ARM Cortex Core mikrovezérlők

Universal Serial Bus
Gyakorlat

Scherer Balázs

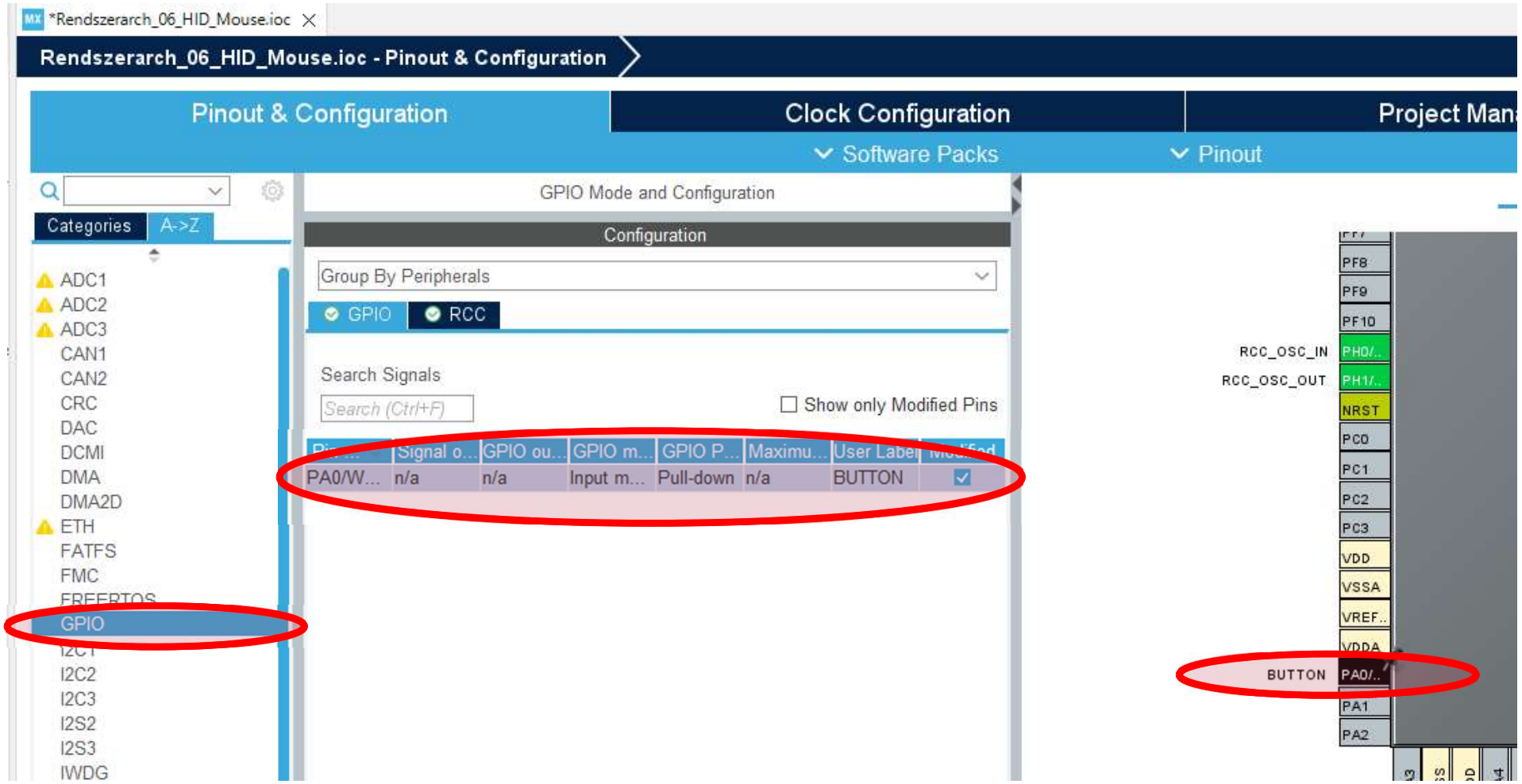
Méréstechnika és
Információs Rendszerek
Tanszék

HID (Human Interface Device)

