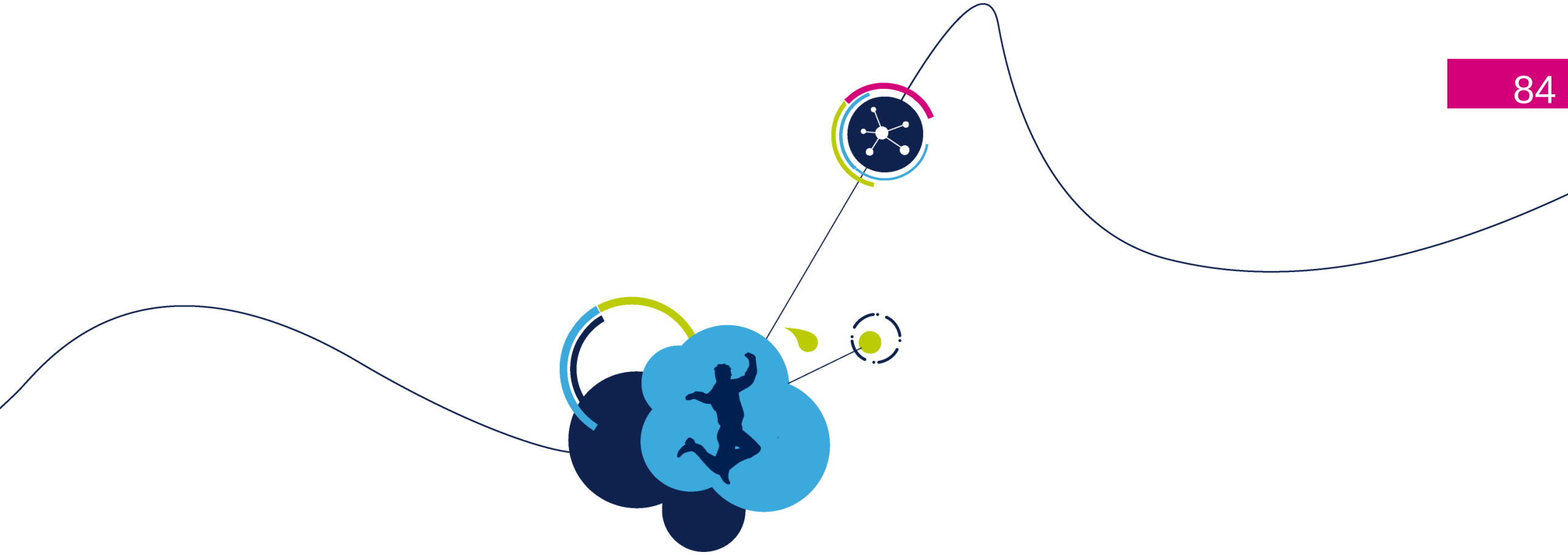


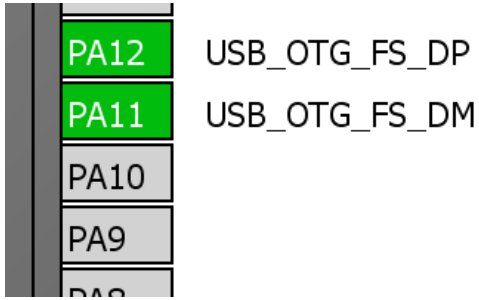
USB Custom HID Device lab

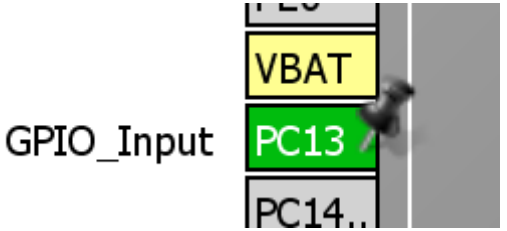
USB Custom HID Device lab

- Use interrupt transfers for data transfer
- Limited throughput compared to CDC bulk (64 kB/s)
- Fixed packet size
- No need for driver on Microsoft Windows
- Following example show you how to create USB Custom HID device, which is echoing the communication from host side

USB Custom HID Device lab

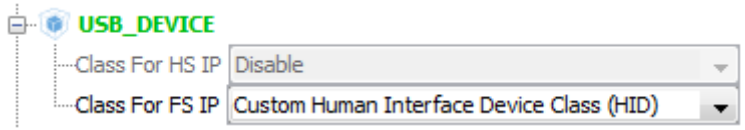
- Create project in CubeMX, configuration is the same like for HID device
  - Menu > File > New Project
  - Select STM32F4 > STM32F446 > LQFP144 > STM32F446ZETx
- Select USB FS OTG in device mode
- Select HSE clock
  - (Bypass HSE from STlink)
- Configure PC13 as input – key button





