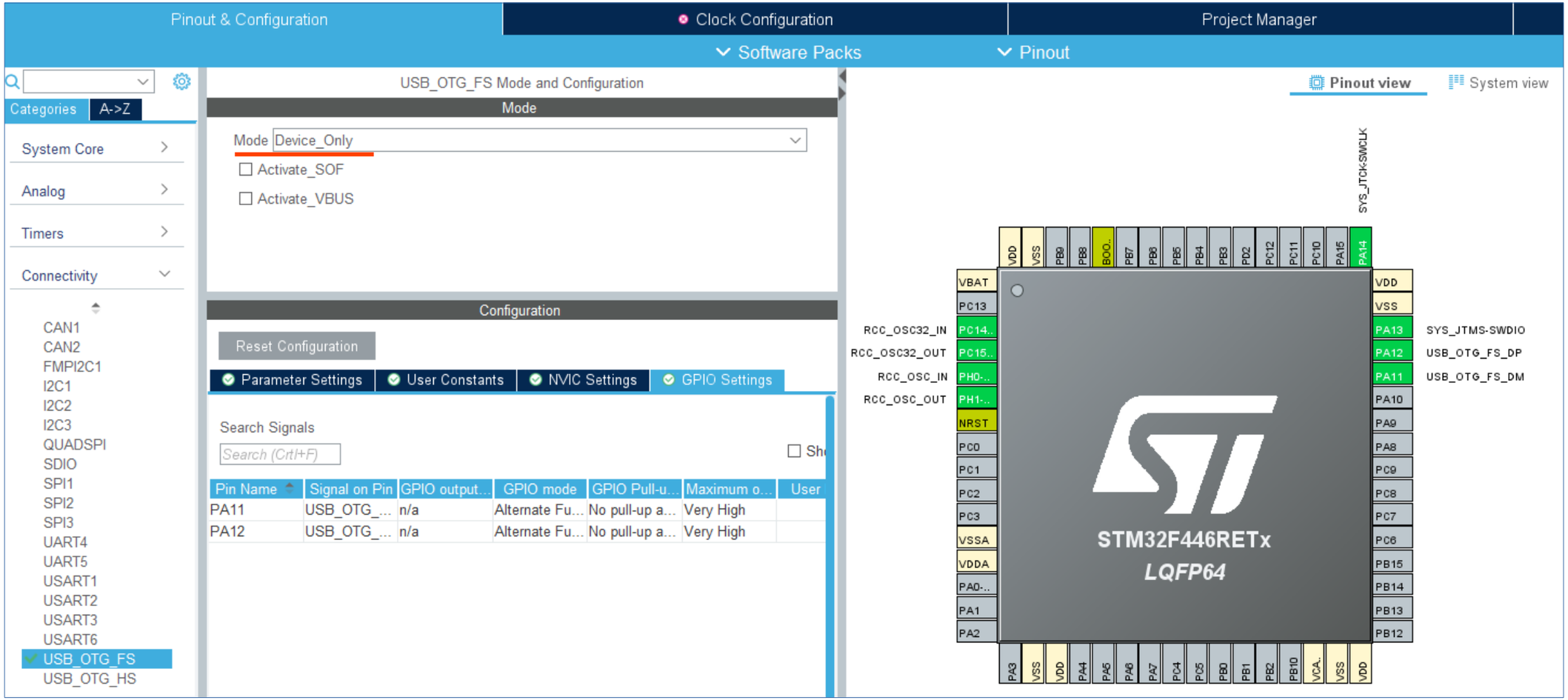
F103_USB_proba: futási eredmény

- A futási eredményt egy terminálablakban (ez lehet az **Arduino IDE** terminál ablaka, a **PuTTY.exe** program, vagy bármi más) tekinthetjük meg
- Csak a virtuális port sorszámát (nálam most COM6) kell beállítani a baudrate nem számít!

