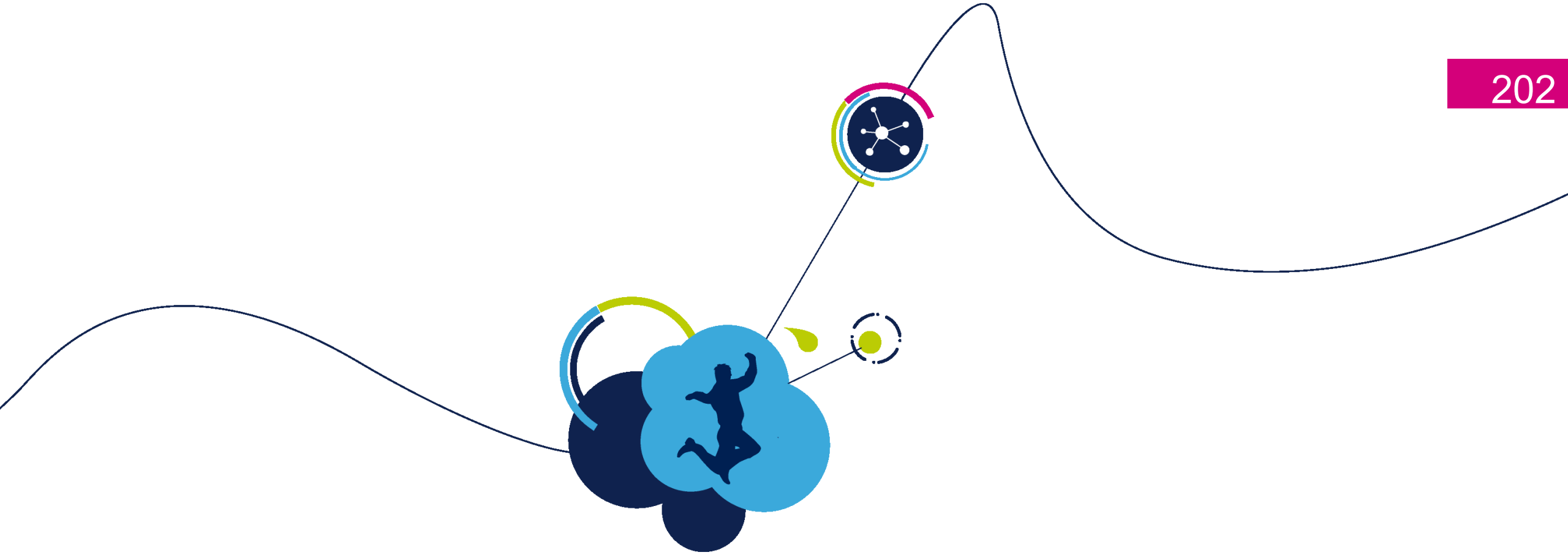


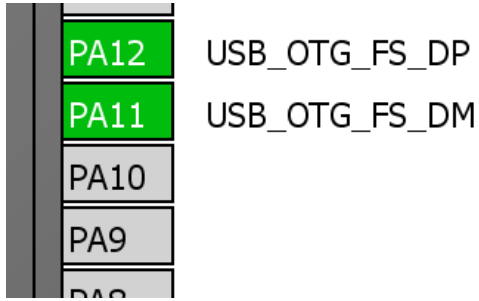
USB HID Host

USB HID Host

- Cube HAL USB host library is capable to communicate with mouse and keyboards
- HID report descriptor parsing is done on host side
- In Cube HAL libraries HID report descriptor parsing is skipped
  - only basic configuration supported, complex parsing demands more complicated parser (flash memory consumption)

USB HID Host

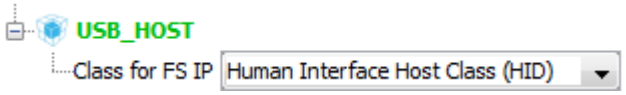
- Create project in CubeMX
  - Menu > File > New Project
  - Select STM32F4 > STM32F446 > LQFP144 > STM32F446ZETx
- Select USB FS OTG in host mode
- Select HSE clock
  - (Bypass HSE from STlink)



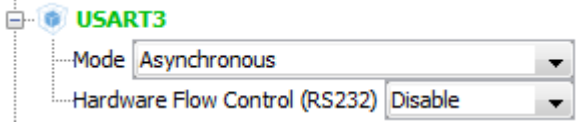
- Configure GPIOs connected to LEDs as GPIO output – PB0, PB7 and PB14