egér

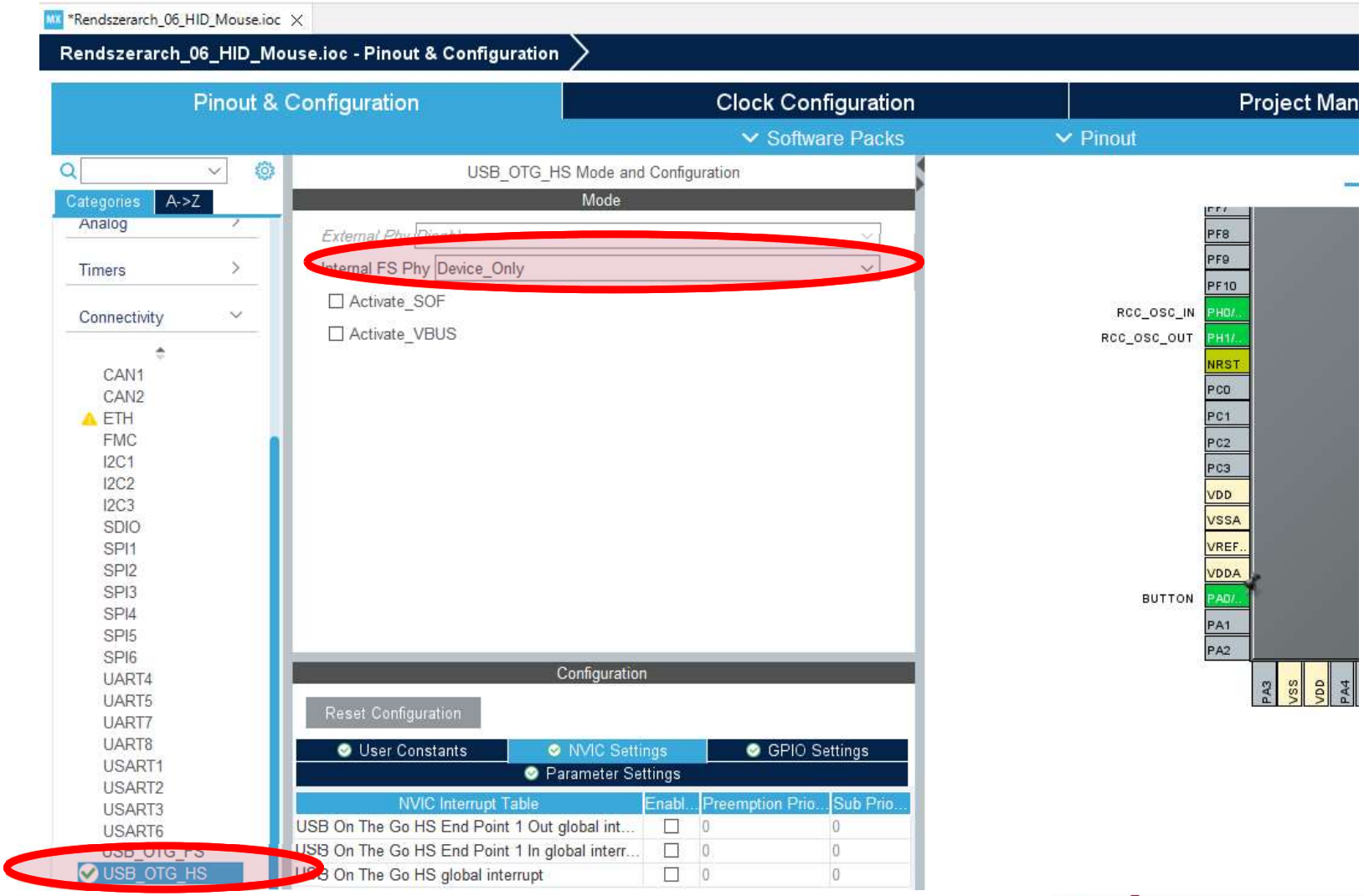
Cube IDE beállítás

- PA0 nyomógomb funkció: GPIO bemenet, pulldown-al



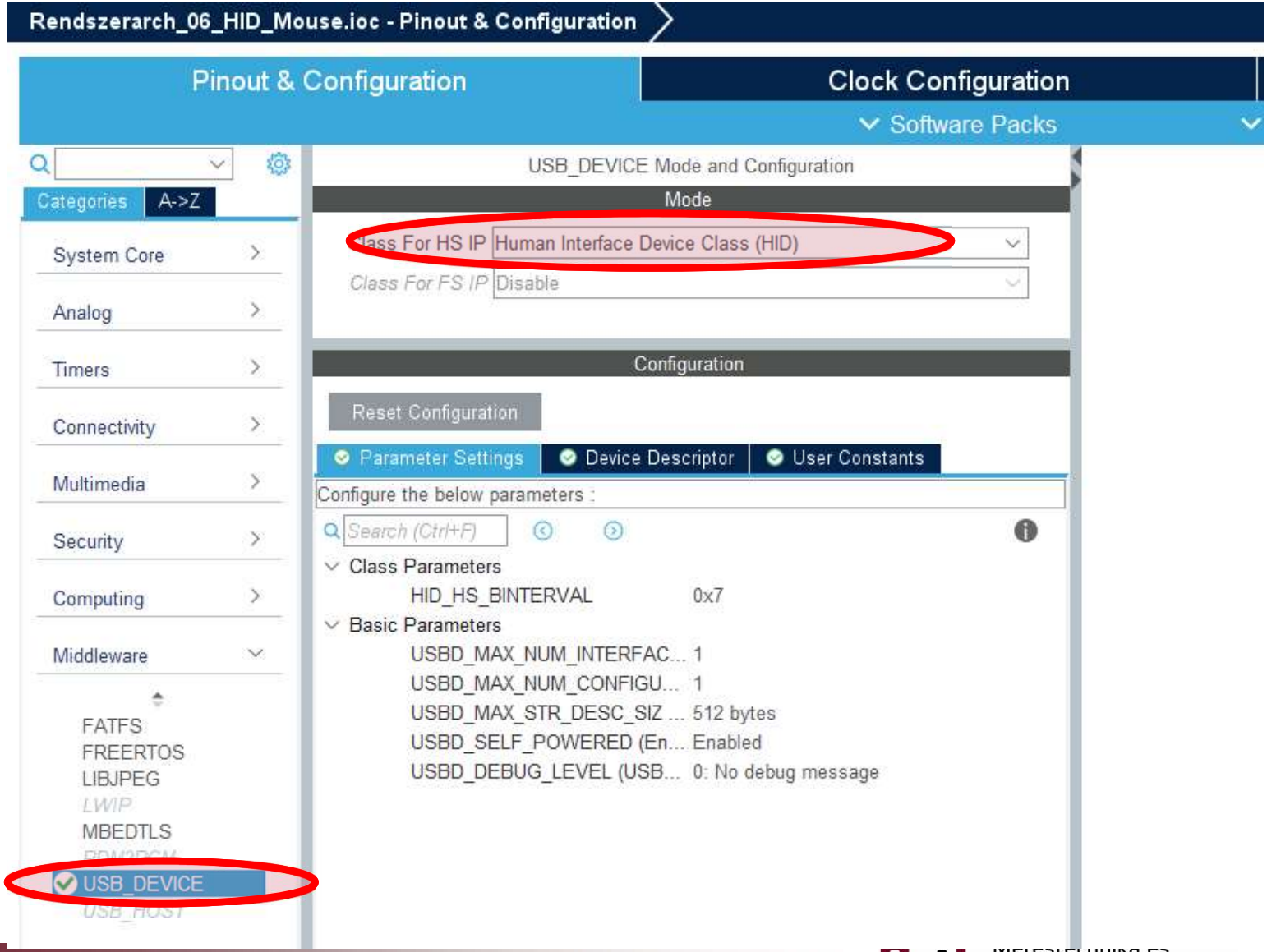
Cube IDE beállítás

- USB beállítása, High speed USB belső Full speed fizikai réteg illesztéssel



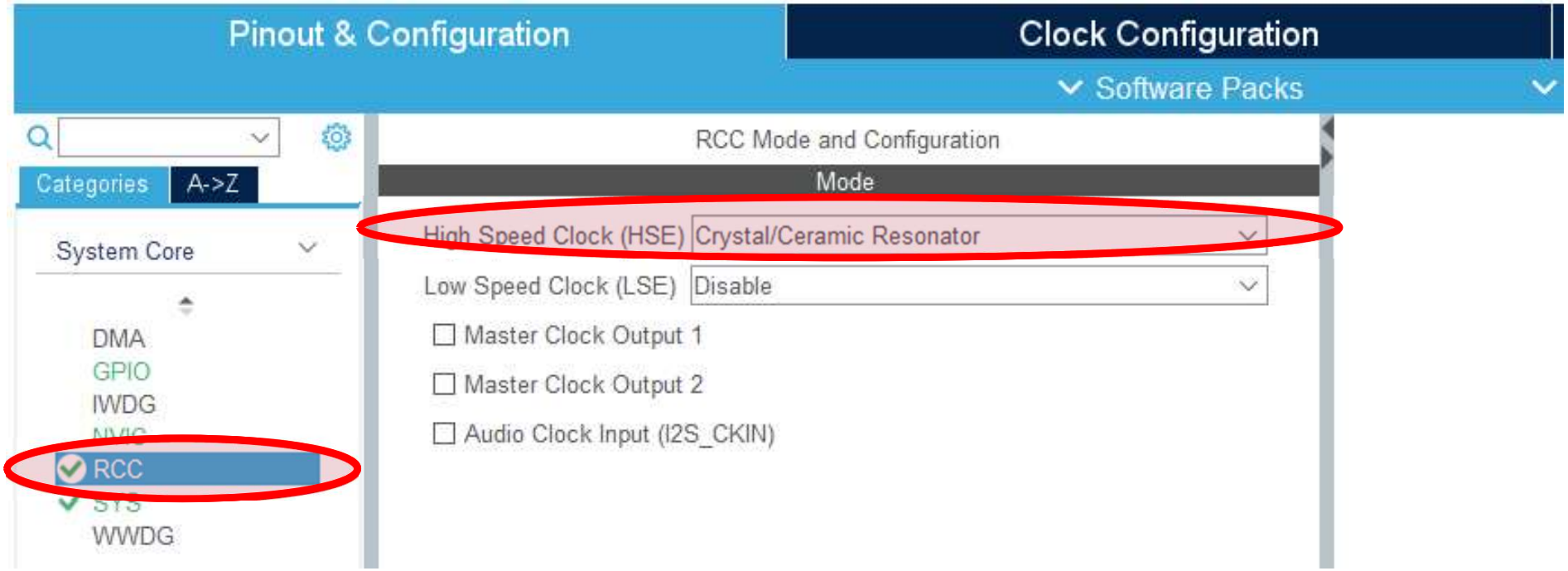
Cube IDE beállítás

- Middleware beállítása, USB HID



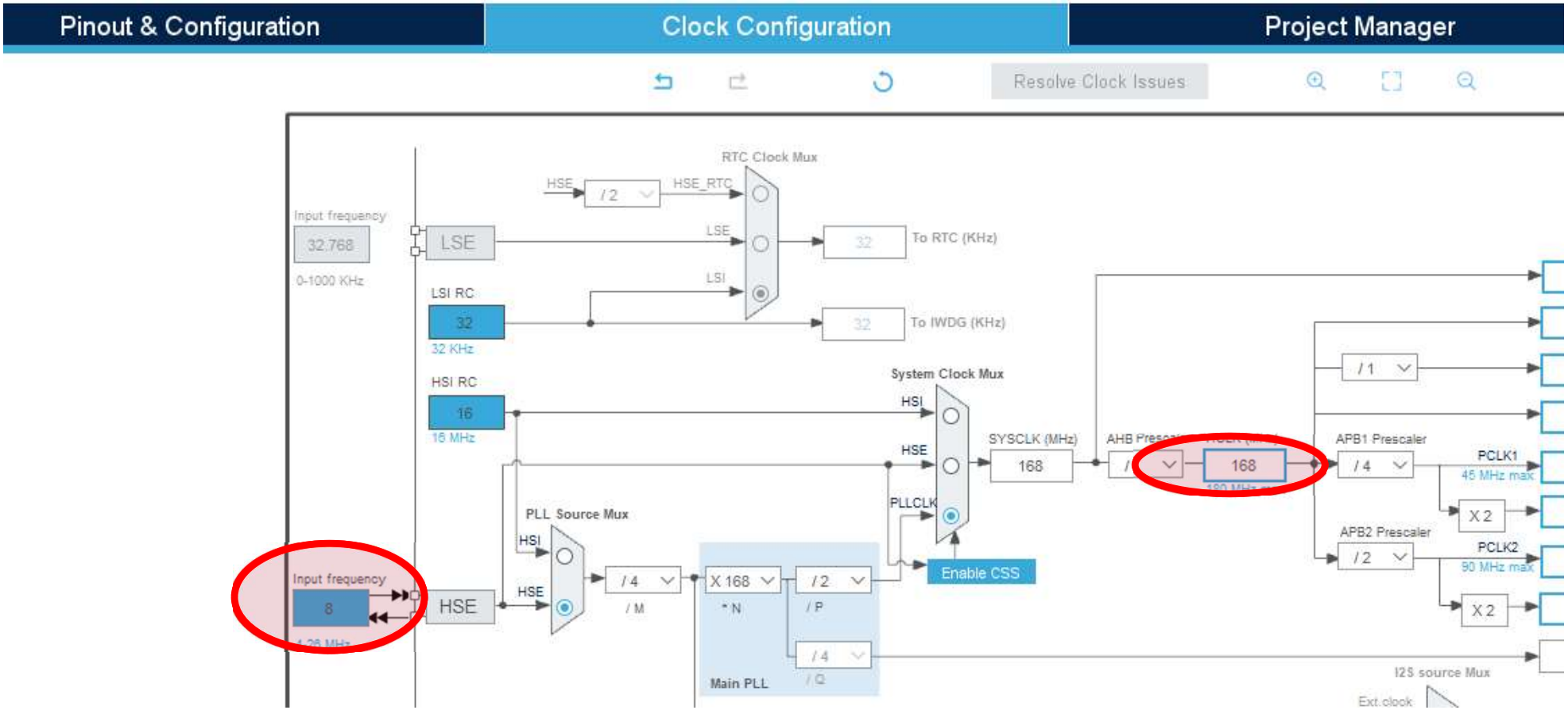
Cube IDE beállítás

- Külső 8 MHz-es kristály beállítása:



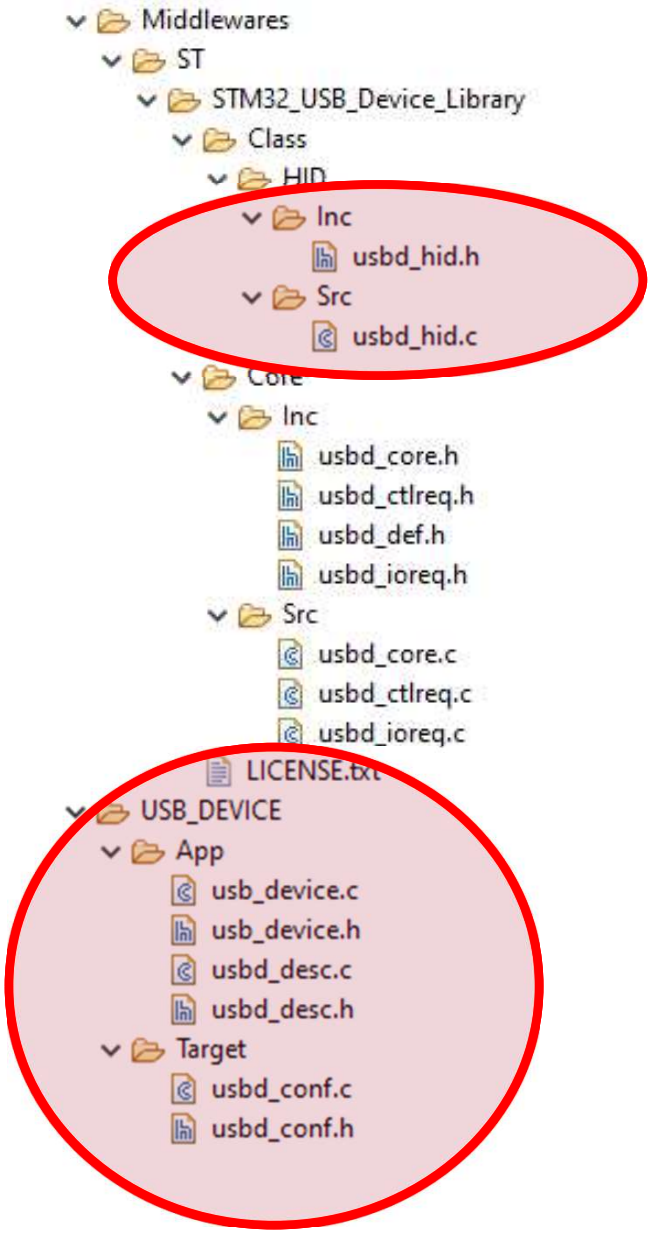
Cube IDE beállítás

- Órajel hálózat a külső 8MHz-es kvarcról.
- 168 MHz CPU frekvencia beállítása



Cube IDE generálódott file-ok

- Külön a Class és az alap device információk

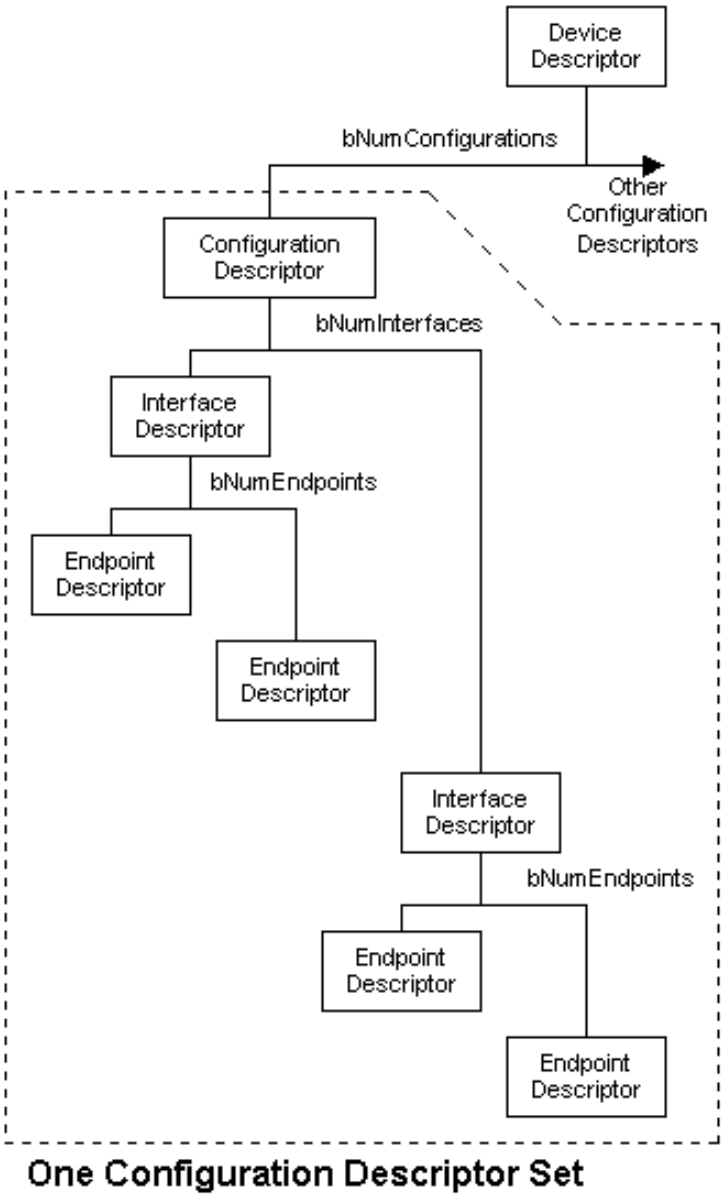


Cube IDE generálódott file-ok

- Inicializálás
  - Device callback struktúra
  - Class callback struktúra

```
void MX_USB_DEVICE_Init(void)
{
  /* USER CODE BEGIN USB_DEVICE_Init_PreTreatment */

  /* USER CODE END USB_DEVICE_Init_PreTreatment */

  /* Init Device Library, add supported class and start the library. */
  if (USBD_Init(&hUsbDeviceHS, &HS_Desc, DEVICE_HS) != USBD_OK)
  {
    Error_Handler();
  }
  if (USBD_RegisterClass(&hUsbDeviceHS, &USBD_HID) != USBD_OK)
  {
    Error_Handler();
  }
  if (USBD_Start(&hUsbDeviceHS) != USBD_OK)
  {
    Error_Handler();
  }
```

Device leírások



One Configuration Descriptor Set

```
USBD_DescriptorsTypeDef HS_Desc =
{
  USBD_HS_DeviceDescriptor
, USBD_HS_LangIDStrDescriptor
, USBD_HS_ManufacturerStrDescriptor
, USBD_HS_ProductStrDescriptor
, USBD_HS_SerialStrDescriptor
, USBD_HS_ConfigStrDescriptor
, USBD_HS_InterfaceStrDescriptor
#if (USBD_LPM_ENABLED == 1)
, USBD_HS_USR_BOSDescriptor
#endif /* (USBD_LPM_ENABLED == 1) */
};
```

