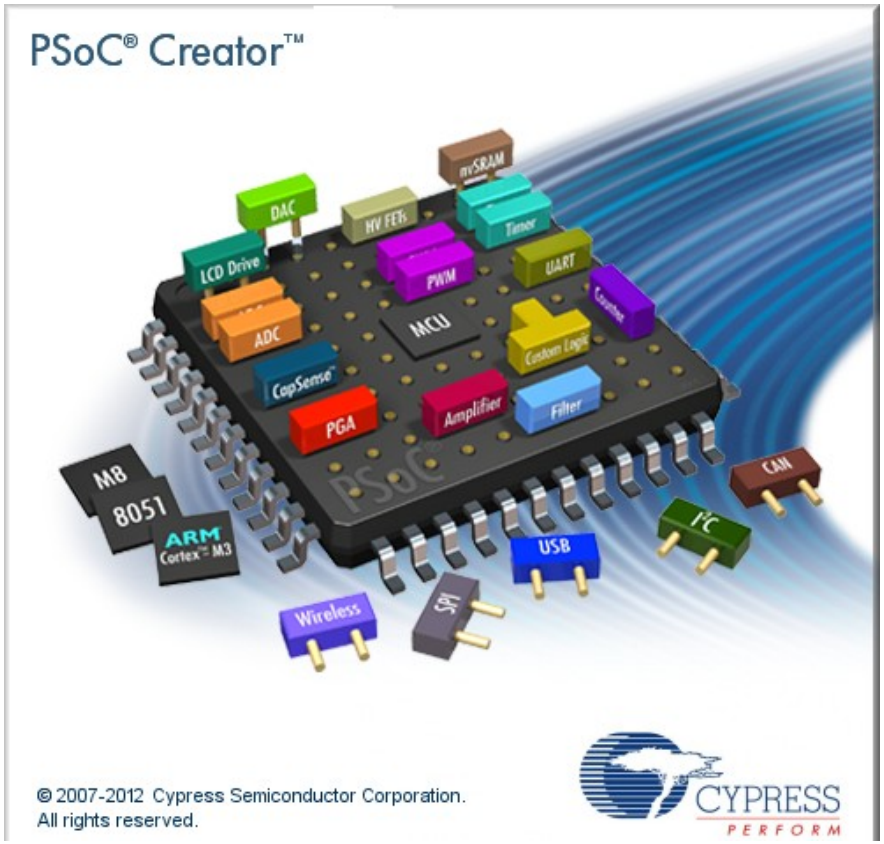
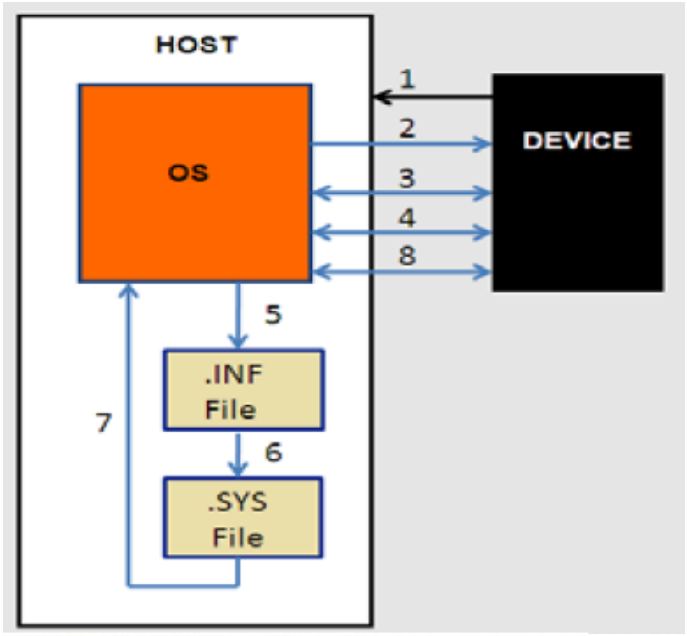
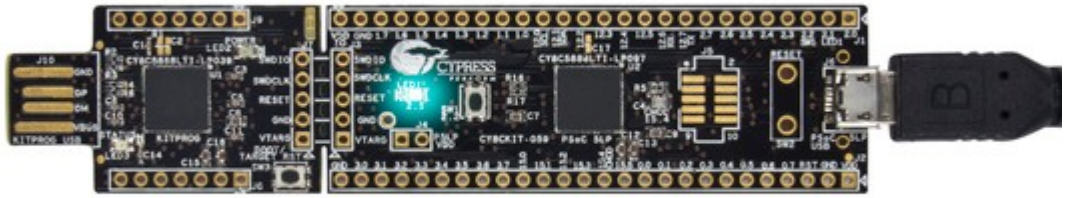
Újrakonfigurálható eszközök







17. Cypress PSOC 5LP – USB kommunikáció

Felhasznált irodalom és segédanyagok

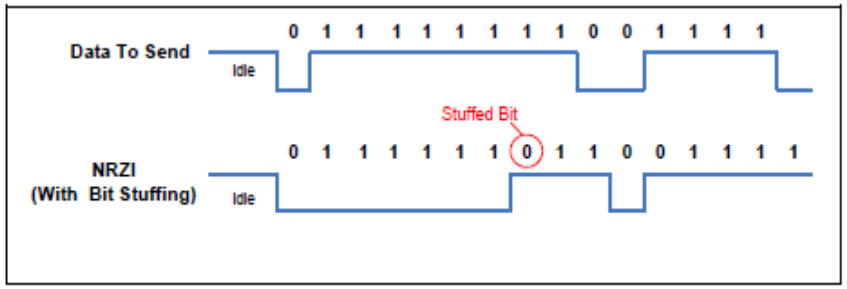
- Cypress: CY8C58LP FamilyDatasheet
- Cypress: PSOC 5LP Architecture Technical Reference Manual)
- Cypress: CY8CKIT-059 Prototyping Kit Guide
- Cypress: AN77 759: Getting Started with PSoC®5LP
- Cypress: PSoC®Creator™ User Guide
- Yuri Magda: Cypress PSoC 5LP Prototyping Kit Measurement Electronics
- Cserny István: PSOC 5LP Mikrokontrollerek programozása
- Cypress: **CE60 246** USBUART in PSoC 3 / PSoC 5
- Cypress: **AN82 072** USB General Data Transfer with Standard OS Drivers

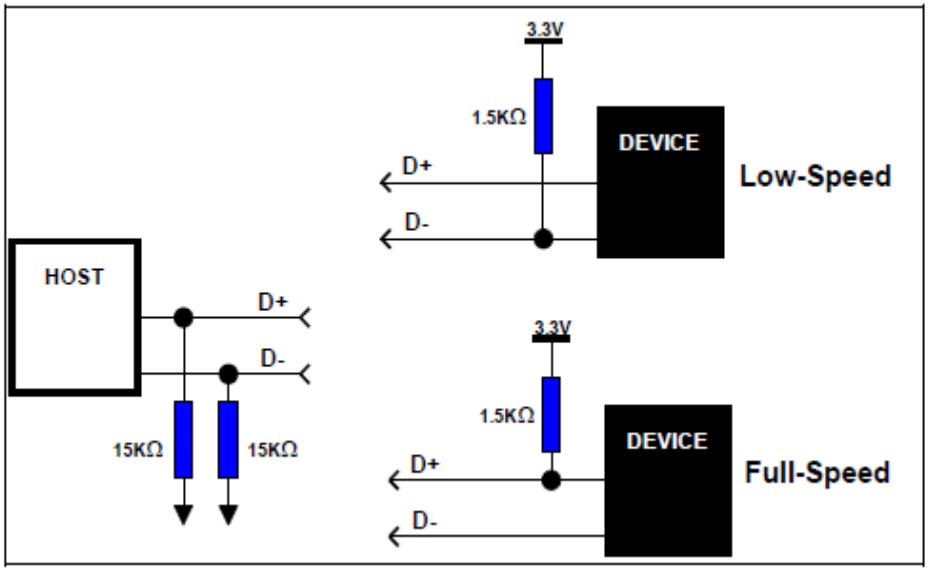
Az USB-ről röviden

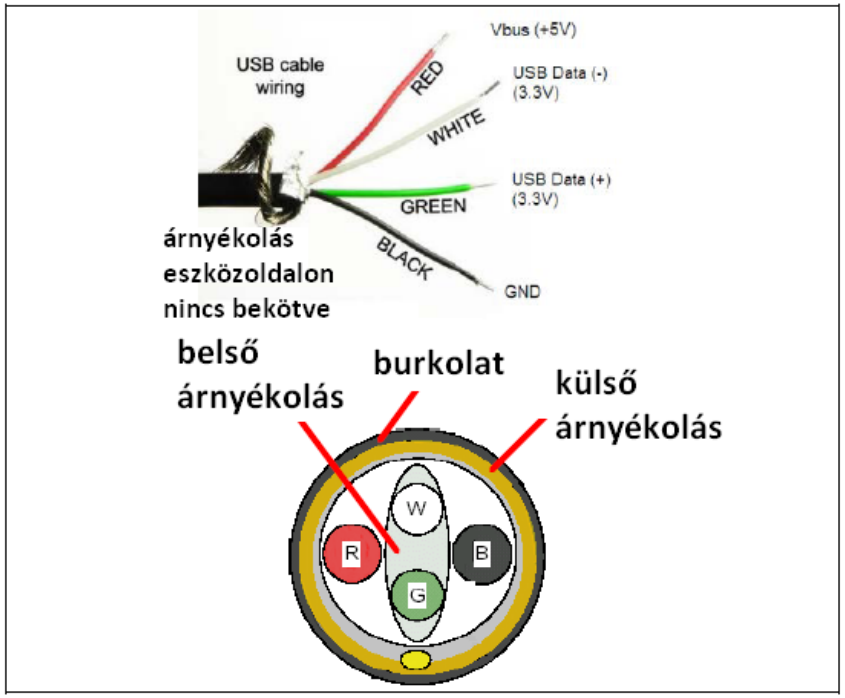
- USB = Univerzális soros busz (**U**niversal **S**erial **B**us)
- Szabvány leírása: www.usb.org
  - USB 1.0 - 1995 1,5 Mbit/s
  - USB 2.0 - 2000 1,5/12/480 Mbit/s (Full speed = 12 Mbit/s)
  - USB 3.0 - 2009 super speed: 5 Gbps (további 2 érpár felhasználásával)
- Minden eszköznek jeleznie kell, hogy 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.