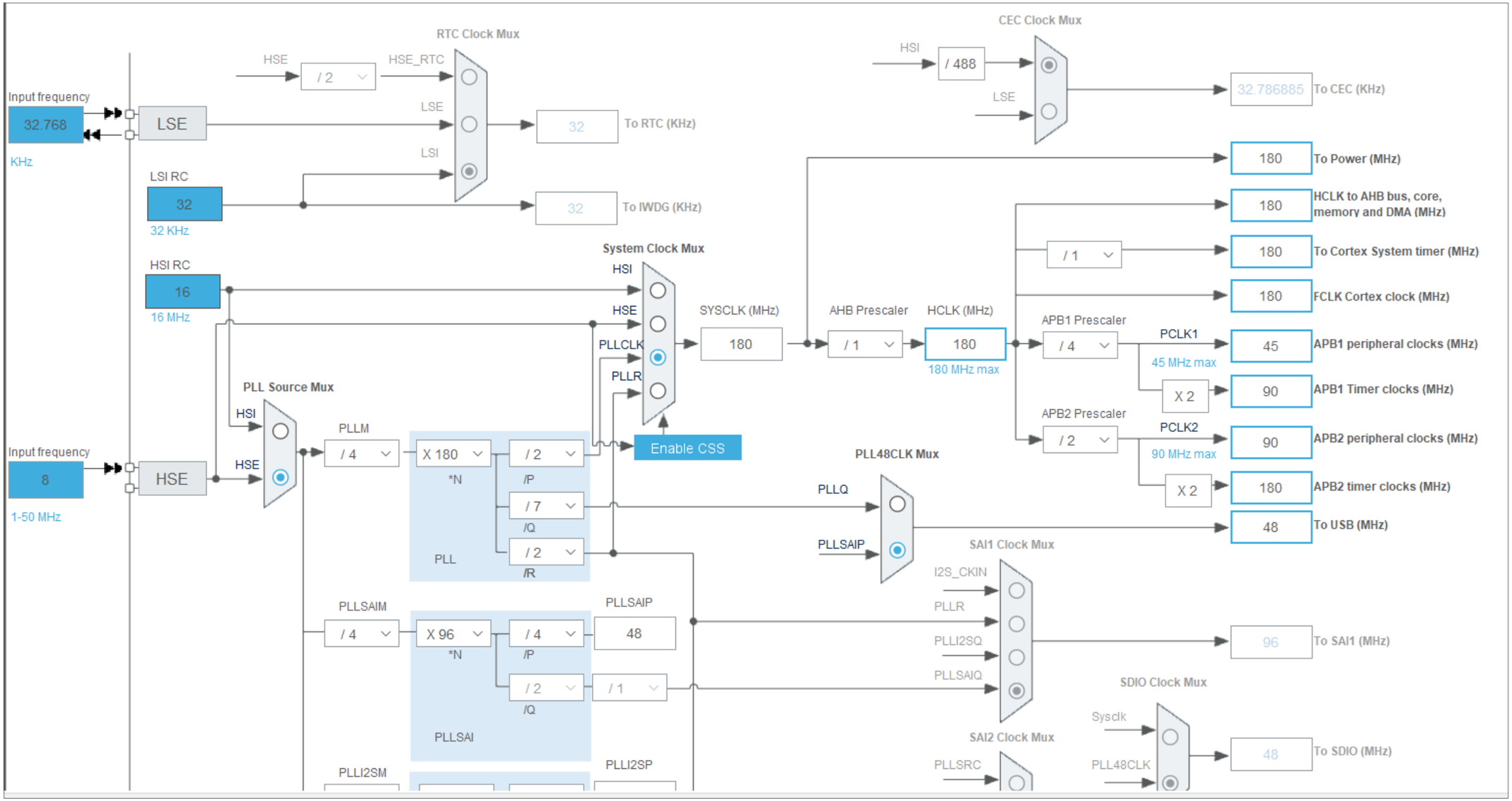


F446RE_USB_proba: USB FS konfigurálás

- Az **STM32F446RE** MCU két USB perifériával rendelkezik, most az **USB FS**-t használjuk, méghozzá eszköz (device) módban. A kivezetések itt is **PA12** (D+) és **PA11** (D-), de nem kell külső felhúzás