USB Custom HID Device lab

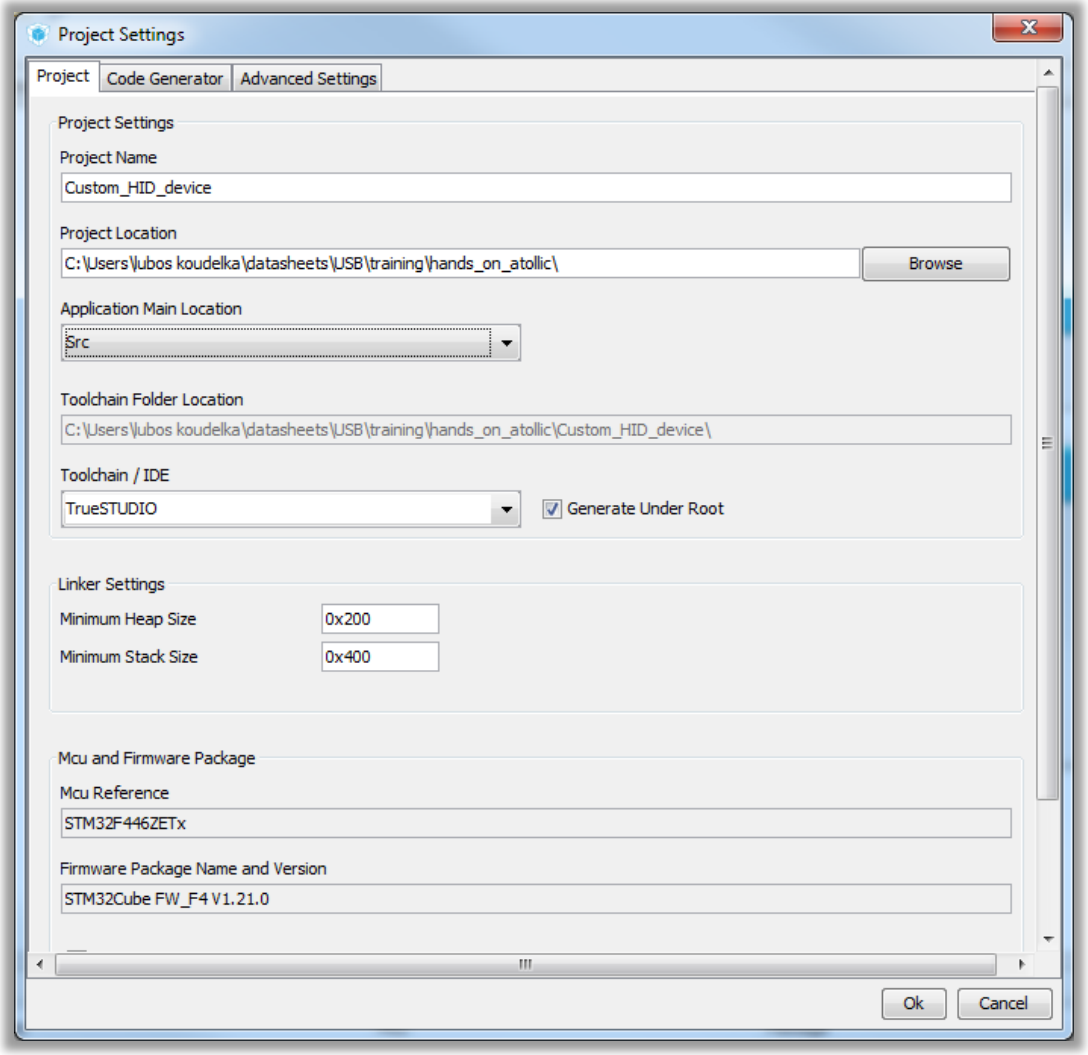
- Select Custom HID class in MiddleWares



- Configure RCC clocks
  - Set 8 MHz HSE as PLL input
  - Set HCLK frequency 168 MHz
  - PLL parameters will be computed automatically

USB Custom HID Device lab

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Linker Settings
  - For custom HID class default Heap size is sufficient
- Now we can Generate Code
  - Menu > Project > Generate Code



life.augmented

USB Custom HID Device lab

- Increase HID report size in usbd_conf.h

```
#define USBD_CUSTOM_HID_REPORT_DESC_SIZE     33
```

- In usbd_customhid.h increase endpoint size and change USB customid structure (default structure message size is limited to 2)

```
#define CUSTOM_HID_EPIN_SIZE                 0x40
#define CUSTOM_HID_EPOUT_SIZE                0x40


typedef struct _USBD_CUSTOM_HID_Itf
{
  uint8_t                   *pReport;
  int8_t (* Init)           (void);
  int8_t (* DeInit)         (void);
  int8_t (* OutEvent)       (uint8_t* );

}USBD_CUSTOM_HID_ItfTypeDef;
```