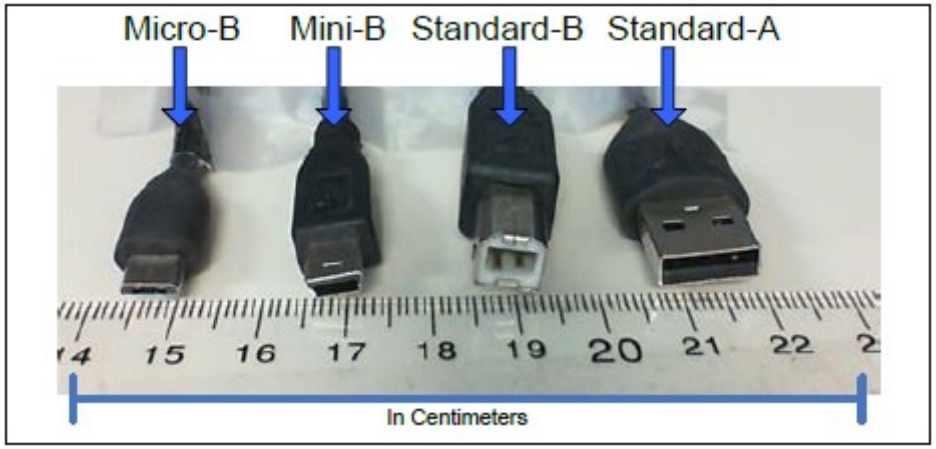
Az USB fizikai megvalósítása

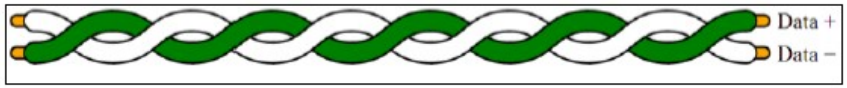
- Az USB 1.0 és 2.0 busz: sodrott adatvezetékek és tápvezetékek





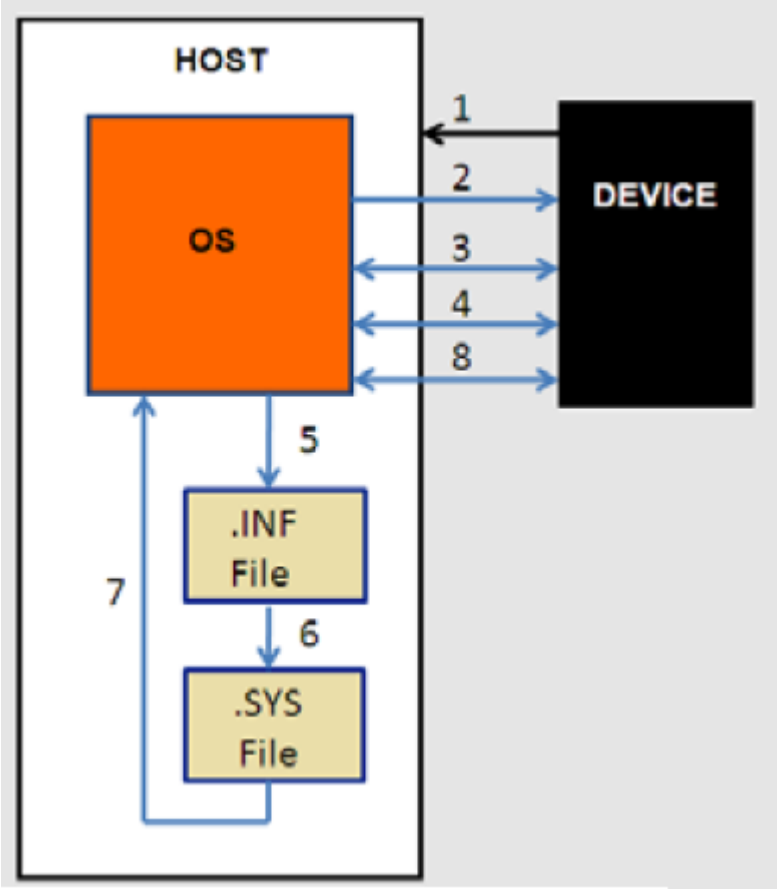






Az USB enumeráció lépései

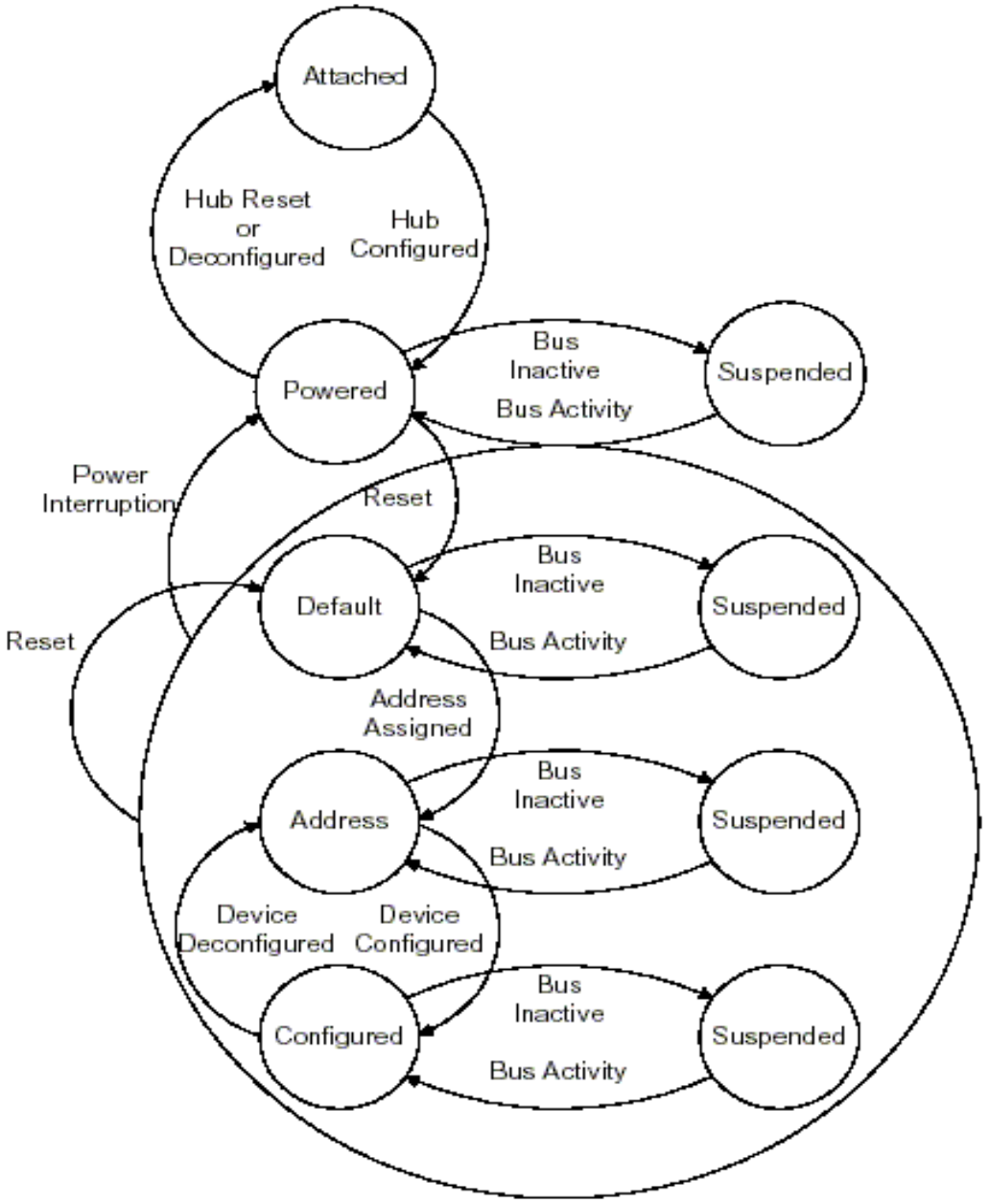
1. Az eszköz csatlakozik a gazdagéphez
2. A hoszt reset parancsot küld az eszköznek
3. Az eszköz válaszol a kérésre és a hoszt új címet állít be az eszkönek.
4. A hoszt eszközleírót kér az új címmel azonosított eszköztől. Az eszköz válaszol
5. A hoszt megkeresi és olvassa az .INF állományt
6. Az .INF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



Az eszköz konfigurált és kész a használatra

Az USB eszköz állapotdiagramja

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

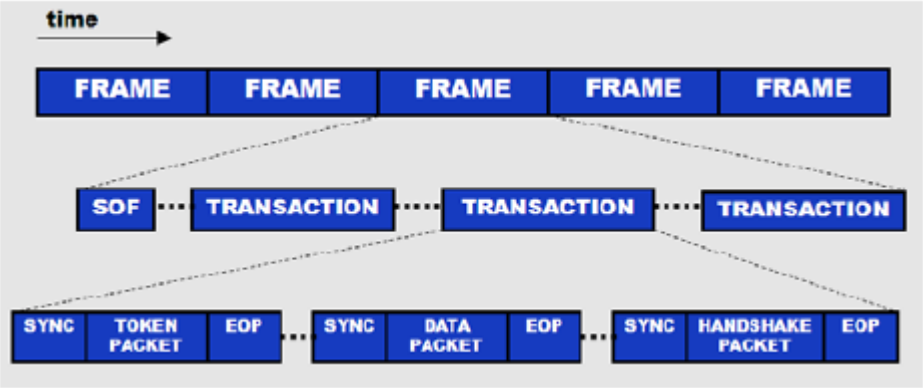


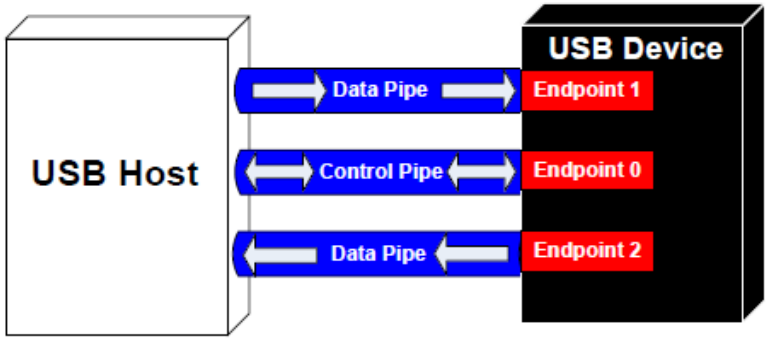
USB kommunikációs protokol

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.

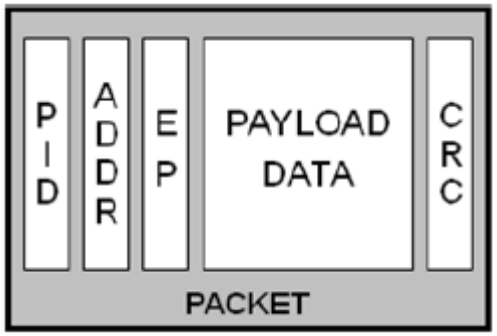
Minden eszközben kell egy vezérlő végpontnak is lennie (ez a nulladik sorszámú).