F448RE_USB_proba: órajelek konfigurálása



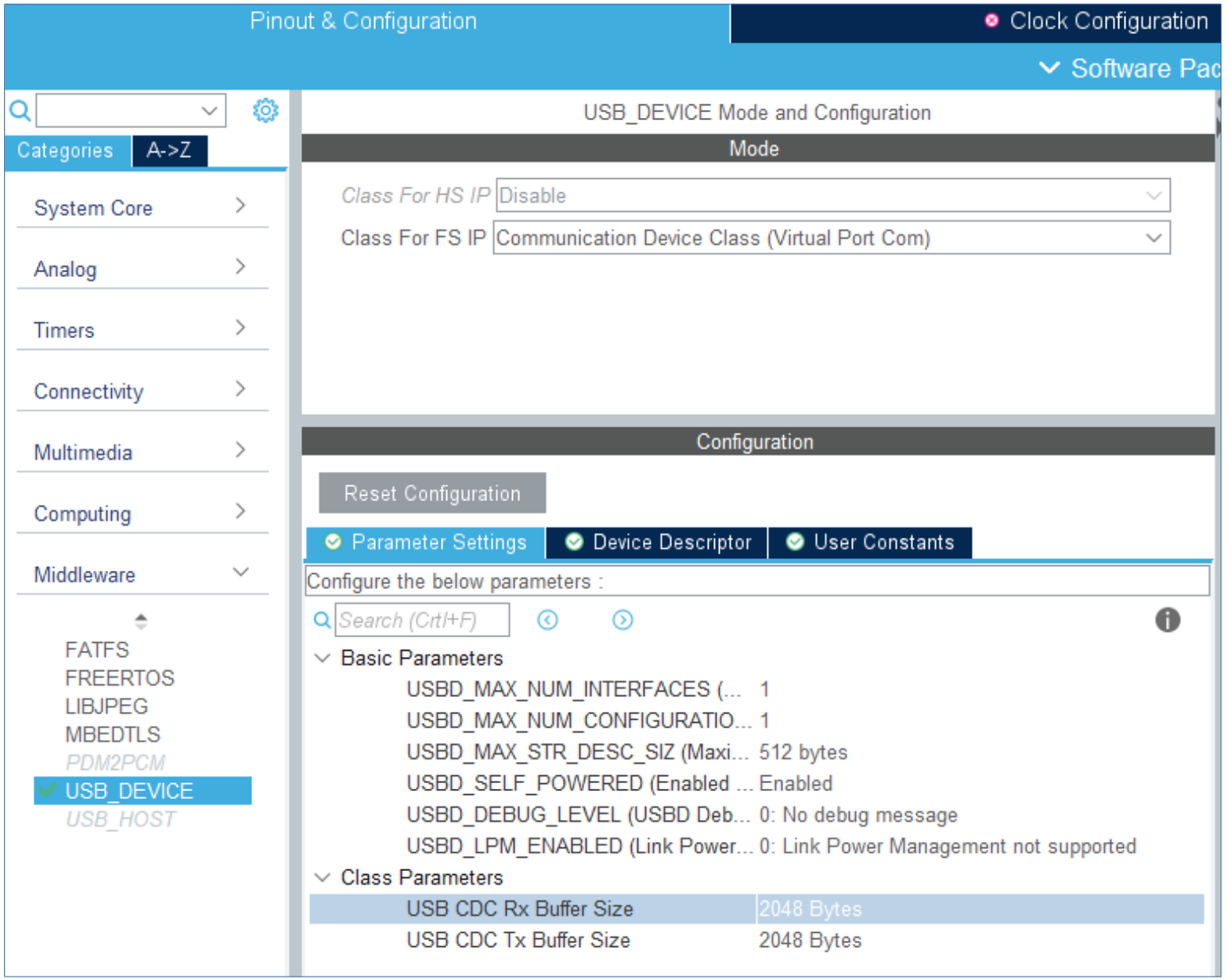
Maximális HCLK eléréshez két lehetőség nyílik:

1. HCLK = 168 MHz és PLLQ = 48 MHz (336 Mhz/7)

2: HCLK = 180 MHz, PLLSAIP = 48 MHz (2 MHzx96/4)