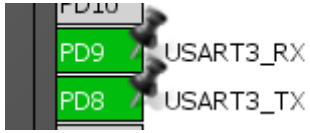
USB HID Host

- Select Human Interface host class in MiddleWares



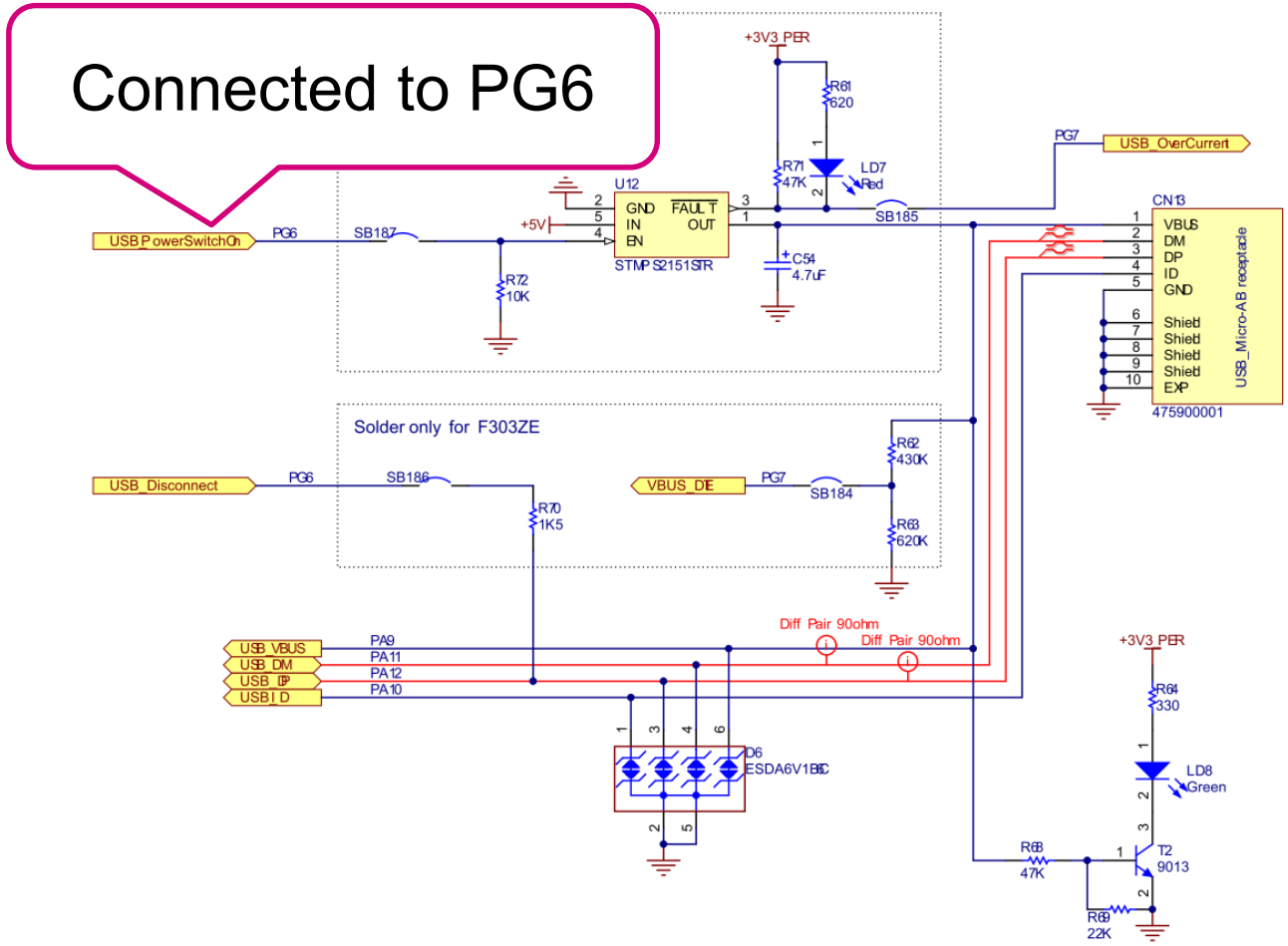
- Configure RCC clocks
  - Set 8 MHz HSE as PLL input and HCLK frequency 168 MHz
- Add USART3 for debug purposes
  - USART3 is connected to STlink virtual COM port functionality
  - PD9 – USART3_RX
  - PD8 – USART3_TX
  - For easier handling more convenient DMA implementation is not used