Device descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of the Descriptor in Bytes (18)
1	bDescriptorType	1	Constant	Device Descriptor (0x01)
2	bcdUSB	2	BCD	USB Specification Number which device complies too.
4	bDeviceClass	1	Class	Class Code (by USB Org)
				If equal to Zero, each interface specifies it's own class code
				If equal to 0xFF, the class code is vendor specified.
				Otherwise field is valid Class Code.
5	bDeviceSubClass	1	SubClass	Subclass Code (by USB Org)
6	bDeviceProtocol	1	Protocol	Protocol Code (by USB Org)
7	bMaxPacketSize	1	Number	Maximum Packet Size for Zero Endpoint. Valid Sizes are 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID (by USB Org)
10	idProduct	2	ID	Product ID (by Manufacturer)
12	bcdDevice	2	BCD	Device Release Number
14	iManufacturer	1	Index	Index of Manufacturer String Descriptor
15	iProduct	1	Index	Index of Product String Descriptor
16	iSerialNumber	1	Index	Index of Serial Number String Descriptor
17	bNumConfigurations	1	Integer	Number of Possible Configurations

```
__ALIGN_BEGIN uint8_t USBD_HS_DeviceDesc[USB_LEN_DEV_DESC] __ALIGN_END =
{
  0x12,                       /*bLength */
  USB_DESC_TYPE_DEVICE,       /*bDescriptorType*/
#if (USBD_LPM_ENABLED == 1)
  0x01,                       /*bcdUSB */ /* changed to USB version 2.01
                                           in order to support LPM L1 suspend
                                           resume test of USBCV3.0*/
#else
  0x00,                       /*bcdUSB */
#endif /* (USBD_LPM_ENABLED == 1) */

  0x02,
  0x00,                       /*bDeviceClass*/
  0x00,                       /*bDeviceSubClass*/
  0x00,                       /*bDeviceProtocol*/
  USB_MAX_EP0_SIZE,           /*bMaxPacketSize*/
  LOBYTE(USBD_VID),           /*idVendor*/
  HIBYTE(USBD_VID),           /*idVendor*/
  LOBYTE(USBD_PID_HS),        /*idProduct*/
  HIBYTE(USBD_PID_HS),        /*idProduct*/
  0x00,                       /*bcdDevice rel. 2.00*/
  0x02,
  USBD_IDX_MFC_STR,           /*Index of manufacturer  string*/
  USBD_IDX_PRODUCT_STR,       /*Index of product string*/
  USBD_IDX_SERIAL_STR,        /*Index of serial number string*/
  USBD_MAX_NUM_CONFIGURATION  /*bNumConfigurations*/
};
```

Device class codes

Base Class	Descriptor Usage	Description
00h	Device	Use class information in the Interface Descriptors
01h	Interface	Audio
02h	Both	Communications and CDC Control
03h	Interface	HID (Human Interface Device)
05h	Interface	Physical
06h	Interface	Image
07h	Interface	Printer
08h	Interface	Mass Storage
09h	Device	Hub
0Ah	Interface	CDC-Data
0Bh	Interface	Smart Card
0Dh	Interface	Content Security
0Eh	Interface	Video
0Fh	Interface	Personal Healthcare
10h	Interface	Audio/Video Devices
DCh	Both	Diagnostic Device
E0h	Interface	Wireless Controller
EFh	Both	Miscellaneous
FEh	Interface	Application Specific
FFh	Both	Vendor Specific

Class callback függvények és leírók

```
USBD_ClassTypeDef USBD_HID =
{
  USBD_HID_Init,
  USBD_HID_DeInit,
  USBD_HID_Setup,
  NULL,                 /* EP0_TxSent */
  NULL,                 /* EP0_RxReady */
  USBD_HID_DataIn,      /* DataIn */
  NULL,                 /* DataOut */
  NULL,                 /* SOF */
  NULL,
  NULL,
#ifdef USE_USBD_COMPOSITE
  NULL,
  NULL,
  NULL,
  NULL,
#else
  USBD_HID_GetHSCfgDesc,
  USBD_HID_GetFSCfgDesc,
  USBD_HID_GetOtherSpeedCfgDesc,
  USBD_HID_GetDeviceQualifierDesc,
#endif /* USE_USBD_COMPOSITE  */
};
```

Callback függvények

Leíró file-ok

Configuration descriptor

Field	Value	Meaning
bLength	9	Valid length
bDescriptorType	2	CONFIGURATION
wTotalLength	34	Total combined size of this set of descriptors
bNumInterfaces	1	Number of interfaces supported by this configuration
bConfigurationValue	1	Value to use as an argument to the SetConfiguration() request to select this configuration
iConfiguration	0	Index of string descriptor describing this configuration
bmAttributes (Self-Powered)	0	Bus-Powered
bmAttributes (Remote Wakeup)	1	Yes
bmAttributes (Other bits)	0x80	Valid
bMaxPower	100 mA	Maximum Current Drawn by Device in This Configuration

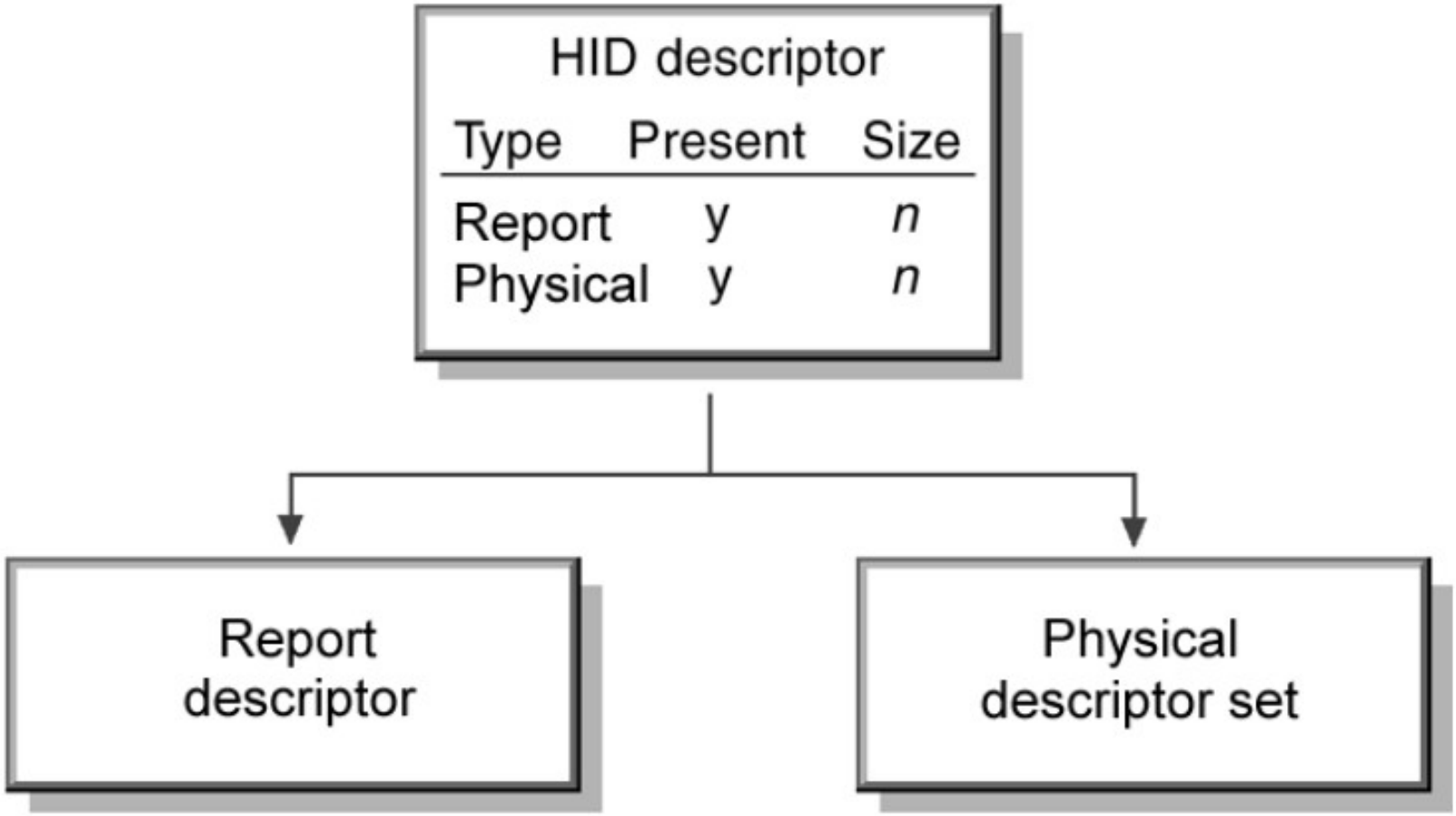
```
  ALIGN_BEGIN static uint8_t USBD_HID_CfgDesc[
{
  0x09,
  USB_DESC_TYPE_CONFIGURATION,
  USB_HID_CONFIG_DESC_SIZ,
  0x00,
  0x01,
  0x01,
  0x00,

#if (USBD_SELF_POWERED == 1U)
  0xE0,
#else
  0xA0,
#endif /* USBD_SELF_POWERED */
  USBD_MAX_POWER,
```