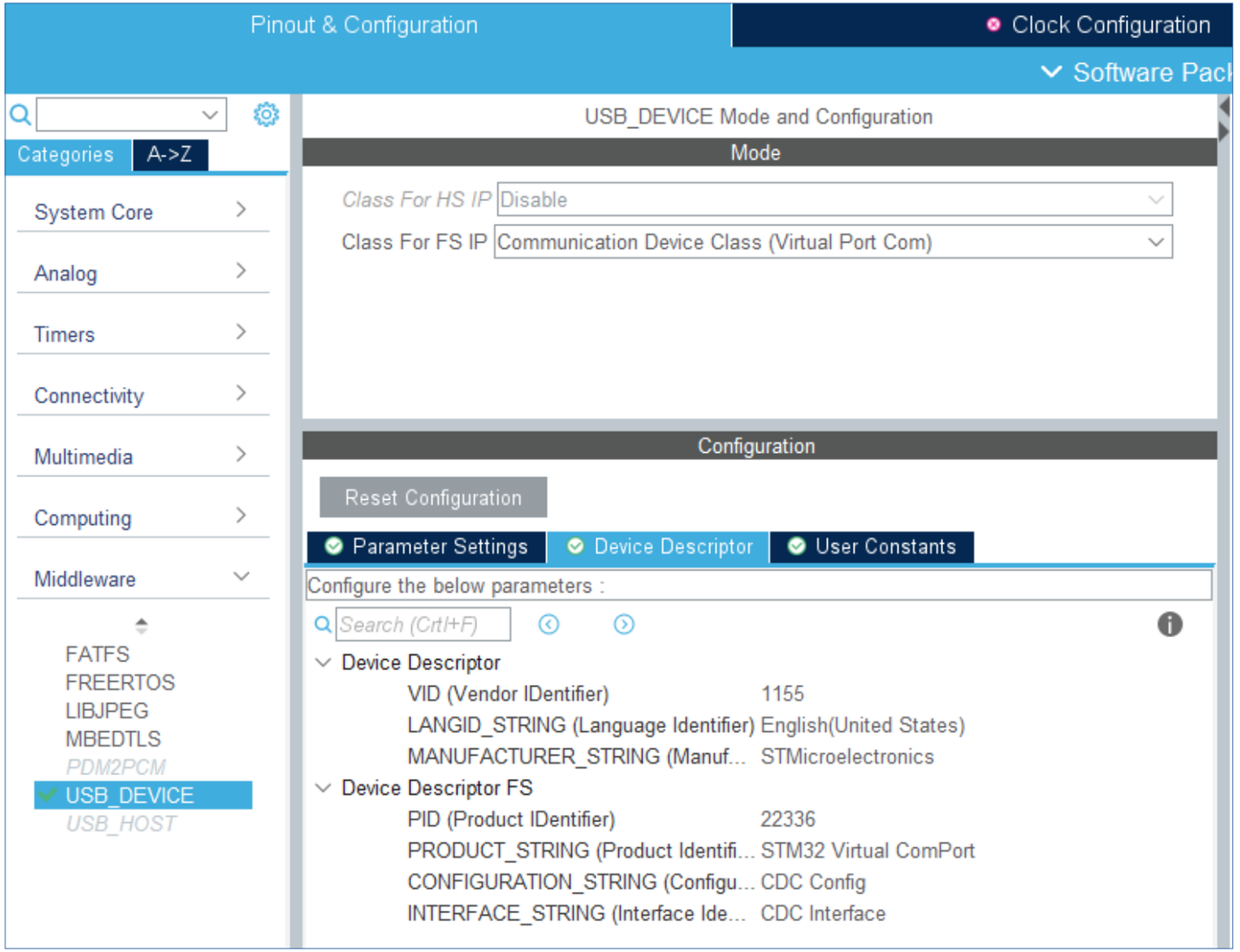
F446RE_USB_proba: USB_DEVICE konfigurálás

- Az **STM32F446RE** MCU-nál a **Middleware** szekcióban USB_HOST és USB_DEVICE programcsomag is található, melyek közül most az utóbbira lesz szükségünk
- A **Virtual Port Com** osztályt (USB CDC class) válasszuk ki!
- A buffer méretét csökkenthetjük (128 bájt is bőven elég), de a default értéket is meghagyhatjuk (bőven telik rá...)



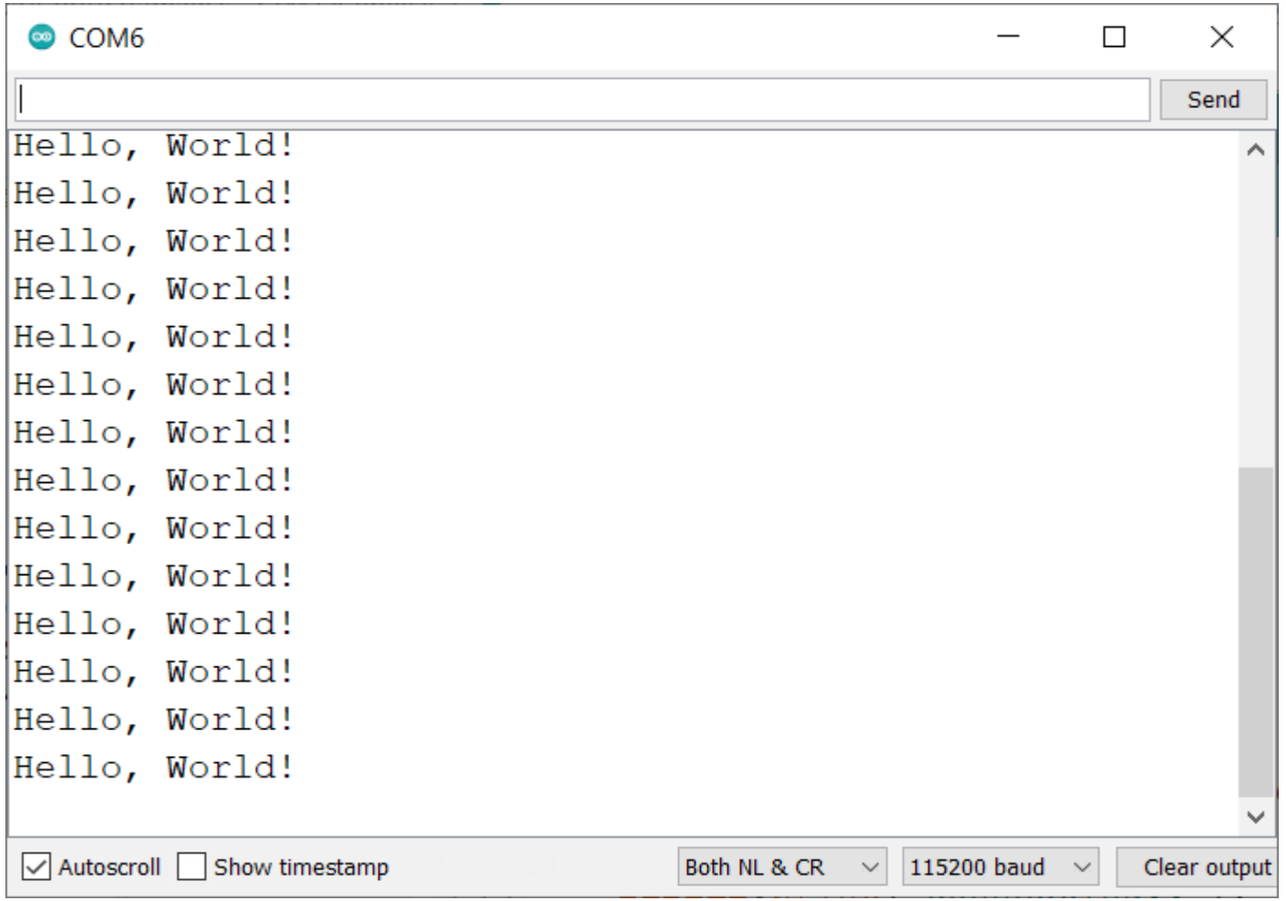
F446RE_USB_proba: USB_DEVICE konfigurálás