Adatcsomag felépítése:

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



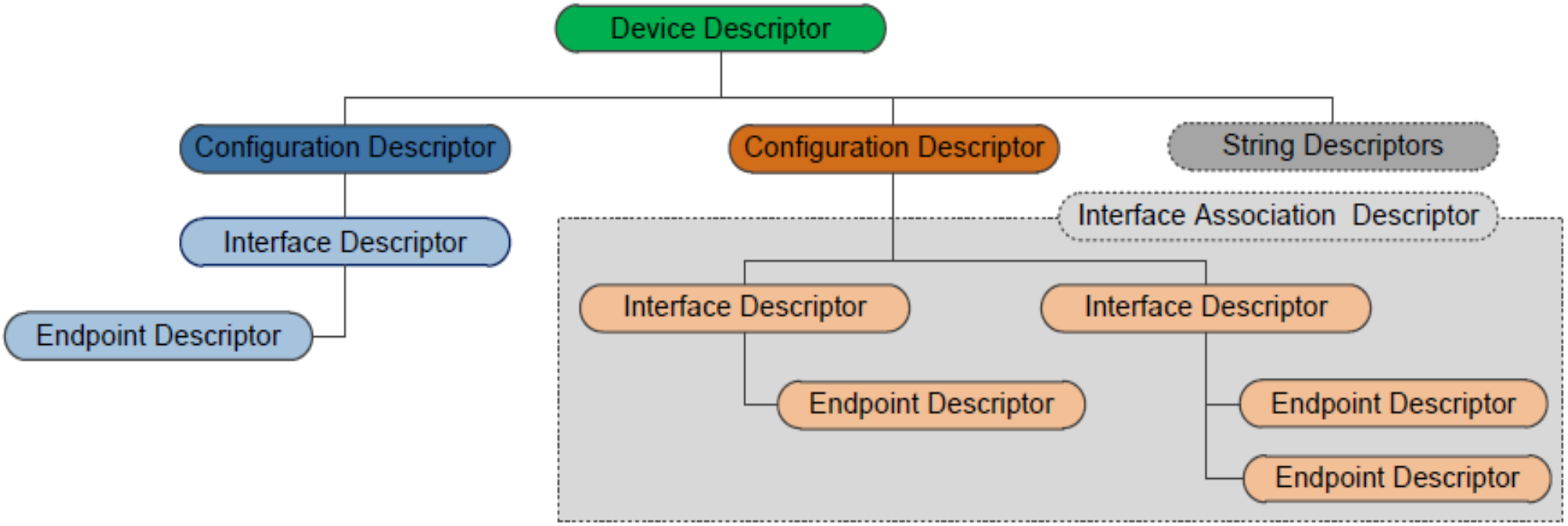
Végpont/kommunikációs osztályok

- 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 mintaprojektekben a CDC és a HID eszközosztályokkal fogunk találkozni.

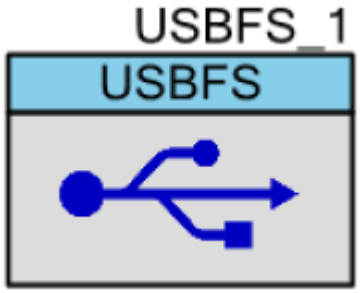
Eszközleírók (descriptors)

- Az eszköz-, konfiguráció-, interfész-, végpont- és egyéb leíró táblák az enumerációhoz szükséges adatokat tartalmazzák.
- Egy eszköz több konfigurációval is rendelkezhet (pl. bus powered és self powered mód)
- Egy konfigurációhoz több Interface tartozhat (többfunkciós eszköz)
- Az Interface leíró egy vagy több végpont leírót tartalmaz



Az USBFS komponens

- Főbb jellemzők
  - USB Full Speed (12 Mbit/s) eszköz
  - Támogatja az interrupt, control, bulk és isochronous átvitelt.
  - USB HID, CDC, MSC (mass storage class), Audio és Midi eszközosztályok támogatása



- Órajel beállítások
  - Az USB órajelet engedélyezni kell
  - ILO beállítása 100 kHz legyen
  - A Bus Clock nem lehet kevesebb 33 MHz-nél
  - PSoC 5 esetén kötelező a külső kvarccal stabilizált 24 MHz-es órajel, PSoC 5LP esetén a belső oszcillátor is használható (automatikusan hozzáigazítja magát az USB hoszt órajel frekvenciájához)

Általános USBFS API függvények

- Az alábbiakban összefoglaljuk az USBFS komponens általános célú API függvényeit

Function	Description
USBFS_Start()	Activates the Component for use with the device and specific voltage mode.
USBFS_Init()	Initializes the Component's hardware.
USBFS_InitComponent()	Initializes the Component's global variables and initiates communication with host by pull-up D+ line.
USBFS_Stop()	Disables the Component.
USBFS_GetConfiguration()	Returns the currently assigned configuration. Returns 0 if the device is not configured.
USBFS_IsConfigurationChanged()	Returns the clear-on-read configuration state.
USBFS_GetInterfaceSetting()	Returns the current alternate setting for the specified interface.
USBFS_GetEPState()	Returns the current state of the specified USBFS endpoint.
USBFS_GetEPAckState()	Determines whether an ACK transaction occurred on this endpoint.
USBFS_GetEPCount()	Returns the current byte count from the specified USBFS endpoint.
USBFS_InitEP_DMA()	Initializes DMA for EP data transfers.

Function	Description
USBFS_Stop_DMA()	Stops DMA channel associated with endpoint.
USBFS_LoadInEP()	Loads and enables the specified USBFS endpoint for an IN transfer.
USBFS_LoadInEP16()	Loads and enables the specified USBFS endpoint for an IN transfer. This API uses the 16-bit Endpoint registers to load the data.
USBFS_ReadOutEP()	Reads the specified number of bytes from the Endpoint RAM and places it in the RAM array pointed to by pSrc. Returns the number of bytes sent by the host.
USBFS_ReadOutEP16()	Reads the specified number of bytes from the Endpoint buffer and places it in the system SRAM. Returns the number of bytes sent by the host. This API uses the 16-bit Endpoint registers to read the data.
USBFS_EnableOutEP()	Enables the specified USB endpoint to accept OUT transfers.
USBFS_DisableOutEP()	Disables the specified USB endpoint to NAK OUT transfers.
USBFS_SetPowerStatus()	Sets the device to self-powered or bus-powered.
USBFS_Force()	Forces a J, K, or SE0 State on the USB Dp/Dm pins. Normally used for remote wakeup.
USBFS_SerialNumString()	Provides the source of the USB device serial number string descriptor during run time.
USBFS_TerminateEP()	Terminates endpoint transfers.
USBFS_VBusPresent()	Determines VBUS presence for self-powered devices.
USBFS_Bcd_DetectPortType()	Determines if the host is capable of charging a downstream port.
USBFS_GetDeviceAddress()	Returns the currently assigned address for the USB device.
USBFS_EnableSofInt()	Enables interrupt generation when a Start-of-Frame (SOF) packet is received from the host.
USBFS_DisableSofInt	Disables interrupt generation when a Start-of-Frame (SOF) packet is received from the host.

Az USB tápfeszültség beállítása

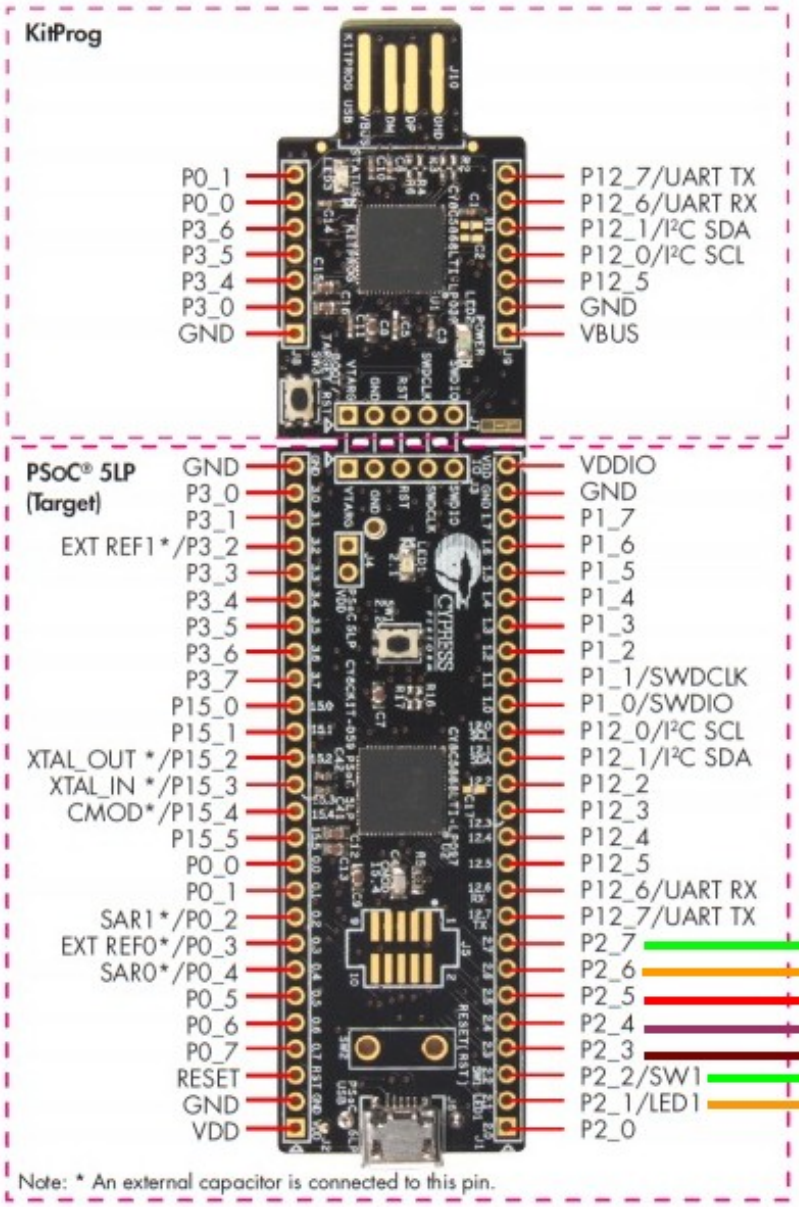
- Az USB működéséhez 3,3 V-os feszültséget kell biztosítani. Ennek beállítását az **USB_Start()** API függvény hívásakor kell megadni.

```
void USBFS_Start(uint8 device, uint8 mode)
```