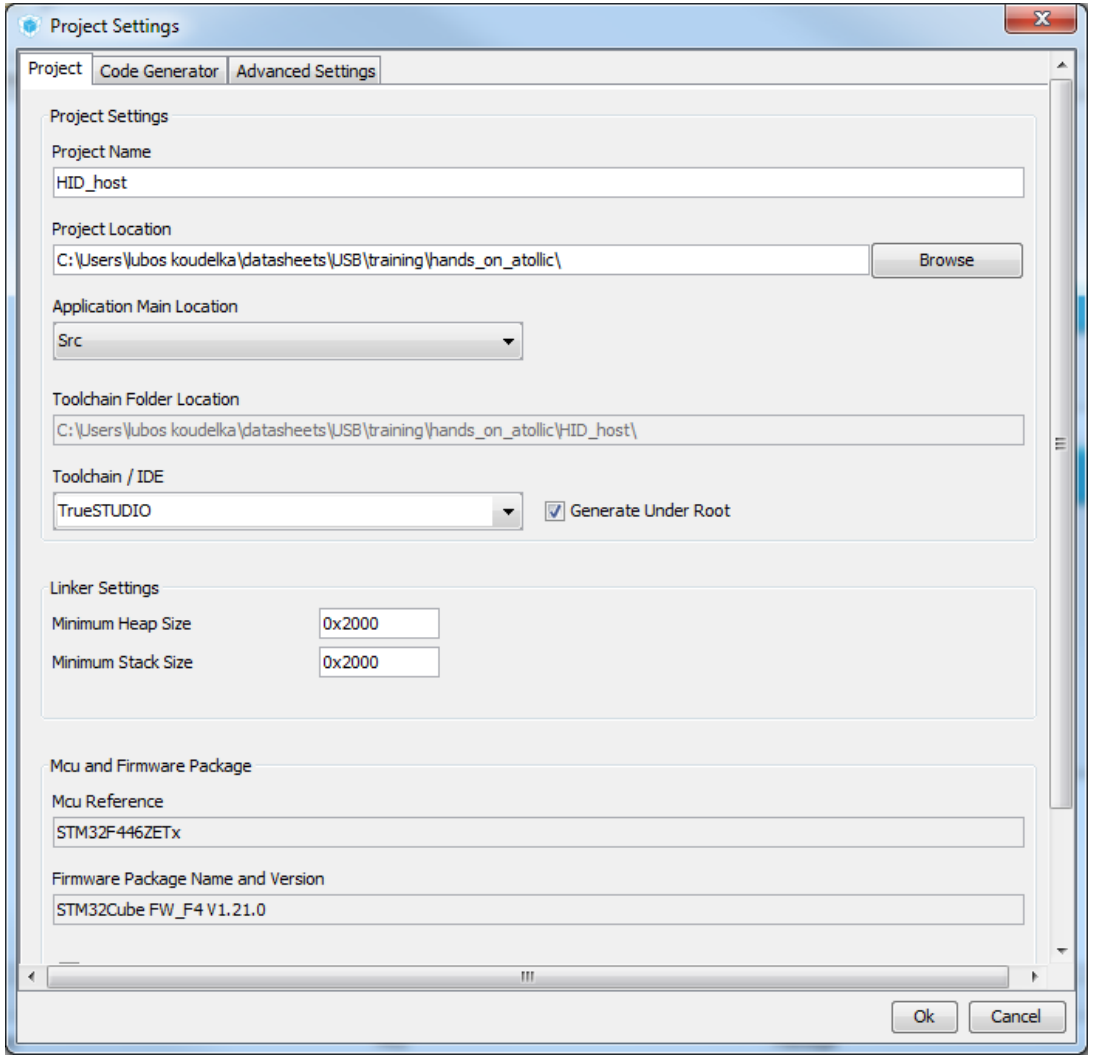
USB HID Host

- HOST must also power the device -> we need to enable voltage regulator connected to VBUS line
- Set PG6 as GPIO output