- A képen látható **VID/PID** párost ismeri fel és támogatja a gyártó által biztosított szoftver ”illesztő” (csak .inf állomány)



F446RE_USB_proba: futási eredmény

- A **main.c** releváns része és a futási eredmény megegyezik az **F103_USB_proba** projektnél látottakkal!



USB VCP Device

- Hogyan tudunk adatokat küldeni és fogadni a virtuális soros porton?
  Az ezekhez szükséges függvények az automatikusan generált **usbd_cdc_if.c** állományban találhatók
- Ugyanitt található az a visszahívási függvény ami a COM port paramétereinek beállításait kezeli

```
static int8_t CDC_Control_FS  (uint8_t cmd, uint8_t* pbuf, uint16_t length)
```

A fenti függvényre nekünk ugyan nincs szükségünk, de a **Windows** (különösen a Windows 7) szeretné visszaolvasni az általa kiküldött beállításokat ehhez definiálunk egy struktúrát, és kiegészítjük a függvényt

```
/* USER CODE BEGIN PRIVATE_VARIABLES */
USBD_CDC_LineCodingTypeDef LineCoding = {
    115200, /* baud rate */
    0x00,   /* stop bits-1 */
    0x00,   /* parity - none */
    0x08    /* nb. of bits 8 */
  };
/* USER CODE END PRIVATE_VARIABLES */
```

USB VCP Device

- A **CDC_Control_FS()** függvényben a **switch** elágazás `CDC_SET_LINE_CODING` és `CDC_GET_LINE` esetét bővítjük ki, az alábbiak szerint (szükség esetén eltároljuk, illetve, visszaolvassuk a soros porti beállításokat)

```
case CDC_SET_LINE_CODING:
            LineCoding.bitrate    = (uint32_t)(pbuf[0] | (pbuf[1] << 8) |\
                                  (pbuf[2] << 16) | (pbuf[3] << 24));
            LineCoding.format     = pbuf[4];
            LineCoding.paritytype = pbuf[5];
            LineCoding.datatype   = pbuf[6];
    break;
  case CDC_GET_LINE_CODING:
            pbuf[0] = (uint8_t)(LineCoding.bitrate);
            pbuf[1] = (uint8_t)(LineCoding.bitrate >> 8);
            pbuf[2] = (uint8_t)(LineCoding.bitrate >> 16);
            pbuf[3] = (uint8_t)(LineCoding.bitrate >> 24);
            pbuf[4] = LineCoding.format;
            pbuf[5] = LineCoding.paritytype;
            pbuf[6] = LineCoding.datatype;
    break;
```

USB VCP Device

- Adat fogadására a **CDC_Receive_FS()** visszahívási függvényen keresztül van lehetőség, de a generált kód csak félkész: nekünk kell megírnunk azt adatok felhasználását

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  return (USBD_OK);
  /* USER CODE END 6 */
}
```

- **Loopback test:** egyszerű visszatükrözés

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
{
  /* USER CODE BEGIN 6 */
  USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  CDC_Transmit_FS(Buf,*Len);
  return (USBD_OK);
  /* USER CODE END 6 */
}
```

Csak visszaküldjük az adatokat

A bufferek mérete lehetőleg egyezzen meg!