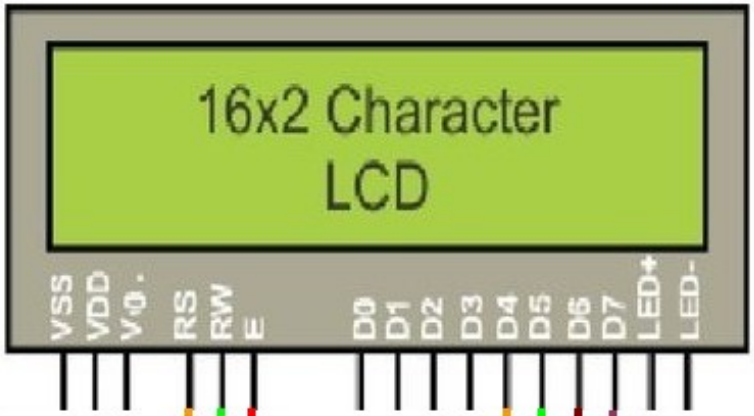
- Ha VDDD = 5 V, akkor **mode** = **USBFS_5V_OPERATION** legyen! Ezzel engedélyezzük a belső feszültségstabilizátort, ami előállítja az USB alrendszernek a 3,3 V-ot.
- Ha VDDD = 3,3 V, akkor **mode** = **USBFS_3V_OPERATION** legyen! Ezzel letiltjuk a belső feszültségstabilizátort, az USB alrendszer közvetlenül a VDDD tápfeszültségre csatlakozik.
- A **mode** = **USBFS_DWR_VDDD_OPERATION** beállítás esetén a PSoC Creatorban a .dwr (Design Wide Resources) állományban a **System settings** lapon beállított tápfeszültséghez igazodóan lesz engedélyezve vagy tiltva a belső feszültségstabilizátor.

USBFS_UART01 projekt

A **CE60 246** mintaalkalmazás adaptációja a **CY8CKIT-059** kártyához. Visszatükrözi a Hyperterminálból (vagy más terminál emulátor szoftverből) érkező karaktereket, s a **CY8CKIT-059** kártyához kapcsolt alfanumerikus LCD kijelzőn kiírja a soros porti beállításokat.

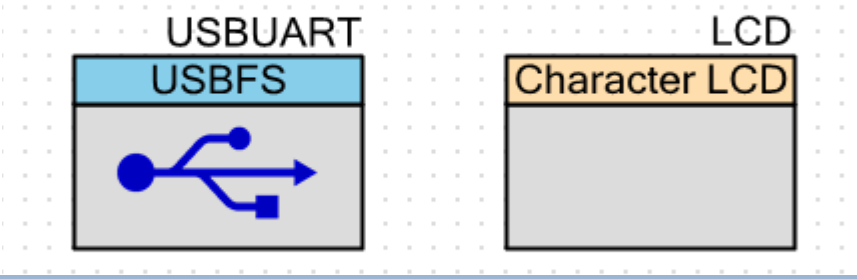


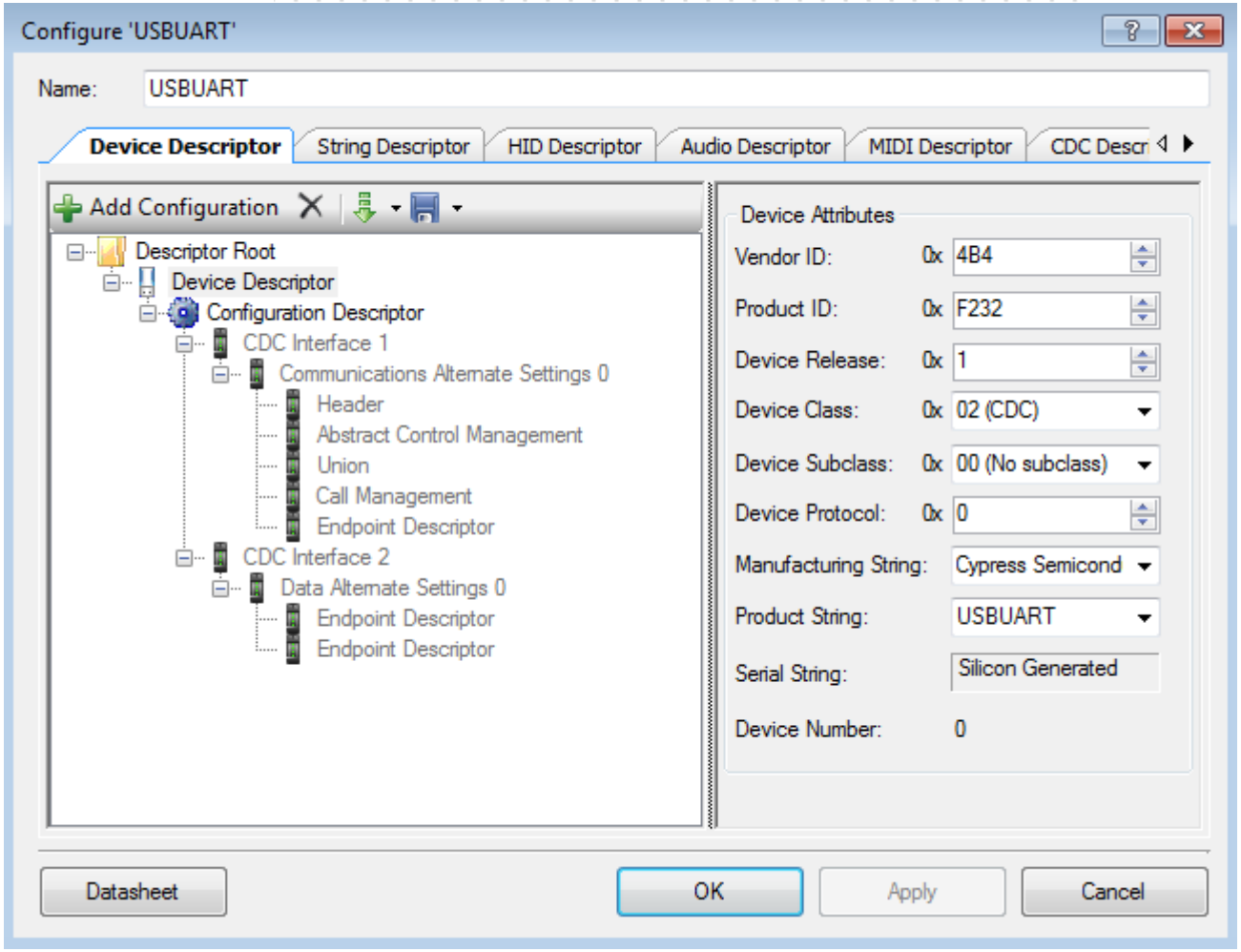
Bekötés
LCD_port = P2[7:1]
választás esetén



USBFS_UART01 projekt

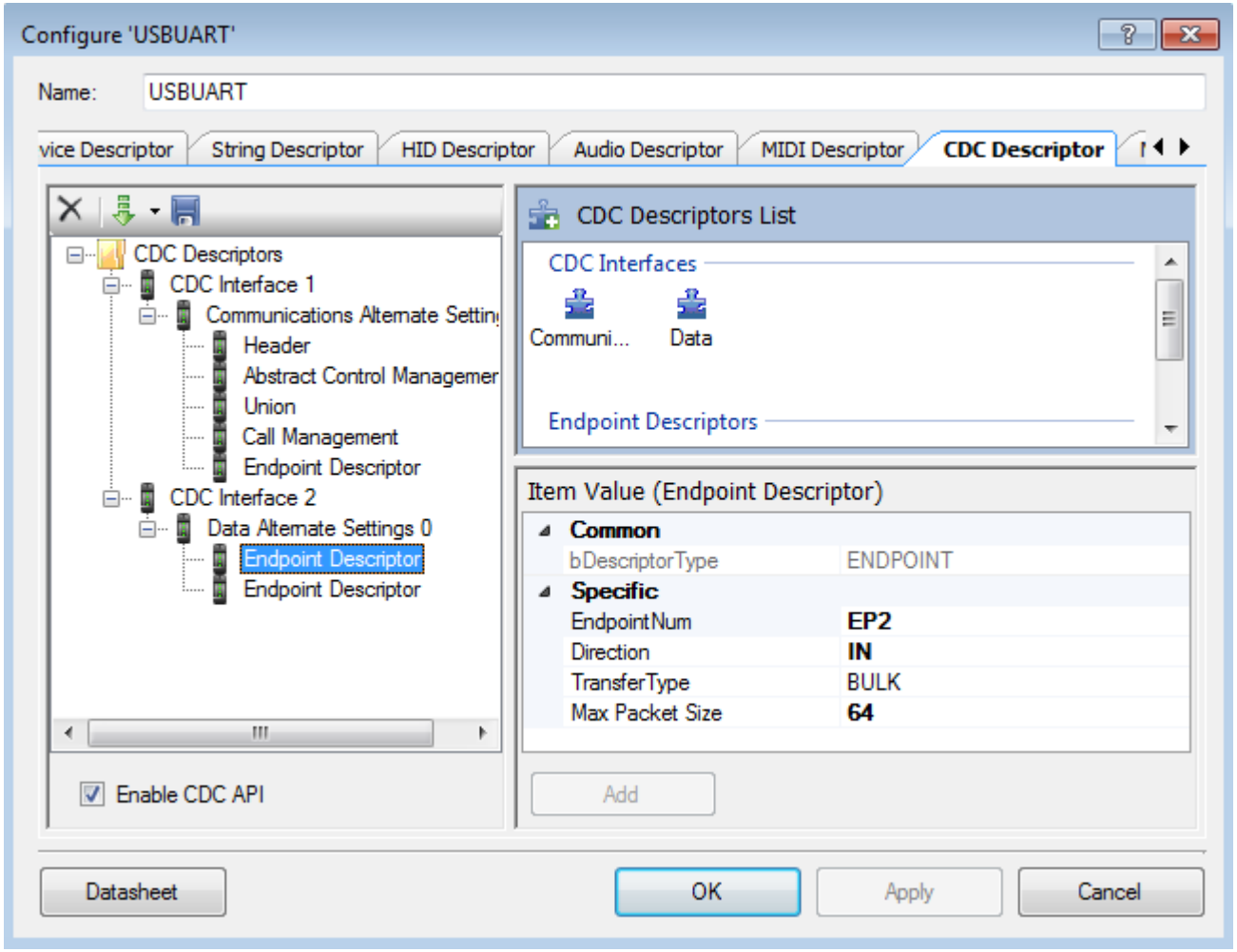
- Az **USBFS** példány neve itt **USBUART**
- Az eszközt CDC (kommunikációs eszközosztály) módba konfiguráljuk
- VID: 04B4 a Cypress azonosítója, csak kísérletezéskor használhatjuk
- PID: F232 a „termék” azonosítója