Interface descriptor

Field	Value	Meaning
bLength	9	Valid length
bDescriptorType	4	INTERFACE
bInterfaceNumber	0	Zero-based Number of this Interface.
bAlternateSetting	0	Value used to select this alternative setting for the interface identified in the prior field
bNumEndpoints	1	Number of endpoints used by this interface (excluding endpoint zero).
bInterfaceClass	0x03	HID
bInterfaceSubClass	0x01	Boot Interface
bInterfaceProtocol	0x02	Mouse
iInterface	0	Index of string descriptor describing this Interface

```
  0x09,
  USB_DESC_TYPE_INTERFACE,
  0x00,
  0x00,
  0x01,
  0x03,
  0x01,
  0x02,
  0,
```

HID descriptor



HID descriptor

Part	Offset/Size (Bytes)	Description
bLength	0/1	Numeric expression that is the total size of the HID descriptor.
bDescriptorType	1/1	Constant name specifying type of HID descriptor.
bcdHID	2/2	Numeric expression identifying the HID Class Specification release.
bCountryCode	4/1	Numeric expression identifying country code of the localized hardware.
bNumDescriptors	5/1	Numeric expression specifying the number of class descriptors (always at least one i.e. Report descriptor.)
bDescriptorType	6/1	Constant name identifying type of class descriptor. See Section 7.1.2: Set_Descriptor Request for a table of class descriptor constants.
wDescriptorLength	7/2	Numeric expression that is the total size of the Report descriptor.
[*bDescriptorType*]...	9/1	Constant name specifying type of optional descriptor.
[*wDescriptorLength*]...	10/2	Numeric expression that is the total size of the optional descriptor.

```
0x09,
HID_DESCRIPTOR_TYPE,
0x11,
0x01,
0x00,
0x01,
0x22,
HID_MOUSE_REPORT_DESC_SIZE,
0x00,
```

HID report descriptor

```
__ALIGN_BEGIN static uint8_t HID_MOUSE_ReportDesc[HID_MOUSE_REPORT_DESC_SIZE] __ALIGN_END =
{
  0x05, 0x01,         /* Usage Page (Generic Desktop Ctrls)      */
  0x09, 0x02,         /* Usage (Mouse)                           */
  0xA1, 0x01,         /* Collection (Application)                */
  0x09, 0x01,         /*   Usage (Pointer)                       */
  0xA1, 0x00,         /*   Collection (Physical)                 */
  0x05, 0x09,         /*     Usage Page (Button)                 */
  0x19, 0x01,         /*     Usage Minimum (0x01)                */
  0x29, 0x03,         /*     Usage Maximum (0x03)                */
  0x15, 0x00,         /*     Logical Minimum (0)                 */
  0x25, 0x01,         /*     Logical Maximum (1)                 */
  0x95, 0x03,         /*     Report Count (3)                    */
  0x75, 0x01,         /*     Report Size (1)                     */
  0x81, 0x02,         /*     Input (Data,Var,Abs)                */
  0x95, 0x01,         /*     Report Count (1)                    */
  0x75, 0x05,         /*     Report Size (5)                     */
  0x81, 0x01,         /*     Input (Const,Array,Abs)             */
  0x05, 0x01,         /*     Usage Page (Generic Desktop Ctrls)  */
  0x09, 0x30,         /*     Usage (X)                           */
  0x09, 0x31,         /*     Usage (Y)                           */
  0x09, 0x38,         /*     Usage (Wheel)                       */
  0x15, 0x81,         /*     Logical Minimum (-127)              */
  0x25, 0x7F,         /*     Logical Maximum (127)               */
  0x75, 0x08,         /*     Report Size (8)                     */
  0x95, 0x03,         /*     Report Count (3)                    */
  0x81, 0x06,         /*     Input (Data,Var,Rel)                */
  0xC0,               /*   End Collection                        */
  0x09, 0x3C,         /*   Usage (Motion Wakeup)                 */
  0x05, 0xFF,         /*   Usage Page (Reserved 0xFF)            */
  0x09, 0x01,         /*   Usage (0x01)                          */
  0x15, 0x00,         /*   Logical Minimum (0)                   */
  0x25, 0x01,         /*   Logical Maximum (1)                   */
  0x75, 0x01,         /*   Report Size (1)                       */
  0x95, 0x02,         /*   Report Count (2)                      */
  0xB1, 0x22,         /*   Feature (Data,Var,Abs,NoWrp)          */
  0x75, 0x06,         /*   Report Size (6)                       */
  0x95, 0x01,         /*   Report Count (1)                      */
  0xB1, 0x01,         /*   Feature (Const,Array,Abs,NoWrp)       */
  0xC0                /* End Collection                          */
};
```

HID report descriptor: boot mouse

```
Usage Page (Generic Desktop),
Usage (Mouse),
Collection (Application),
  Usage (Pointer),
  Collection (Physical),
    Report Count (3),
    Report Size (1),
    Usage Page (Buttons),
    Usage Minimum (1),
    Usage Maximum (3),
    Logical Minimum (0),
    Logical Maximum (1),
    Input (Data, Variable, Absolute),
    Report Count (1),
    Report Size (5),
    Input (Constant),
    Report Size (8),
    Report Count (2),
    Usage Page (Generic Desktop),
    Usage (X),
    Usage (Y),
    Logical Minimum (-127),
    Logical Maximum (127),
    Input (Data, Variable, Relative),
  End Collection,
End Collection
```

Byte	Bits	Description
0	0	Button 1
	1	Button 2
	2	Button 3
	4 to 7	Device-specific
1	0 to 7	X displacement
2	0 to 7	Y displacement
3 to n	0 to 7	Device specific (optional)

Endpoint descriptor

Field	Value	Meaning
bLength	7	Valid length
bDescriptorType	5	ENDPOINT
bEndpointAddress	0x81	Endpoint 1 - IN
bmAttributes	0x03	Interrupt. Data Endpoint.
wMaxPacketSize	0x0004	Maximum Packet Size is 4
bInterval	0x0A	10 Frames (10 ms)

```
  /******************** Descriptor of Mouse endpoint ********************/
  /* 27 */
  0x07,                                                    /* bLength: Endpoint Descriptor size */
  USB_DESC_TYPE_ENDPOINT,                                  /* bDescriptorType:*/

  HID_EPIN_ADDR,                                           /* bEndpointAddress: Endpoint Address (IN) */
  0x03,                                                    /* bmAttributes: Interrupt endpoint */
  HID_EPIN_SIZE,                                           /* wMaxPacketSize: 4 Bytes max */
  0x00,
  HID_FS_BINTERVAL,                                        /* bInterval: Polling Interval */
  /* 34 */
```