Egy hasznosabb megközelítés: körkörös tár

Egy hasznos mintapélda található a hackaday.io oldalán, ezt építjük be a **usbd_cdc_if.c** állományba

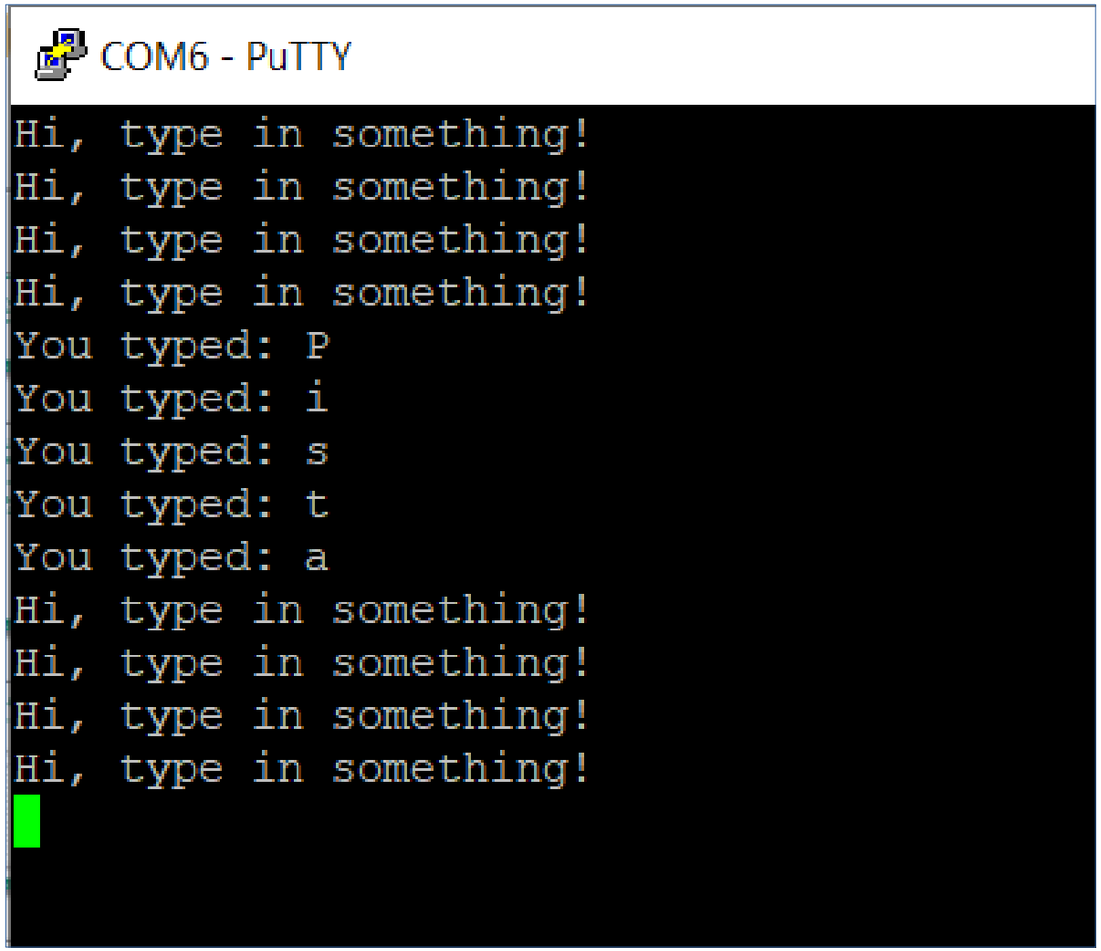
```
#define  FIFO_SIZE 256   // must be 2^N
#define FIFO_INCR(x) (((x)+1)&((FIFO_SIZE)-1))

typedef struct FIFO {    // Structure of FIFO
  uint32_t head;
  uint32_t tail;
  uint8_t data[FIFO_SIZE];
} FIFO;

FIFO RX_FIFO = {.head = 0, .tail = 0};

// Read one character from the FIFO buffer
uint8_t usb_cdc_getc() {
  int ch = 0;
  if (RX_FIFO.head != RX_FIFO.tail) {
    ch = RX_FIFO.data[RX_FIFO.tail];
    RX_FIFO.tail = FIFO_INCR(RX_FIFO.tail);
  }
  return ch;
}
```

```
static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len) {
  uint32_t len = *Len;
  if (hUsbDeviceFS.dev_state != USBD_STATE_CONFIGURED) {
    return USBD_FAIL;
  }
  if (((Buf == NULL) || (Len == NULL)) || (*Len <= 0)) {
    return USBD_FAIL;
  }
  uint8_t result = USBD_OK;
  do {
    result = USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
  } while (result != USBD_OK);
  do {
    result = USBD_CDC_ReceivePacket(&hUsbDeviceFS);
  } while (result != USBD_OK);
  while (len--)       // add data to FIFO
    if (FIFO_INCR(RX_FIFO.head) == RX_FIFO.tail)
      return USBD_FAIL;  // overrun
    else
    {
      RX_FIFO.data[RX_FIFO.head] = *Buf++;
      RX_FIFO.head = FIFO_INCR(RX_FIFO.head);
    }
  return (USBD_OK);
}
```