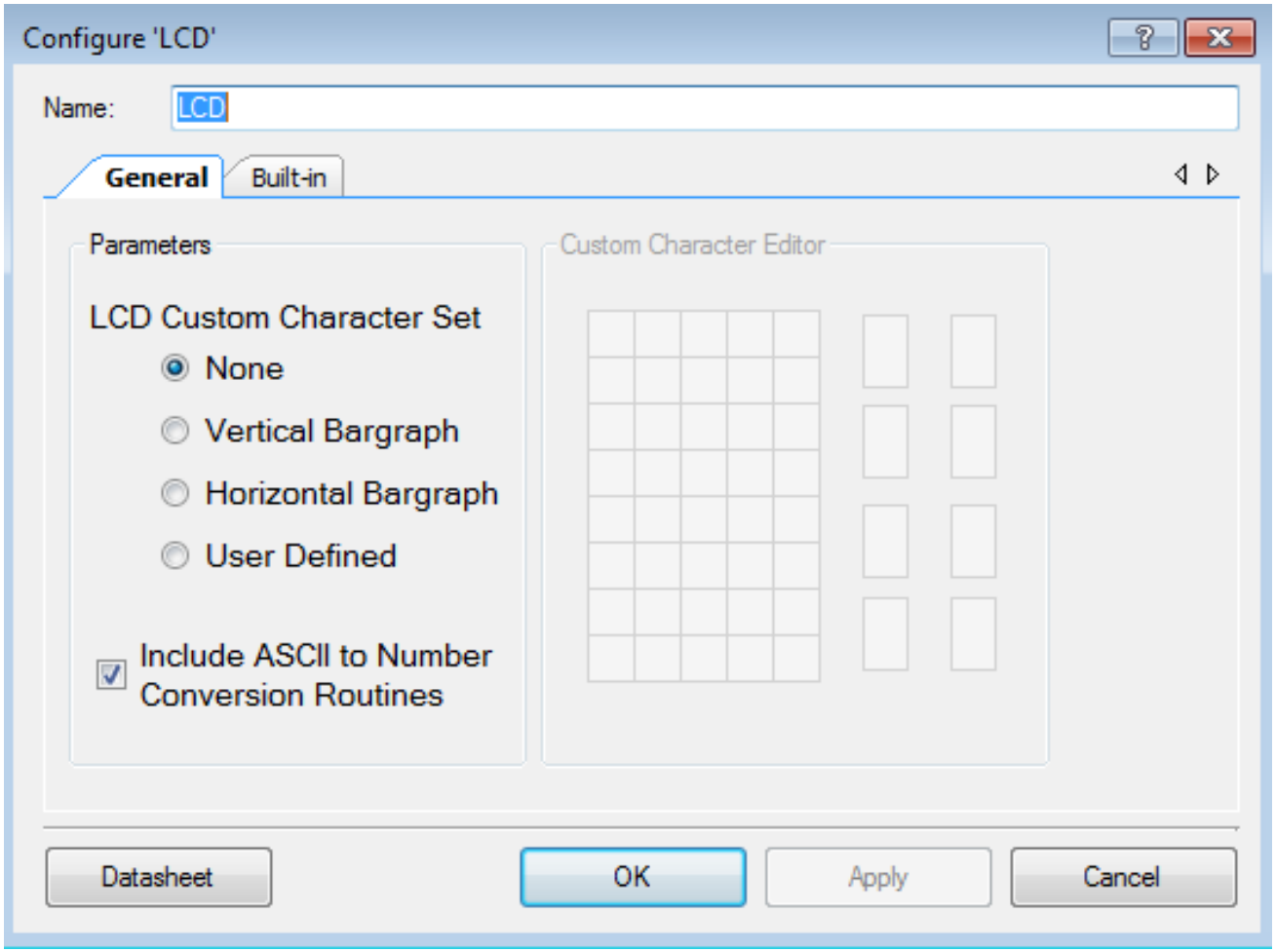
USBFS_UART01 projekt

- Az USB_CDC eszközosztályhoz négy végpontot rendelünk:
- **EP0** a vezérlő csatorna, ez minden eszköznél alapértelmezett
- **EP1** 8 bájtos INTERRUPT csatorna (adatfolyam vezérlés)
- **EP2** Bulk IN, 64 bájt adatcsatorna
- **EP3** Bulk, OUT, 64 bájt adatcsatorna



USBFS_UART01 projekt

- Az alfanumerikus LCD konfigurálásakor nincs szükségünk speciális karakterek definiálására
- Az ASCII → numerikus konverzió opcionális



A főprogram fontosabb részei

```
CyGlobalIntEnable;
USBUART_Start(0u, USBUART_DWR_VDDD_OPERATION);
while(!USBUART_GetConfiguration());              //Wait for Device to enumerate
USBUART_CDC_Init();
for(;;) {
    if(USBUART_DataIsReady() != 0) {
       count = USBUART_GetAll(buffer);
       if(count != 0u) {
           while(USBUART_CDCIsReady() == 0);
           USBUART_PutData(buffer, count);
       }
    }
    state = USBUART_IsLineChanged();             //Line settings changed?
    if(state != 0) {
       if(state&USBUART_LINE_CODING_CHANGED)     // Show new settings
       { //Display UART settings getting by USBUART_GetDTERate(),
         //USBUART_GetDataBits(), USBUART_GetCharFormat(), USBUART_GetParityType()
       }
    }
    if(state & USBUART_LINE_CONTROL_CHANGED) {   //Show new settings
       state = USBUART_GetLineControl();
       //Display state & USBUART_LINE_CONTROL_DTR
       //Display state & USBUART_LINE_CONTROL_RTS
    }
}
```

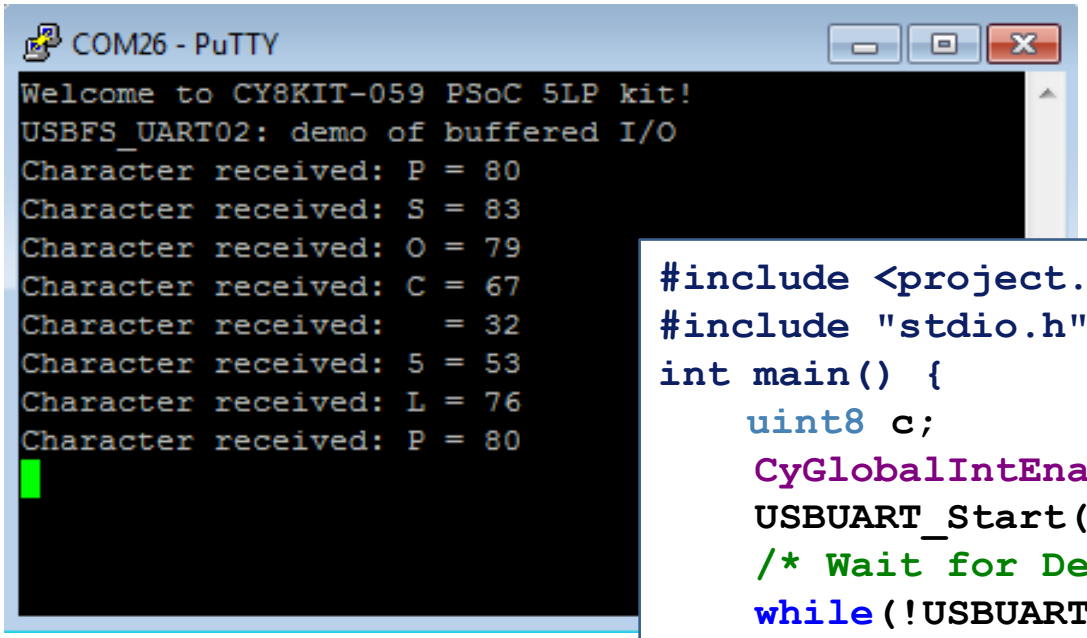
A vett karakterek visszatükrözése

Az USB_UART02 mintapélda

- Az előző programot kibővítettük az USB-UART adatforgalom kettős bufferelésével
- Implementáltunk egy **getc()** és egy **putc()** függvényt
- Megoldottuk a **printf()** standard output átirányítását
- A főprogram kiírja a kapott karaktert és annak ASCII kódját.

USBFS_UART02 főprogram

- A program kiírja a számítógéptől kapott karaktert és annak ASCII kódját.




```
#include <project.h>
#include "stdio.h"
int main() {
    uint8 c;
    CyGlobalIntEnable;  /* Enable Global Interrupts */
    USBUART_Start(0u, USBUART_DWR_VDDD_OPERATION);
    /* Wait for Device to enumerate */
    while(!USBUART_GetConfiguration());
    USBUART_CDC_Init();
    while(!USB_cdc_DTR) ProcessIO(); //wait for connect
    printf("Welcome to CY8KIT-059 PSoC 5LP kit!\r\n");
    printf("USBFS_UART02: demo of buffered I/O\r\n");
    for(;;) {
        c=USB_cdc_getc();
        printf("Character received: %c = %d\r\n",c,c);
    }
}
```

USBFS_UART02: kettős bufferelés

```
uint8 USB_In_Buffer[64];        // USB input buffer (seeing from host side)
uint8 USB_Out_Buffer[64];       // USB output buffer (seeing from host side)
uint8 numBytesRead=0;           // Number of bytes received from USB host
uint8 numBytesToSend=0;         // Number of bytes to be sent to host
uint8 Buffercp=0;               // Buffer pointer for getc()
uint8 USB_cdc_RTS=0;            // RTS control signal of the virtual COM port
uint8 USB_cdc_DTR=0;            // DTR control signal of the virtual COM port

void ProcessIO(void) {          // USB I/O feldolgozás
  uint16 state;
  //-- Read data from host if the input buffer is empty
  if (USBUART_DataIsReady() && (numBytesRead==Buffercp)) {
      numBytesRead = USBUART_GetAll(USB_Out_Buffer);
      Buffercp=0;
    }
  if(USBUART_IsLineChanged() & USBUART_LINE_CONTROL_CHANGED) {
     state = USBUART_GetLineControl(); //-- Update RTS and DTR states
     USB_cdc_RTS = state & USBUART_LINE_CONTROL_RTS;
     USB_cdc_DTR = state & USBUART_LINE_CONTROL_DTR;
  }
//-- Send data to host if the output buffer is not empty
  if (USBUART_CDCIsReady() && (numBytesToSend!=0)) {
      USBUART_PutData(USB_In_Buffer,numBytesToSend);
      numBytesToSend=0;
  }
}       //end ProcessIO
```

getc() és putc() implementálása

```
/*****************************************************
 * Read one character from the input buffer, or wait for it
 * if the buffer is empty. This is a blocking function
 * therefore it must call ProcessIO() repeatedly.
 *****************************************************/
uint8 USB_cdc_getc(void)
{
    while (Buffercp==numBytesRead) ProcessIO();
    return(USB_Out_Buffer[Buffercp++]);
}

/*****************************************************
 * Write one character into the input buffer, or wait for it
 * if the buffer is full. This is a blocking function
 * therefore it must call ProcessIO() repeatedly.
 *****************************************************/
int USB_cdc_putc(uint8 c)
{
    while (numBytesToSend>60)   ProcessIO();
    USB_In_Buffer[numBytesToSend++]=c;
    return (int)c;
}
```