HID kód kiegészítése a report küldésével

- Az egér kurzor lassú mozgatása a jobb alsó sarok irányába

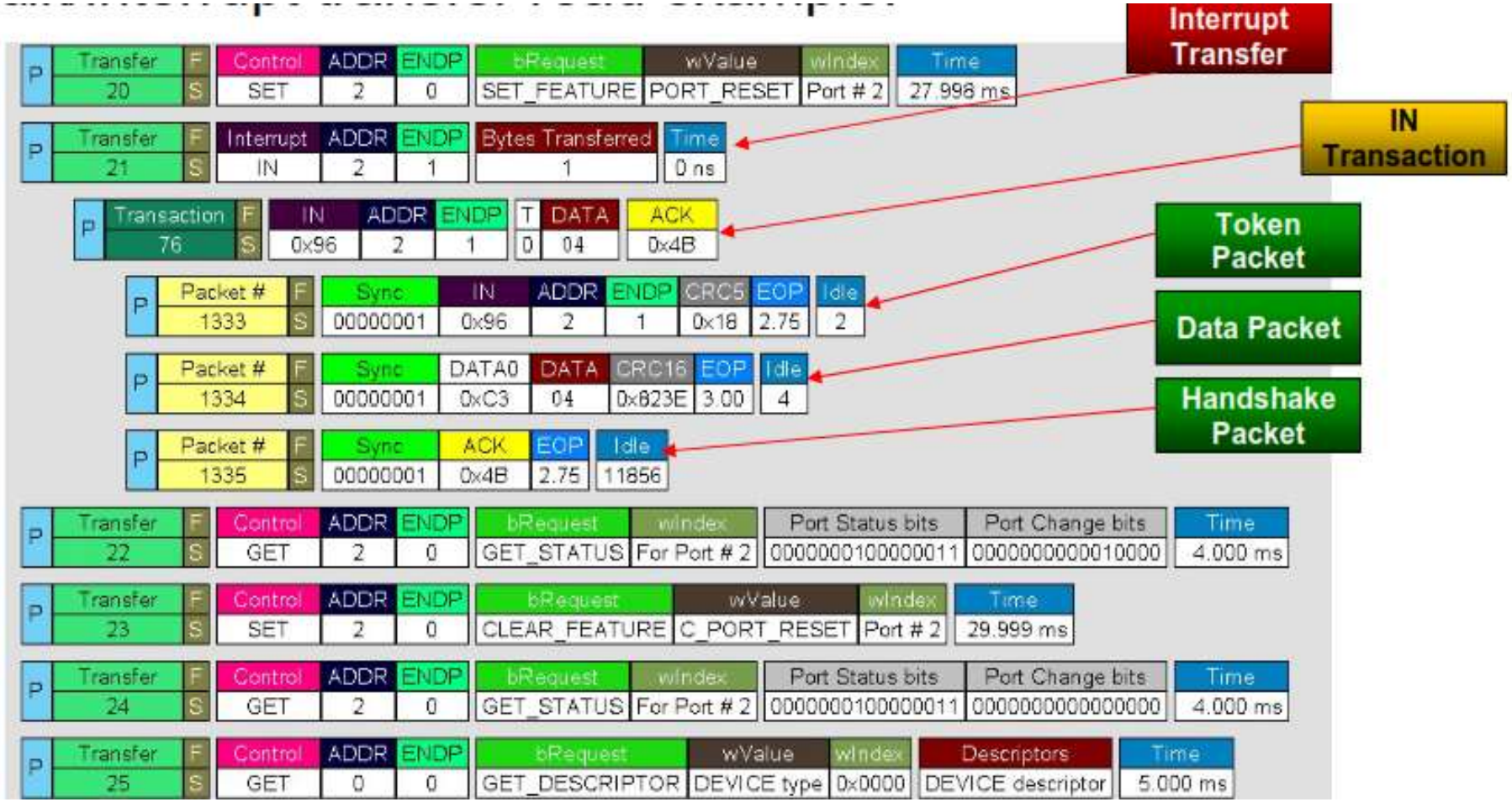
```
/* USER CODE BEGIN 0 */
uint8_t report[4];
/* USER CODE END 0 */


/* USER CODE BEGIN WHILE */
while (1)
{
    report[0] = 0;
    report[1] = 1;
    report[2] = 1;
    report[3] = 0;
    USBD_HID_SendReport(&hUsbDeviceHS, report, 4);
    HAL_Delay(100);

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Felvétel WireShark-al

- Indulás folyamata

151	5.165043	host	2.2.0	USB	36 GET DESCRIPTOR Request DEVICE
152	5.165283	2.2.0	host	USB	46 GET DESCRIPTOR Response DEVICE
153	5.165308	host	2.2.0	USB	36 GET DESCRIPTOR Request CONFIGURATION
154	5.165523	2.2.0	host	USB	37 GET DESCRIPTOR Response CONFIGURATION
155	5.165546	host	2.2.0	USB	36 GET DESCRIPTOR Request CONFIGURATION
156	5.165774	2.2.0	host	USB	62 GET DESCRIPTOR Response CONFIGURATION
157	5.165816	host	2.2.0	USB	36 SET CONFIGURATION Request
158	5.166073	2.2.0	host	USB	28 SET CONFIGURATION Response
159	5.166098	host	2.2.0	USBHID	36 SET_IDLE Request
160	5.166273	2.2.0	host	USBHID	28 SET_IDLE Response
161	5.166529	host	2.2.0	USBHID	36 GET DESCRIPTOR Request HID Report
162	5.166882	2.2.0	host	USBHID	102 GET DESCRIPTOR Response HID Report

Tényleges Interupt transzfer



Felvétel WireShark-al

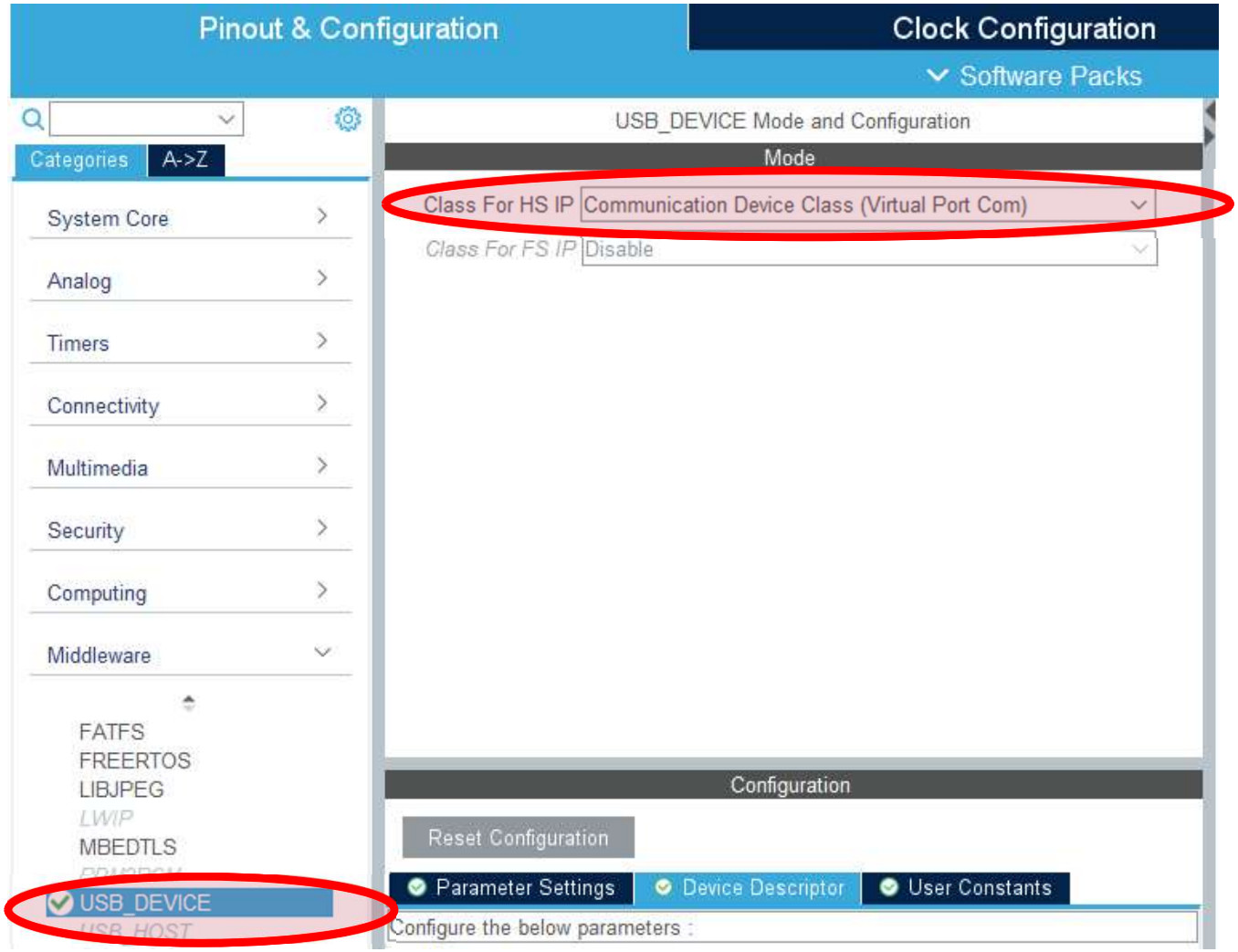
- Vannak speciálisan USB analizátorok azok általában picit szebb eredményt adnak.