USB HID Host

- Now we set the project details for generation
  - Menu > Project > Project Settings
  - Set the project name
  - Project location
  - Type of toolchain
- Linker Settings
  - Increase Heap size to 0x2000
  - Increase Stack size to 0x2000
- Now we can Generate Code
  - Menu > Project > Generate Code



USB HID Host

- In usbh_conf.c is function for handling USB VBUS voltage level - USBH_LL_DriverVBUS
- Pin PG6 controls power source for USB VBUS

```
USBH_StatusTypeDef  USBH_LL_DriverVBUS
(USBH_HandleTypeDef *phost, uint8_t state)
{
  /* USER CODE BEGIN 0 */
  /* USER CODE END 0*/
  if (phost->id == HOST_FS)
  {
    if (state == 0)
    {
      /* Deactivate Charge pump */
     HAL_GPIO_WritePin(GPIOG,GPIO_PIN_6,GPIO_PIN_RESET);
      /* USER CODE END DRIVE_HIGH_CHARGE_FOR_FS */
    }
    else
    {
      /* Activate Charge pump */
      HAL_GPIO_WritePin(GPIOG,GPIO_PIN_6,GPIO_PIN_SET);
      /* USER CODE END DRIVE_LOW_CHARGE_FOR_FS */
    }
  }
  HAL_Delay(200);
  return USBH_OK;
}
```

USB HID Host

- To main.c add functionality print out messages from HID device

```
#include "usbh_hid.h"
static int32_t uart_length=0;
uint8_t uart_tx_buffer[100];
extern HID_MOUSE_Info_TypeDef    mouse_info;
```

- FIFO for messages from HID device is already implemented in the library

- Add the functions for HID reports decoding into USBH_HID_EventCallback, which occurs once HID report is received

USB HID Host


```
/* USER CODE BEGIN 4 */
void USBH_HID_EventCallback(USBH_HandleTypeDef *phost)
{
HID_KEYBD_Info_TypeDef *keybd_info;
uint8_t keycode;
  HID_HandleTypeDef *HID_Handle =  (HID_HandleTypeDef *) phost->pActiveClass->pData;
  if(HID_Handle->Init ==  USBH_HID_KeybdInit){
  keybd_info =  USBH_HID_GetKeybdInfo(phost);
  keycode = USBH_HID_GetASCIICode(keybd_info);
  uart_length=sprintf(uart_tx_buffer, "Key pressed: 0x%x\n",keycode);
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);

  }else if(HID_Handle->Init ==  USBH_HID_MouseInit){
  USBH_HID_GetMouseInfo(phost);
  uart_length=sprintf(uart_tx_buffer, "Mouse action: x=  0x%x, y=  0x%x, button1 =
0x%x, button2 = 0x%x, button3 = 0x%x \n",mouse_info.x, mouse_info.y,
mouse_info.buttons[0], mouse_info.buttons[1], mouse_info.buttons[2]);
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
  }

}
/* USER CODE END 4 */
```

USB HID Host

- Default HID host library demands also descriptors, which are not mandatory and not supported by some devices
- Result of this descriptors can be ignored for simplification if not needed by application

```
 case HID_REQ_SET_PROTOCOL:
    /* set protocol */
    if (USBH_HID_SetProtocol (phost, 0) != USBH_BUSY)
    {
      HID_Handle->ctl_state = HID_IDLE;
      /* all requests performed*/
      phost->pUser(phost, HOST_USER_CLASS_ACTIVE);
      status = USBH_OK;
    }
break;
```

USB HID Host

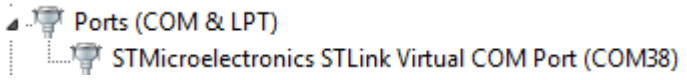
- Default HID host library demands also descriptors, which are not mandatory and not supported by some devices
- Result of this descriptors can be ignored for simplification if not needed by application

```
case HID_IDLE:
 if( USBH_HID_GetReport (phost,
                                0x01,
                                 0,
                                 HID_Handle->pData,
                                 HID_Handle->length)!=USBH_OK){


        fifo_write(&HID_Handle->fifo, HID_Handle->pData, HID_Handle->length);
        HID_Handle->state = HID_SYNC;
 }
break;
```

USB HID Host

- Then in device manager find COM port number of connected host board Stlink



- Debug output with more instructions can be displayed in any COM port terminal application