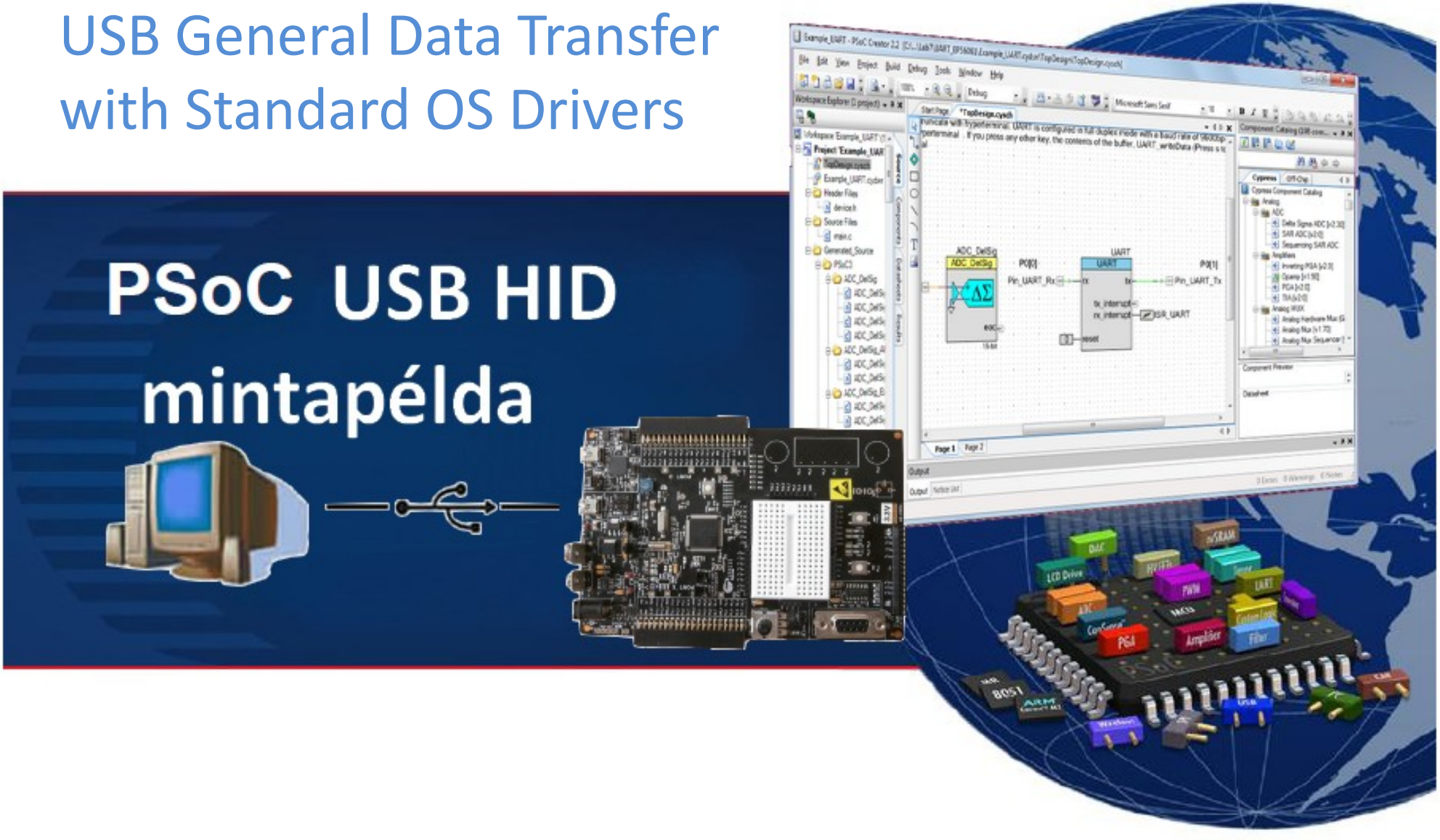
A printf() kimenetének átirányítása

```
/**********************************************************
 * If we need the printf() function, we must redefine the
 * _write() function for PSoC 5LP (GCC compiler)
 *
 * We also add an explicit reference to the floating point
 * printf() library to allow the usage of floating point
 * conversion specifier (optional)
 * Ref: http://www.cypress.com/?app=forum&id=2233&rID=75601
 **********************************************************/
int _write(int file, char *ptr, int len)
{
    int i;
    for (i = 0; i < len; i++)
    {
          USB_cdc_putc(*ptr++);
    }
    return len;
}

#if defined (__GNUC__)
    asm (".global _printf_float");
#endif
```

AN82072

USB General Data Transfer with Standard OS Drivers



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.

Mi a Generic HID?

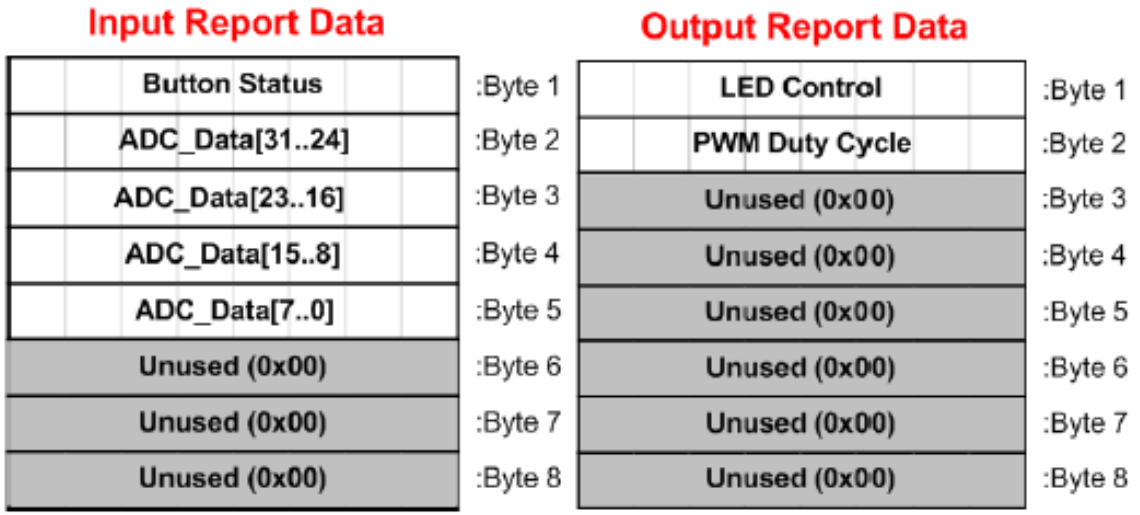
- **A szabványos HID eszközöknél** az ún. HID report descriptor (HID jelentés leíró tábla) megmondja, hogy az átvitt adatcsomag egyes bájtjai (vagy bitjei) mit jelentenek.
- **Az általános (generic) HID eszközöknél** a HID report descriptor csak az átvitt adatok mennyiségét mondja meg, azok értelmezése 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 driver (az oprendszer „tudja” kezelni az eszközt)
  - Garantált átviteli időzítés

Adatforgalom a mintaprojektben

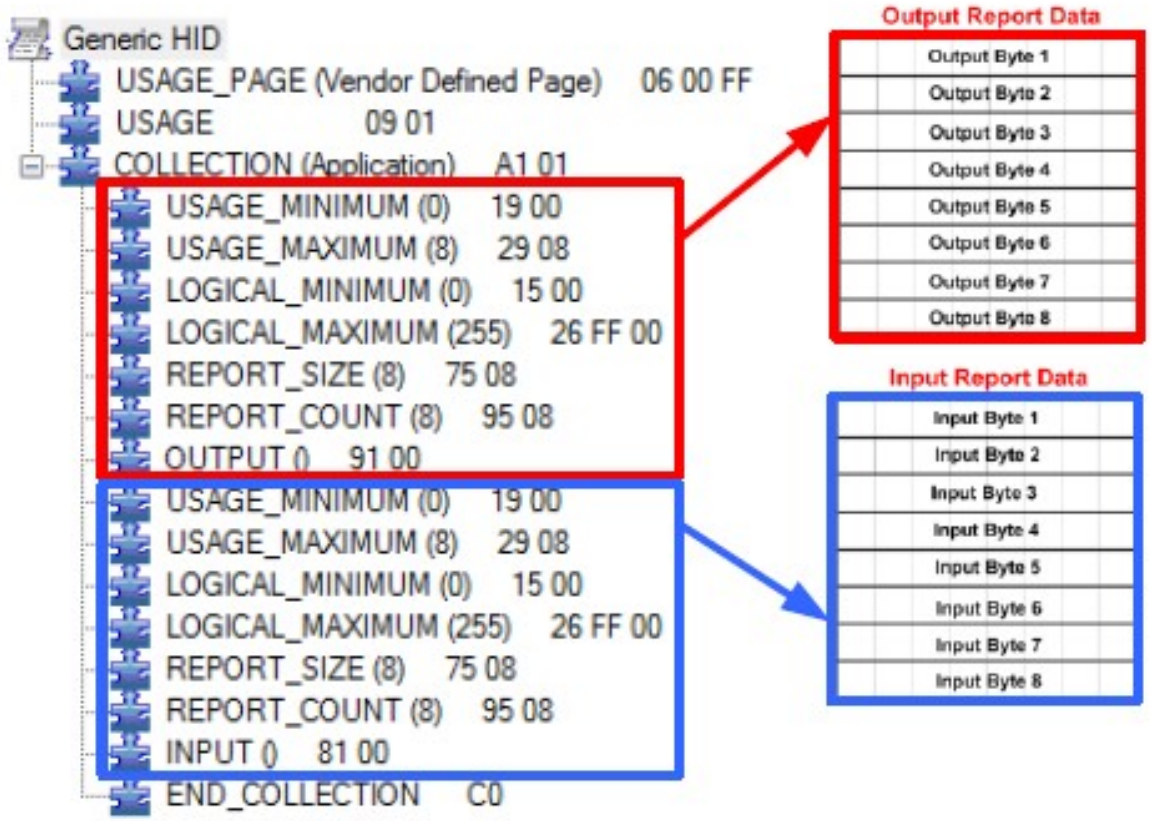
- Az adatáramlási irány mindig a gazdagép szemszögéből értendő!