USB Custom HID Device lab

- In file usbd_customhid.c is optional change of bInterval value to get faster response from the device

```
/******************** Descriptor of Custom HID endpoints ********************/
 /* 27 */
 0x07,          /*bLength: Endpoint Descriptor size*/
 USB_DESC_TYPE_ENDPOINT, /*bDescriptorType:*/

 CUSTOM_HID_EPIN_ADDR,     /*bEndpointAddress: Endpoint Address (IN)*/
 0x03,          /*bmAttributes: Interrupt endpoint*/
 CUSTOM_HID_EPIN_SIZE, /*wMaxPacketSize*/
 0x00,
 0xa,          /*bInterval: Polling Interval (10 ms)*/
 /* 34 */

 0x07,         /* bLength: Endpoint Descriptor size */
 USB_DESC_TYPE_ENDPOINT,/* bDescriptorType: */
 CUSTOM_HID_EPOUT_ADDR,  /*bEndpointAddress: Endpoint Address (OUT)*/
 0x03,/* bmAttributes: Interrupt endpoint */
 CUSTOM_HID_EPOUT_SIZE,/* wMaxPacketSize*/
 0x00,
 0xa,/* bInterval: Polling Interval (10 ms) */
 /* 41 */
```

USB Custom HID Device lab

- In file usbd_customhid.c change call of OUT events

```
 static uint8_t  USBD_CUSTOM_HID_DataOut (USBD_HandleTypeDef *pdev,
                                uint8_t epnum)
{

  USBD_CUSTOM_HID_HandleTypeDef     *hhid = (USBD_CUSTOM_HID_HandleTypeDef*)pdev->pClassData;

  ((USBD_CUSTOM_HID_ItfTypeDef *)pdev->pUserData)->OutEvent(hhid->Report_buf);

  USBD_LL_PrepareReceive(pdev, CUSTOM_HID_EPOUT_ADDR , hhid->Report_buf,
                         USBD_CUSTOMHID_OUTREPORT_BUF_SIZE);

  return USBD_OK;
}

uint8_t USBD_CUSTOM_HID_EP0_RxReady(USBD_HandleTypeDef *pdev)
{
  USBD_CUSTOM_HID_HandleTypeDef     *hhid = (USBD_CUSTOM_HID_HandleTypeDef*)pdev->pClassData;

  if (hhid->IsReportAvailable == 1)
  {
  ((USBD_CUSTOM_HID_ItfTypeDef *)pdev->pUserData)->OutEvent(hhid->Report_buf);
    hhid->IsReportAvailable = 0;
  }

  return USBD_OK;
}
```

USB Custom HID Device lab

- In file usbd_custom_hid_if.c add buffer for user USB message

```
/* USER CODE BEGIN PRIVATE_DEFINES */
uint8_t buffer[0x40];
/* USER CODE END PRIVATE_DEFINES */
```

- And modify CUSTOM_HID_OutEvent_FS function declaration and definition

```
static int8_t CUSTOM_HID_OutEvent_FS (uint8_t* state);


static int8_t CUSTOM_HID_OutEvent_FS  (uint8_t* state)
{
  /* USER CODE BEGIN 6 */
  memcpy(buffer,state,0x40);
  USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS,(uint8_t*)buffer,0x40);
  return (0);
  /* USER CODE END 6 */
}
```

USB Custom HID Device lab

- And in file usbd_custom_hid_if.c add HID report descriptor

```
__ALIGN_BEGIN static uint8_t CUSTOM_HID_ReportDesc_FS[USBD_CUSTOM_HID_REPORT_DESC_SIZE] __ALIGN_END =

{
  /* USER CODE BEGIN 0 */
    0x06, 0x00, 0xff,              //Usage Page(Undefined )
    0x09, 0x01,                   // USAGE (Undefined)
    0xa1, 0x01,                   // COLLECTION (Application)
    0x15, 0x00,                   //   LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,             //   LOGICAL_MAXIMUM (255)
    0x75, 0x08,                   //   REPORT_SIZE (8)
    0x95, 0x40,                   //   REPORT_COUNT (64)
    0x09, 0x01,                   //   USAGE (Undefined)
    0x81, 0x02,                   //   INPUT (Data,Var,Abs)
    0x95, 0x40,                   //   REPORT_COUNT (64)
    0x09, 0x01,                   //   USAGE (Undefined)
    0x91, 0x02,                   //   OUTPUT (Data,Var,Abs)
    0x95, 0x01,                   //   REPORT_COUNT (1)
    0x09, 0x01,                   //   USAGE (Undefined)
    0xb1, 0x02,                   //   FEATURE (Data,Var,Abs)
  /* USER CODE END 0 */
  0xC0    /*     END_COLLECTION             */

};
```

life.augmented

USB Custom HID Device lab

- Now is the device ready for test
- For communication on host side you can use attached HID terminal (C#)

HID_terminal.zip