```
> Frame 99: 31 bytes on wire (248 bits), 31 bytes captured (248 bits) on interface \\.\USBPcap2, id 0
v USB URB
    [Source: 2.2.1]
    [Destination: host]
    USBPcap pseudoheader length: 27
    IRP ID: 0xffffa00f12bb3010
    IRP USBD_STATUS: USBD_STATUS_SUCCESS (0x00000000)
    URB Function: URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009)
  > IRP information: 0x01, Direction: PDO -> FDO
    URB bus id: 2
    Device address: 2
  > Endpoint: 0x81, Direction: IN
    URB transfer type: URB_INTERRUPT (0x01)
    Packet Data Length: 4
    [Request in: 96]
    [Time from request: 0.199905000 seconds]
    [bInterfaceClass: HID (0x03)]
HID Data: 00010100
```

CDC VCP, virtuális soros port

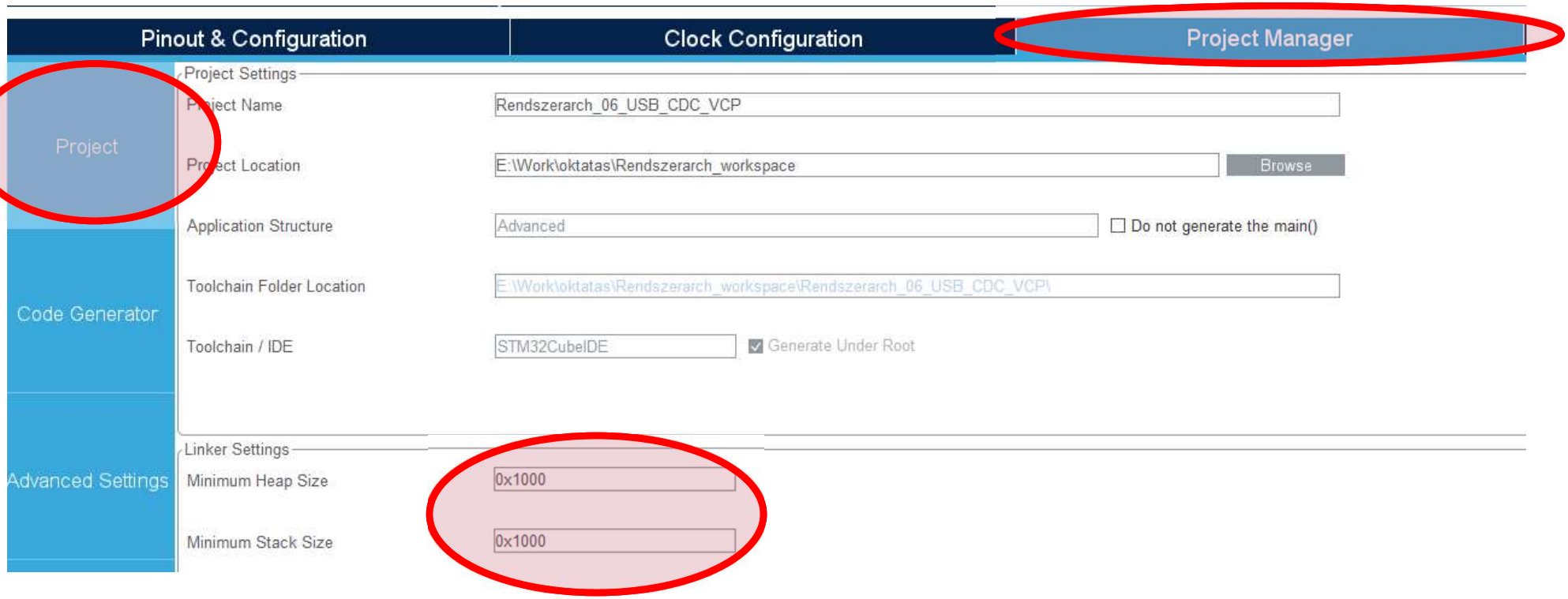
Cube IDE beállítás

- Minden ugyan az, kivéve a Class megadását, meg a heap/stack méret megnövelését



Cube IDE beállítás

- Stack, heap méret növelés



Device descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of the Descriptor in Bytes (18)
1	bDescriptorType	1	Constant	Device Descriptor (0x01)
2	bcdUSB	2	BCD	USB Specification Number which device complies too.
4	bDeviceClass	1	Class	Class Code (by USB Org)
				If equal to Zero, each interface specifies it's own class code
				If equal to 0xFF, the class code is vendor specified.
				Otherwise field is valid Class Code.
5	bDeviceSubClass	1	SubClass	Subclass Code (by USB Org)
6	bDeviceProtocol	1	Protocol	Protocol Code (by USB Org)
7	bMaxPacketSize	1	Number	Maximum Packet Size for Zero Endpoint. Valid Sizes are 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID (by USB Org)
10	idProduct	2	ID	Product ID (by Manufacturer)
12	bcdDevice	2	BCD	Device Release Number
14	iManufacturer	1	Index	Index of Manufacturer String Descriptor
15	iProduct	1	Index	Index of Product String Descriptor
16	iSerialNumber	1	Index	Index of Serial Number String Descriptor
17	bNumConfigurations	1	Integer	Number of Possible Configurations

```
/** USB standard device descriptor. */
__ALIGN_BEGIN uint8_t USBD_HS_DeviceDesc[USB_LEN_DEV_DESC] __ALIGN_END =
{
  0x12,                       /*bLength */
  USB_DESC_TYPE_DEVICE,       /*bDescriptorType*/
#if (USBD_LPM_ENABLED == 1)
  0x01,                       /*bcdUSB */ /* changed to USB version 2.01
                                             in order to support LPM L1 suspend
                                             resume test of USBCV3.0*/
#else
  0x00,                       /*bcdUSB */
#endif /* (USBD_LPM_ENABLED == 1) */

  0x02,
  0x02,                       /*bDeviceClass*/
  0x02,                       /*bDeviceSubClass*/
  0x00,                       /*bDeviceProtocol*/
  USB_MAX_EP0_SIZE,           /*bMaxPacketSize*/
  LOBYTE(USBD_VID),           /*idVendor*/
  HIBYTE(USBD_VID),           /*idVendor*/
  LOBYTE(USBD_PID_HS),        /*idProduct*/
  HIBYTE(USBD_PID_HS),        /*idProduct*/
  0x00,                       /*bcdDevice rel. 2.00*/
  0x02,
  USBD_IDX_MFC_STR,           /*Index of manufacturer  string*/
  USBD_IDX_PRODUCT_STR,       /*Index of product string*/
  USBD_IDX_SERIAL_STR,        /*Index of serial number string*/
  USBD_MAX_NUM_CONFIGURATION  /*bNumConfigurations*/
};
```

Device class codes

Base Class	Descriptor Usage	Description
00h	Device	Use class information in the Interface Descriptors
01h	Interface	Audio
02h	Both	Communications and CDC Control
03h	Interface	HID (Human Interface Device)
05h	Interface	Physical
06h	Interface	Image
07h	Interface	Printer
08h	Interface	Mass Storage
09h	Device	Hub
0Ah	Interface	CDC-Data
0Bh	Interface	Smart Card
0Dh	Interface	Content Security
0Eh	Interface	Video
0Fh	Interface	Personal Healthcare
10h	Interface	Audio/Video Devices
DCh	Both	Diagnostic Device
E0h	Interface	Wireless Controller
EFh	Both	Miscellaneous
FEh	Interface	Application Specific
FFh	Both	Vendor Specific

Configuration descriptor

```
__ALIGN_BEGIN static uint8_t USBD_CDC_CfgDesc[USB_CDC_CONFIG_DESC_SIZ] __ALIGN_END =
{
  /* Configuration Descriptor */
  0x09,                                 /* bLength: Configuration Descriptor size */
  USB_DESC_TYPE_CONFIGURATION,          /* bDescriptorType: Configuration */
  USB_CDC_CONFIG_DESC_SIZ,              /* wTotalLength */
  0x00,
  0x02,                                 /* bNumInterfaces: 2 interfaces */
  0x01,                                 /* bConfigurationValue: Configuration value */
  0x00,                                 /* iConfiguration: Index of string descriptor
                                           describing the configuration */
#if (USBD_SELF_POWERED == 1U)
  0xC0,                                 /* bmAttributes: Bus Powered according to user configuration */
#else
  0x80,                                 /* bmAttributes: Bus Powered according to user configuration */
#endif /* USBD_SELF_POWERED */
  USBD_MAX_POWER,                       /* MaxPower (mA) */
```

Interface descriptor

- Egy CDC eszköznek jellemzően két interfésze van egy **Communication interface**-e, amin keresztül a management dolgok bonyolíthatóak és egy **Data interface**-e, amin keresztül az adatot továbbítja

```
/* Interface Descriptor */
0x09,                                   /* bLength: Interface Descriptor size */
USB_DESC_TYPE_INTERFACE,                /* bDescriptorType: Interface */
/* Interface descriptor type */
0x00,                                   /* bInterfaceNumber: Number of Interface */
0x00,                                   /* bAlternateSetting: Alternate setting */
0x01,                                   /* bNumEndpoints: One endpoint used */
0x02,                                   /* bInterfaceClass: Communication Interface Class */
0x02,                                   /* bInterfaceSubClass: Abstract Control Model */
0x01,                                   /* bInterfaceProtocol: Common AT commands */
0x00,                                   /* iInterface */
```

Functional descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this descriptor.
1	*bDescriptorType*	1	Constant	CS_INTERFACE, as defined in Table 12.
2	*bDescriptorSubtype*	1	Constant	Identifier (ID) of functional descriptor. For a list of the supported values, see Table 13.
3	(function specific data0)	1	Misc.	First function specific data byte. These fields will vary depending on the functional descriptor being represented.
…	…	…	…	…
N+2	(functional specific data N-1)	1	Misc.	Nth function specific data byte. These fields will vary depending on the functional descriptor being represented.