Jelentés	Funkció	Bájt
Bemenet	Nyomógomb állapota (le van-e nyomva?)	1
Bemenet	ADC mérési eredmény	4
Kimenet	LED vezérlés (be vagy ki?)	1
Kimenet	PWM kitöltési tényező	1

- A 8 bájtos adatcsomagokat csak a firmware és a PC alkalmazás fogja tudni értelmezni!



A Generic HID leíró tábla

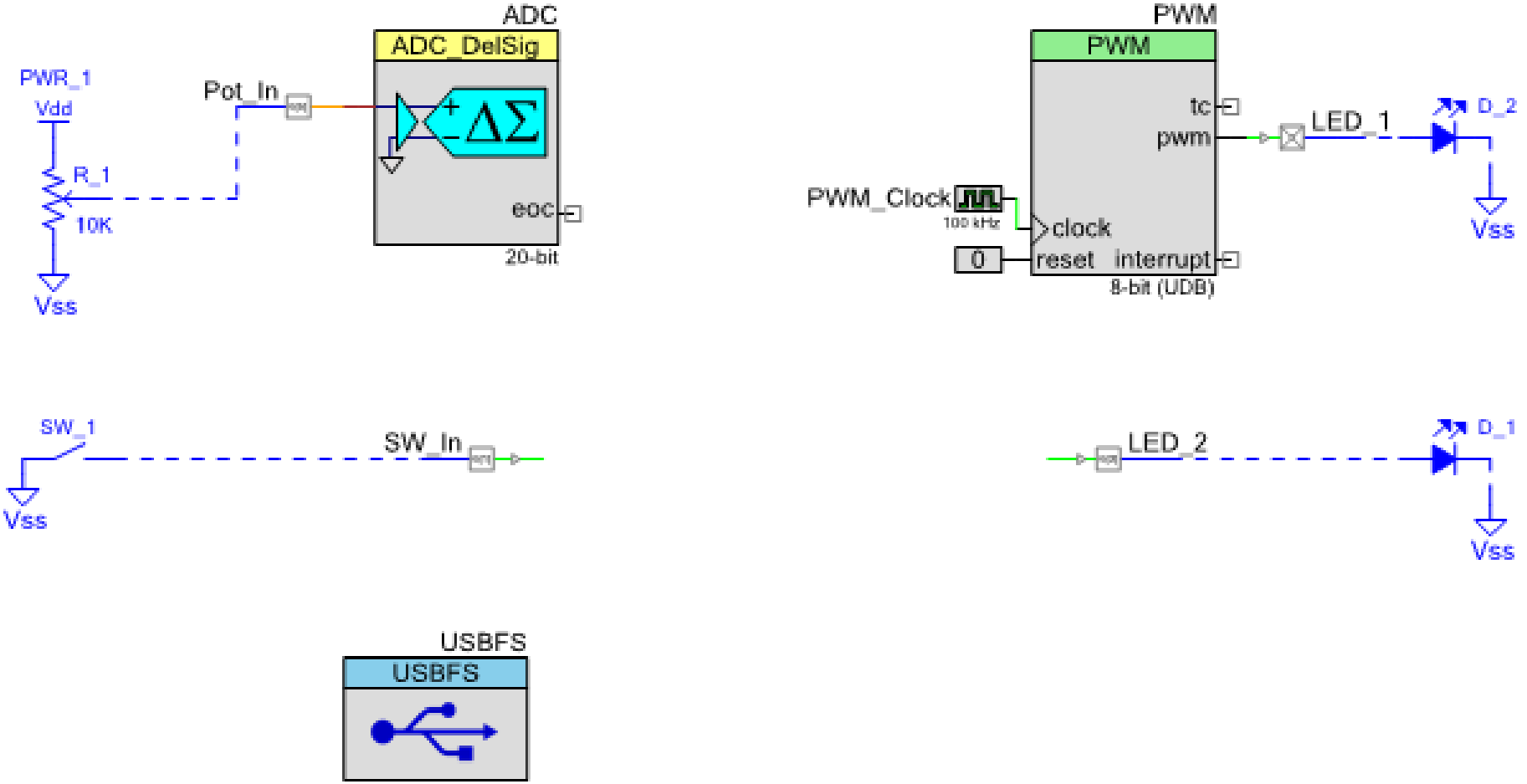


- A tábla gyártóspecifikus jelentéseket definiál, külön a kimenet és a bemenet számára.
- Bővebb leírás az AN82 072 és az AN57 473 dokumentumokban található.

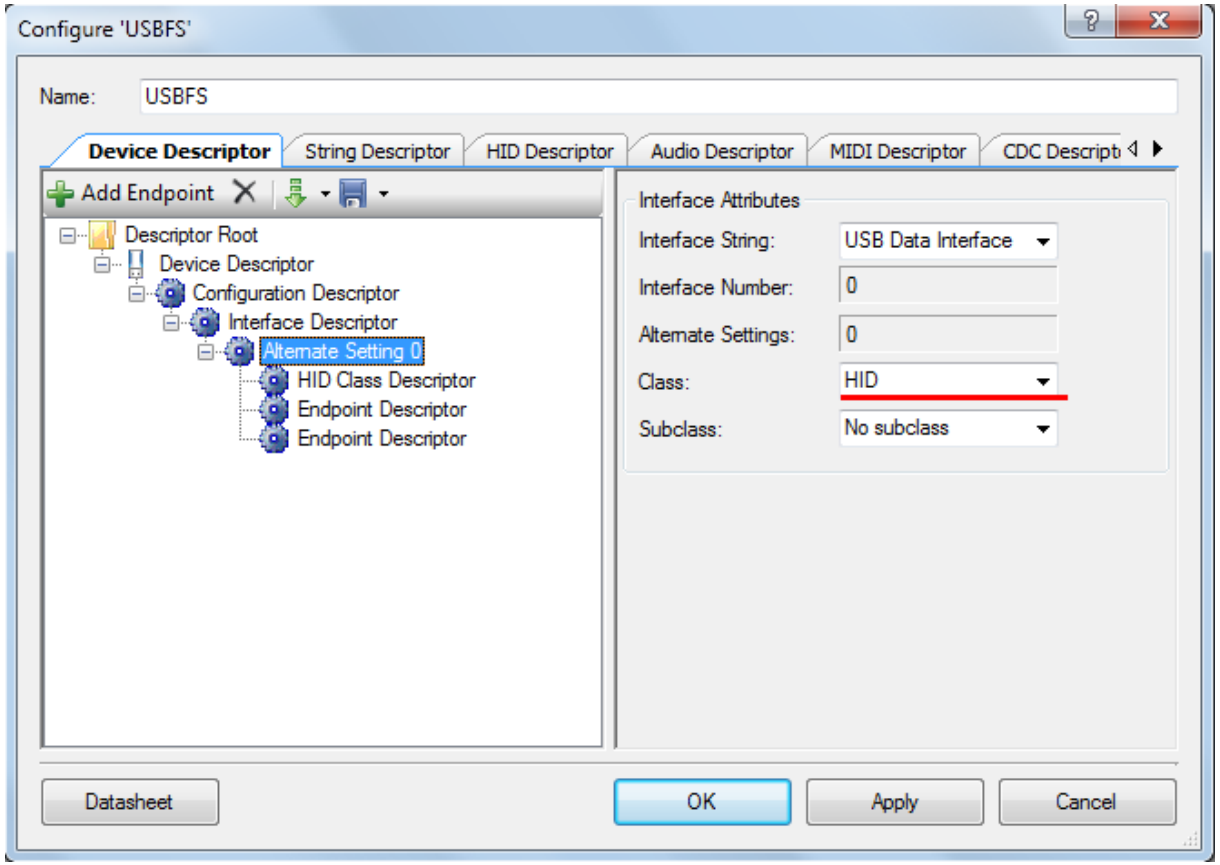
Az AN82072 mintaprojekt

Input

Output

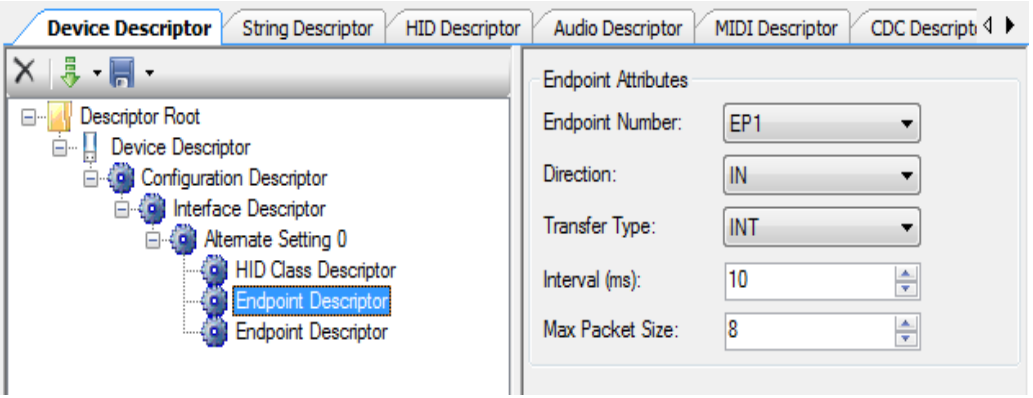


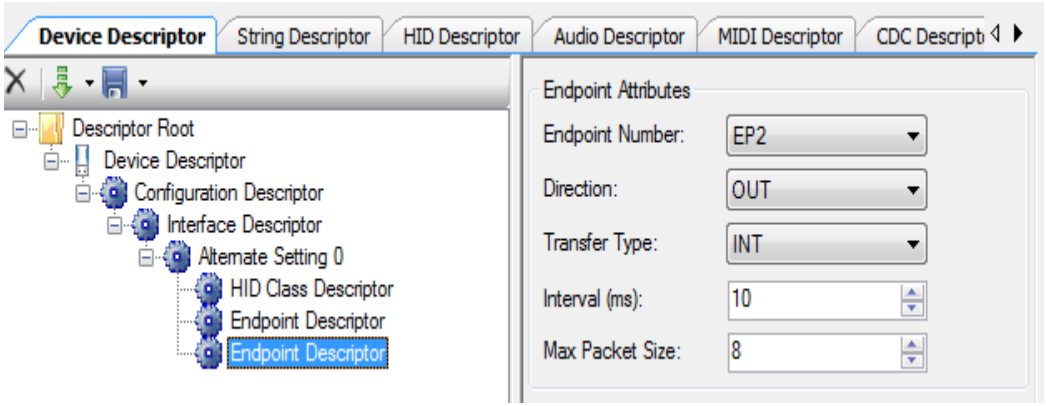
USBFS konfigurálása



EP1: IN, azaz a PC felé megy az adat (a mikrovezérlő küldi a PC-nek)