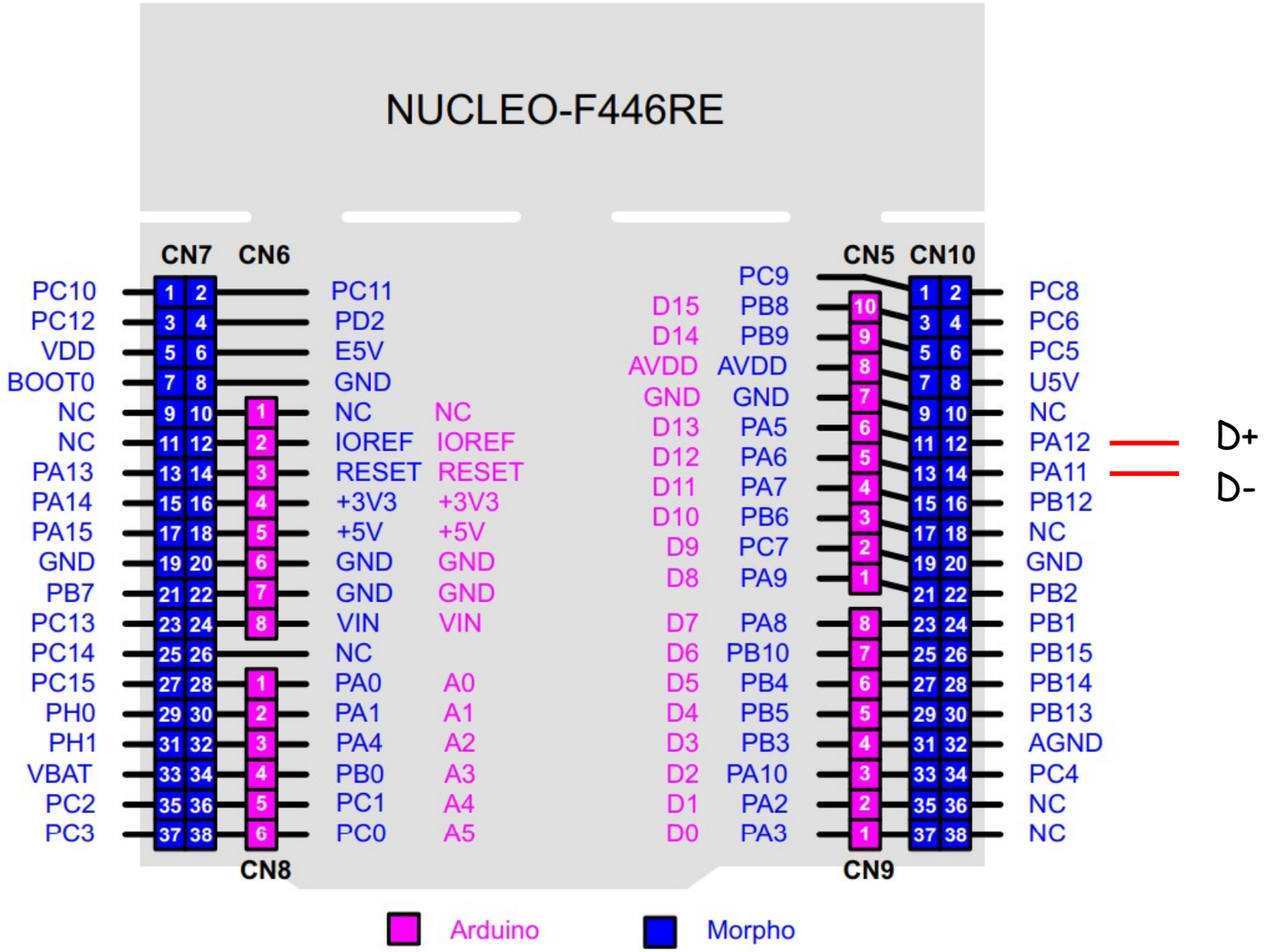
F103_USB_CDC és F446RE_USB_CDC: main.c (részlet)