```
/* Header Functional Descriptor */
0x05,                   /* bLength: Endpoint Descriptor size */
0x24,                   /* bDescriptorType: CS_INTERFACE */
0x00,                   /* bDescriptorSubtype: Header Func Desc */
0x10,                   /* bcdCDC: spec release number */
0x01,
```

Call management

Offset	Field	Size	Value	Description
3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports: D7..D2: RESERVED (Reset to zero) D1: 0 - Device sends/receives call management information only over the Communications Class interface. 1 - Device can send/receive call management information over a Data Class interface. D0: 0 - Device does not handle call management itself. 1 - Device handles call management itself. The previous bits, in combination, identify which call management scenario is used. If bit D0 is reset to 0, then the value of bit D1 is ignored. In this case, bit D1 is reset to zero for future compatibility.
4	*bDataInterface*	1	Number	Interface number of Data Class interface optionally used for call management. *

* Zero based index of the interface in this configuration (*bInterfaceNum*)

```
/* Call Management Functional Descriptor */
0x05,                                   /* bFunctionLength */
0x24,                                   /* bDescriptorType: CS_INTERFACE */
0x01,                                   /* bDescriptorSubtype: Call Management Func Desc */
0x00,                                   /* bmCapabilities: D0+D1 */
0x01,                                   /* bDataInterface */
```

Functional descriptor, ACM functional

3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports. (A bit value of zero means that the request is not supported.) D7..D4: RESERVED (Reset to zero) D3: 1 - Device supports the notification Network_Connection. D2: 1 - Device supports the request Send_Break D1: 1 - Device supports the request combination of Set_Line_Coding, Set_Control_Line_State, Get_Line_Coding, and the notification Serial_State. D0: 1 - Device supports the request combination of Set_Comm_Feature, Clear_Comm_Feature, and Get_Comm_Feature. The previous bits, in combination, identify which requests/notifications are supported by a CommunicationsClass interface with the SubClass code of Abstract Control Model.

```
/* ACM Functional Descriptor */
0x04,                                   /* bFunctionLength */
0x24,                                   /* bDescriptorType: CS_INTERFACE */
0x02,                                   /* bDescriptorSubtype: Abstract Control Management desc */
0x02,                                   /* bmCapabilities */
```

Union functional descriptor, interface endpoint

```
/* Union Functional Descriptor */
0x05,                                   /* bFunctionLength */
0x24,                                   /* bDescriptorType: CS_INTERFACE */
0x06,                                   /* bDescriptorSubtype: Union func desc */
0x00,                                   /* bMasterInterface: Communication class interface */
0x01,                                   /* bSlaveInterface0: Data Class Interface */

/* Endpoint 2 Descriptor */
0x07,                                   /* bLength: Endpoint Descriptor size */
USB_DESC_TYPE_ENDPOINT,                 /* bDescriptorType: Endpoint */
CDC_CMD_EP,                             /* bEndpointAddress */
0x03,                                   /* bmAttributes: Interrupt */
LOBYTE(CDC_CMD_PACKET_SIZE),            /* wMaxPacketSize */
HIBYTE(CDC_CMD_PACKET_SIZE),
CDC_FS_BINTERVAL,                       /* bInterval */
/*---------------------------------------------------------------------------*/
```

Data Interface

```
  /* Data class interface descriptor */
  0x09,                                       /* bLength: Endpoint Descriptor size */
  USB_DESC_TYPE_INTERFACE,                    /* bDescriptorType: */
  0x01,                                       /* bInterfaceNumber: Number of Interface */
  0x00,                                       /* bAlternateSetting: Alternate setting */
  0x02,                                       /* bNumEndpoints: Two endpoints used */
  0x0A,                                       /* bInterfaceClass: CDC */
  0x00,                                       /* bInterfaceSubClass */
  0x00,                                       /* bInterfaceProtocol */
  0x00,                                       /* iInterface */

  /* Endpoint OUT Descriptor */
  0x07,                                       /* bLength: Endpoint Descriptor size */
  USB_DESC_TYPE_ENDPOINT,                     /* bDescriptorType: Endpoint */
  CDC_OUT_EP,                                 /* bEndpointAddress */
  0x02,                                       /* bmAttributes: Bulk */
  LOBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),        /* wMaxPacketSize */
  HIBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),
  0x00,                                       /* bInterval */

  /* Endpoint IN Descriptor */
  0x07,                                       /* bLength: Endpoint Descriptor size */
  USB_DESC_TYPE_ENDPOINT,                     /* bDescriptorType: Endpoint */
  CDC_IN_EP,                                  /* bEndpointAddress */
  0x02,                                       /* bmAttributes: Bulk */
  LOBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),        /* wMaxPacketSize */
  HIBYTE(CDC_DATA_FS_MAX_PACKET_SIZE),
  0x00                                        /* bInterval */
};
#endif /* USE_USBD_COMPOSITE  */
```

Kód kiegészítés

- Class specifikus setup kérések kiegészítése
- usbd_cdc_if.c

```
/*******************************************************************************/
/* Line Coding Structure                                                       */
/*-----------------------------------------------------------------------------*/
/* Offset | Field       | Size | Value  | Description                         */
/* 0      | dwDTERate   |   4  | Number |Data terminal rate, in bits per second*/
/* 4      | bCharFormat |   1  | Number | Stop bits                           */
/*                                        0 - 1 Stop bit                       */
/*                                        1 - 1.5 Stop bits                    */
/*                                        2 - 2 Stop bits                      */
/* 5      | bParityType |  1   | Number | Parity                              */
/*                                        0 - None                             */
/*                                        1 - Odd                              */
/*                                        2 - Even                             */
/*                                        3 - Mark                             */
/*                                        4 - Space                            */
/* 6      | bDataBits  |   1   | Number Data bits (5, 6, 7, 8 or 16).          */
/*******************************************************************************/
case CDC_SET_LINE_CODING:

    lineCoding[0] = pbuf[0];
    lineCoding[1] = pbuf[1];
    lineCoding[2] = pbuf[2];
    lineCoding[3] = pbuf[3];
    lineCoding[4] = pbuf[4];
    lineCoding[5] = pbuf[5];
    lineCoding[6] = pbuf[6];
  break;

case CDC_GET_LINE_CODING:
    pbuf[0] = lineCoding[0] ;
    pbuf[1] = lineCoding[1] ;
    pbuf[2] = lineCoding[2] ;
    pbuf[3] = lineCoding[3] ;
    pbuf[4] = lineCoding[4] ;
    pbuf[5] = lineCoding[5] ;
    pbuf[6] = lineCoding[6] ;

  break;
```

Funkció

- Mivel a Line coding-ot ráhagytuk a Host-ra, így mindegy, hogy milyen baudrate-et állítunk be

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    char data[32];
    sprintf(data,"Hello USB\r\n");
    CDC_Transmit_HS(data, strlen(data));

    HAL_Delay(1000);

  /* USER CODE END WHILE */
```

Mérés

16	1.551466	host	2.2.1	USB	27 URB_BULK in
17	2.552393	2.2.1	host	USB	38 URB_BULK in
18	2.552525	host	2.2.1	USB	27 URB_BULK in
19	3.553396	2.2.1	host	USB	38 URB_BULK in
20	3.553530	host	2.2.1	USB	27 URB_BULK in

```
> Frame 17: 38 bytes on wire (304 bits), 38 bytes captured (304 bits) on interface \\.\USBPcap2, id 0
v USB URB
    [Source: 2.2.1]
    [Destination: host]
    USBPcap pseudoheader length: 27
    IRP ID: 0xffffa00f1651d6a0
    IRP USBD_STATUS: USBD_STATUS_SUCCESS (0x00000000)
    URB Function: URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009)
  > IRP information: 0x01, Direction: PDO -> FDO
    URB bus id: 2
    Device address: 2
  > Endpoint: 0x81, Direction: IN
    URB transfer type: URB_BULK (0x03)
    Packet Data Length: 11
    [Request in: 16]
    [Time from request: 1.000927000 seconds]
    [bInterfaceClass: CDC-Data (0x0a)]
  Leftover Capture Data: 48656c6c6f205553420d0a
```

```
0000  1b 00 a0 d6 51 16 0f a0  ff ff 00 00 00 00 09 00   ····Q··· ········
0010  01 02 00 02 00 81 03 0b  00 00 00 48 65 6c 6c 6f   ········ ···Hello
0020  20 55 53 42 0d 0a                                  USB··
```