EP2: OUT azaz a PC felől jön az adata (a PC alkalmazás küldi)





Részletek a programból

```
/* Input and Output Data Buffers */
uint8 IN_Data_Buffer[8];  /* [0]=Button Status,[1..4]=ADC Result, [5..7]=Unused */
uint8 OUT_Data_Buffer[8]; /* [0]=LED State, [1]=PWM Duty Cycle, [2..7] = Unused */

/* Function Prototypes */
void Process_EP2 (void);
void Process_EP1 (void);
void Update_LCD (void);

void main (void) {
    CYGlobalIntEnable;    /*Enable Global Interrupts */
    ADC_Start();          /*Initalize ADC */
    ADC_StartConvert();
    LCD_Start();          /*Initalize LCD and place static text */
    PWM_Start();
    PWM_WriteCompare(10);
    LED_2_Write(1);
    USBFS_Start(0, USBFS_DWR_VDDD_OPERATION);
    while(USBFS_bGetConfiguration() == 0x00);
```

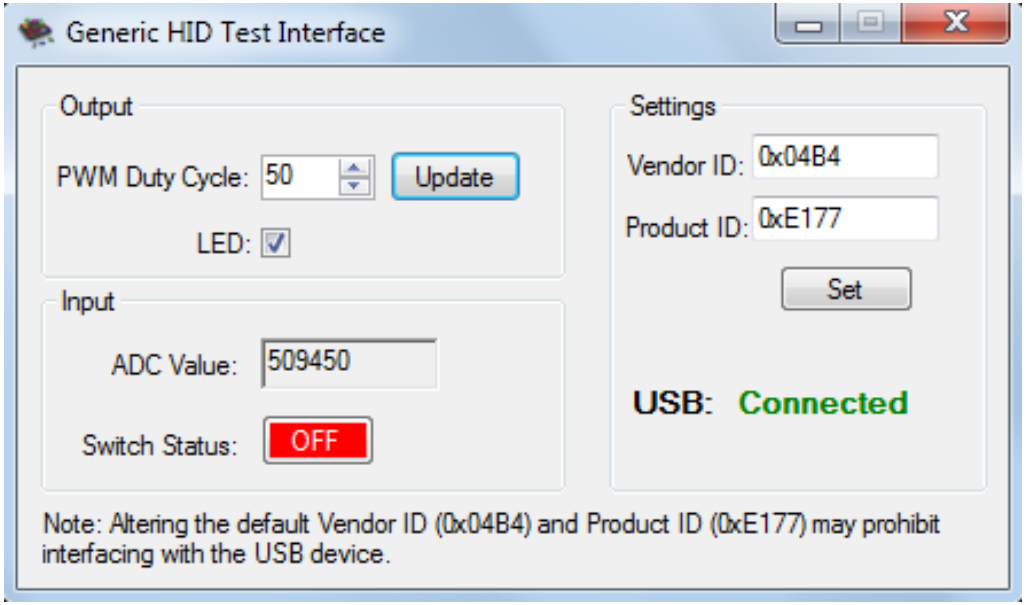
… folytatás a következő lapon

További részletek

```
for(;;)  {
  Process_EP1();
  /*Check if the IN Endpoint is empty. Load Input data to be tranfered */
  if(USBFS_GetEPState(IN_ENDPOINT) == USBFS_IN_BUFFER_EMPTY) {
      USBFS_LoadEP(IN_ENDPOINT, IN_Data_Buffer, MAX_NUM_BYTES);
      USBFS_EnableOutEP(OUT_ENDPOINT);
  }
  if(USBFS_GetEPState(OUT_ENDPOINT) == USBFS_OUT_BUFFER_FULL) {
    /* Get the number of bytes recieved */
    OUT_COUNT = USBFS_GetEPCount(OUT_ENDPOINT);
    /* Read the OUT endpoint and store data in OUT_Data_Buffer */
    USBFS_ReadOutEP(OUT_ENDPOINT, OUT_Data_Buffer, OUT_COUNT);
    /* Enable the OUT endpoint to recieve data */
    USBFS_EnableOutEP(OUT_ENDPOINT);
  }
  Process_EP2();   /* Process the data recieved from the host */
  /* Update LCD (not shown here…) */
}
```

Process_EP1() - ADC konverzió eredményének és a nyomógomb állapotának beolvasása és betöltése az **IN_Data_Buffer** tömbbe.
Process_EP2() - A LED és a PWM beállítása az **OUT_Data_Buffer** tömbből.

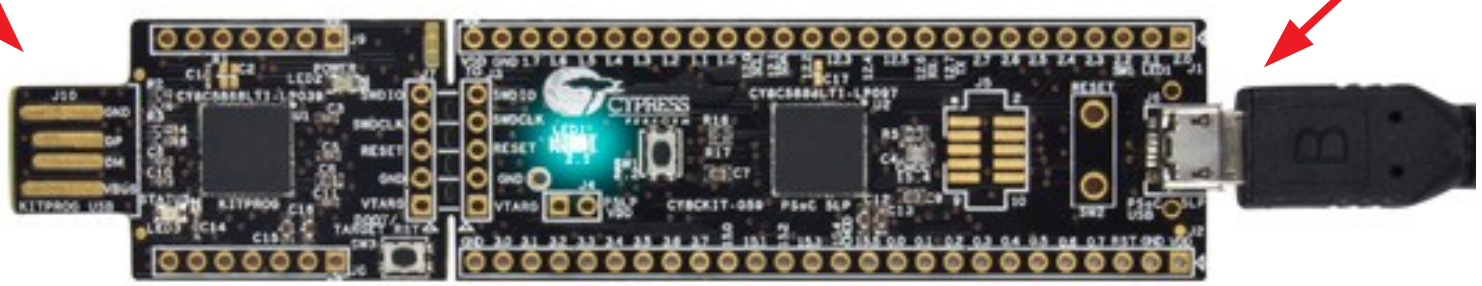
Programletöltés, futtatás



A PC alkalmazáshoz a Generic HID UI.exe program mellett a CyUSB.dll állományra is szükség van

Programletöltésnél ezt a csatlakozót használjuk!

Programfuttatásnál ezt a csatlakozót használjuk!



CY8CKIT-059 fejlesztői kártya

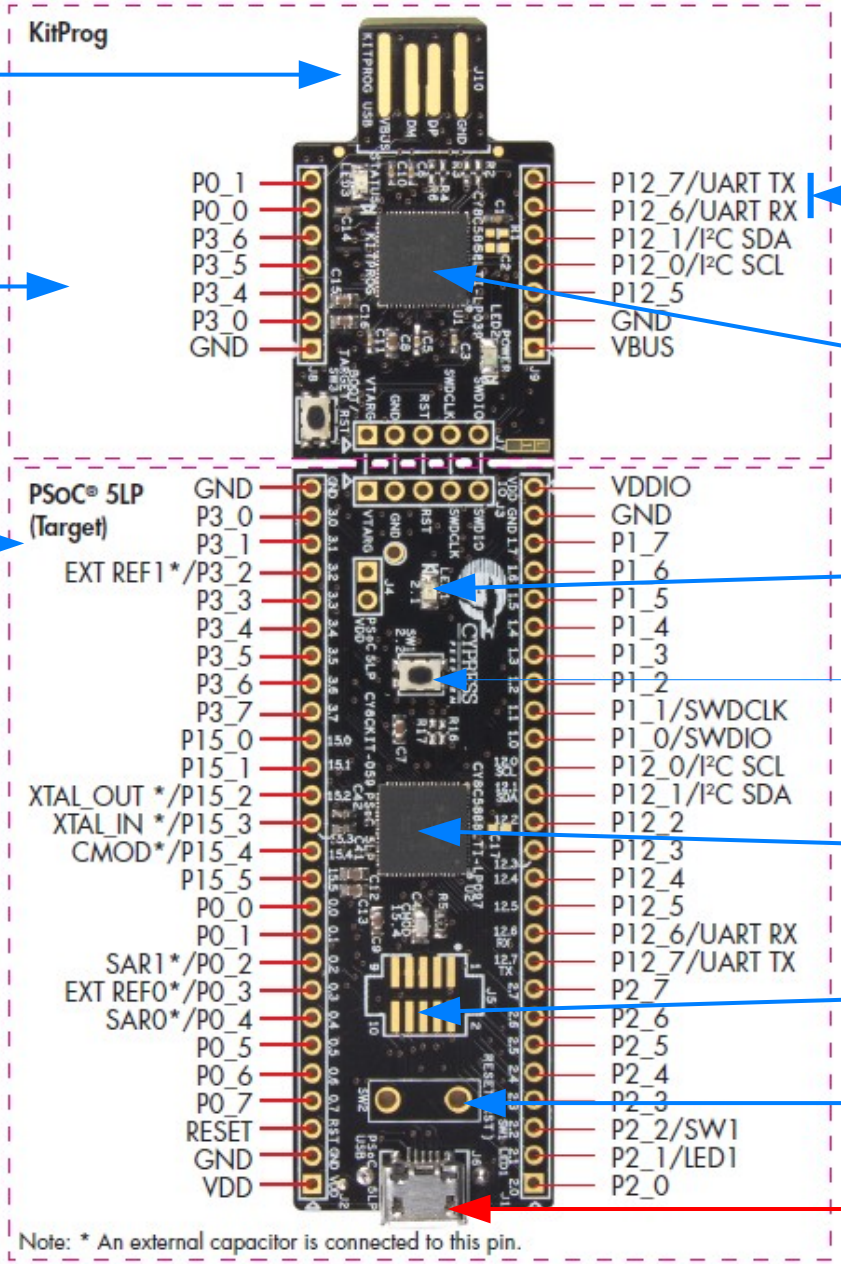
USB csatlakozás a PC-hez

KitProg programozó és hibavadász

PSOC 5LP Target áramkör

A tápellátás történhet a programozó felől (5V), Az alkalmazói USB csatlakozóról (5V), vagy a VDD csatlakozáson keresztül (3,3 – 5 V).

Utóbbi esetben a D1 és D2 diódákat el kell távolítani az USB-re csatlakozás előtt!



USB – UART Kivezetések

C8C5868LTI-LP039

LED1 (2.1 kivezetés)

SW1 (2.2 kivezetés)

CY8C5888LTI-LP097

JTAG csatlakozás

RESET gomb helye

USB alkalmazói csatl.

A céláramkör kapcsolási rajza