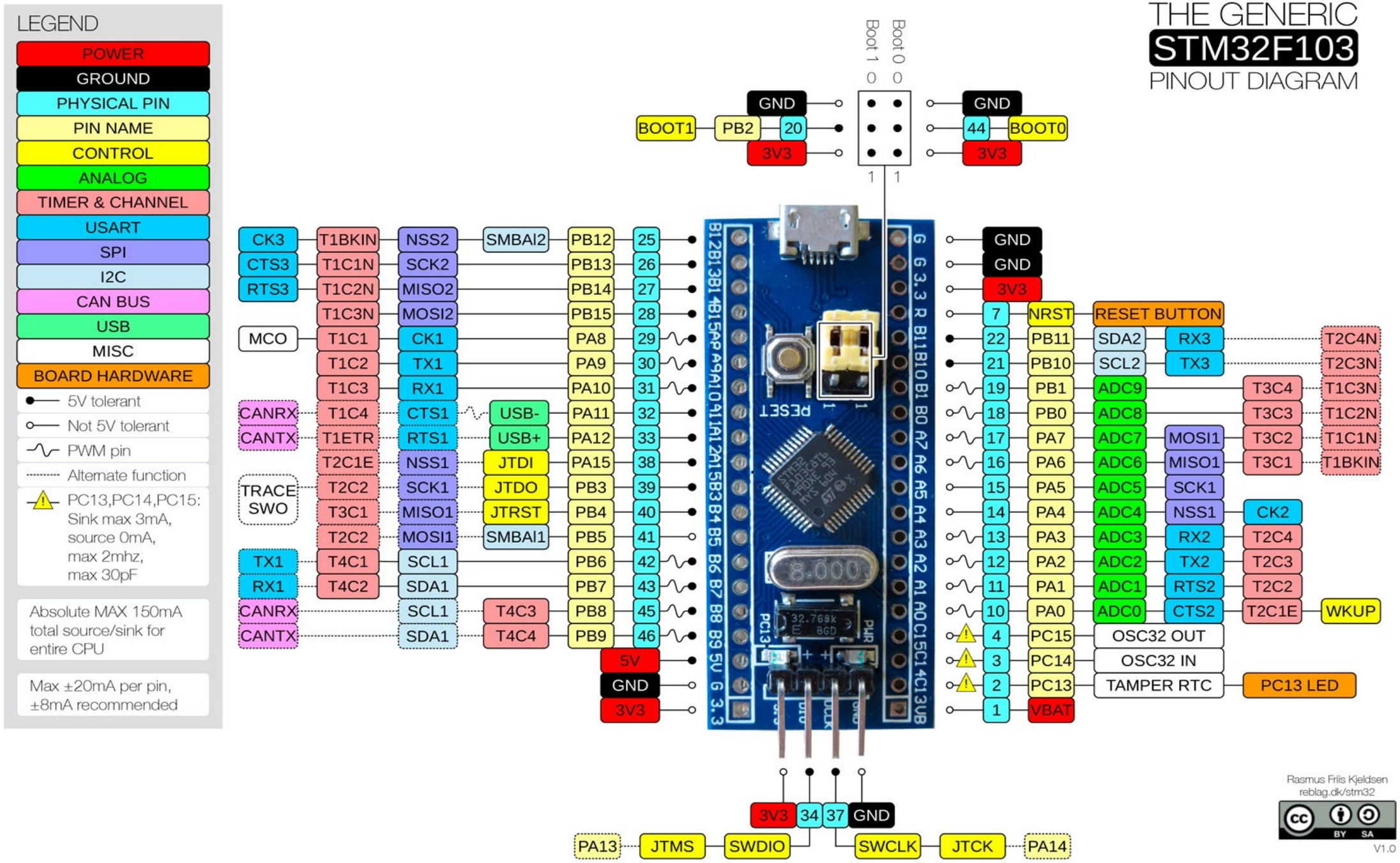
- A vett karaktereket egyenként visszaírjuk

```
#include "main.h"
#include "usb_device.h"
#include "usbd_cdc_if.h"
uint8_t buffer[] = "Hi, type in something!\r\n";
uint8_t data[] = "You typed:  \r\n";
uint8_t ch;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);

int main(void) {
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USB_DEVICE_Init();
  while (1) {
     CDC_Transmit_FS(buffer, sizeof(buffer));
     HAL_Delay(1000);
     ch = usb_cdc_getc();
     if(ch > 32) {
     data[11] = ch;
     CDC_Transmit_FS(data, sizeof(data));
     }
  }
}
```



NUCLEO-F446RE kivezetések




LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14
Rasmus Friis Kjeldsen
reblag.dk/stm32
V1.0