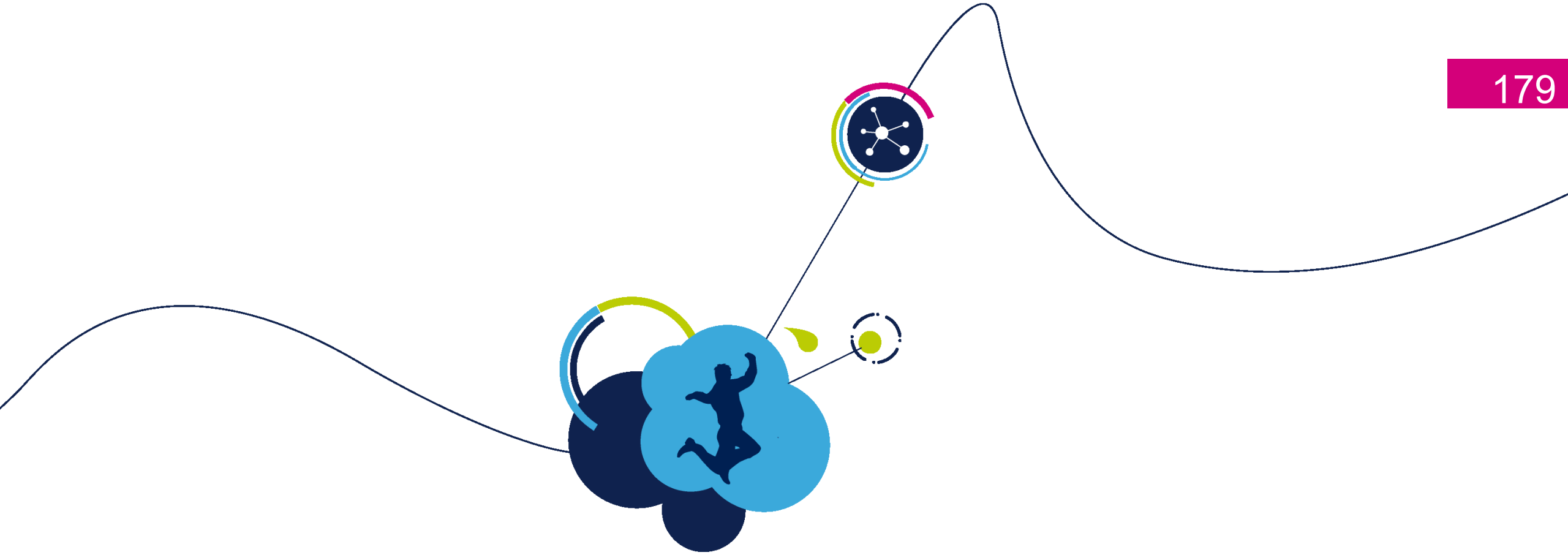


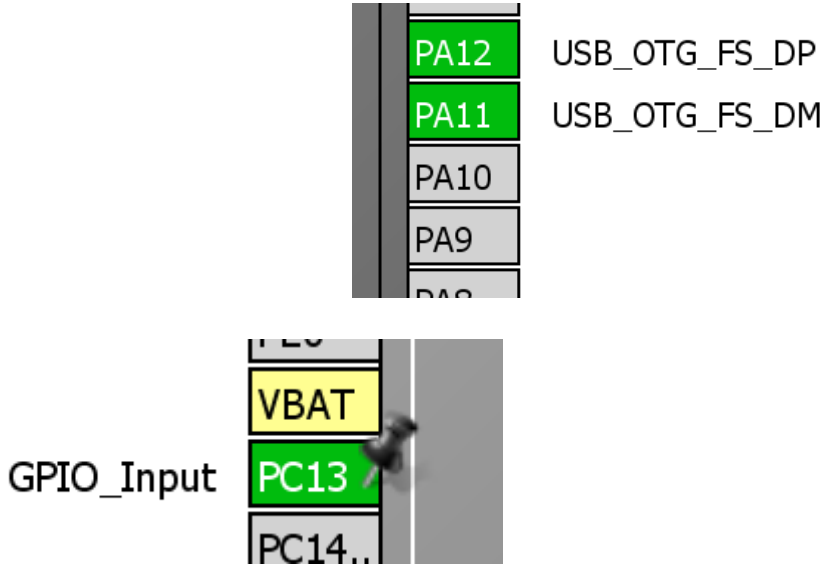
USB MSC DFU Host

USB MSC DFU Host

- In this example will be demonstrated FW upgrade from connected MSC device
- Previous USB MSC host lab reused in this example, part of the code is taken from USB DFU device
- Only binary file format for FW update supported in this example other format support may be added

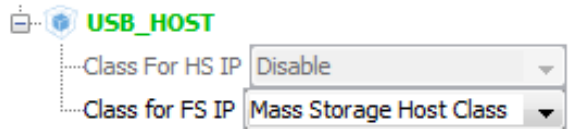
USB MSC DFU Host

- 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 PC13 as input – key button
- Configure GPIOs connected to LEDs as GPIO output – PB0, PB7 and PB14

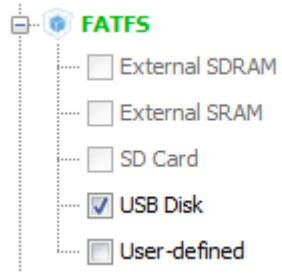


USB MSC DFU Host

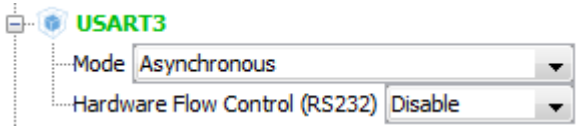
- Select Communication host class in MiddleWares

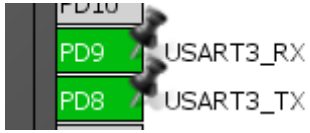


- Configure FAT files system on USB disk



- 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

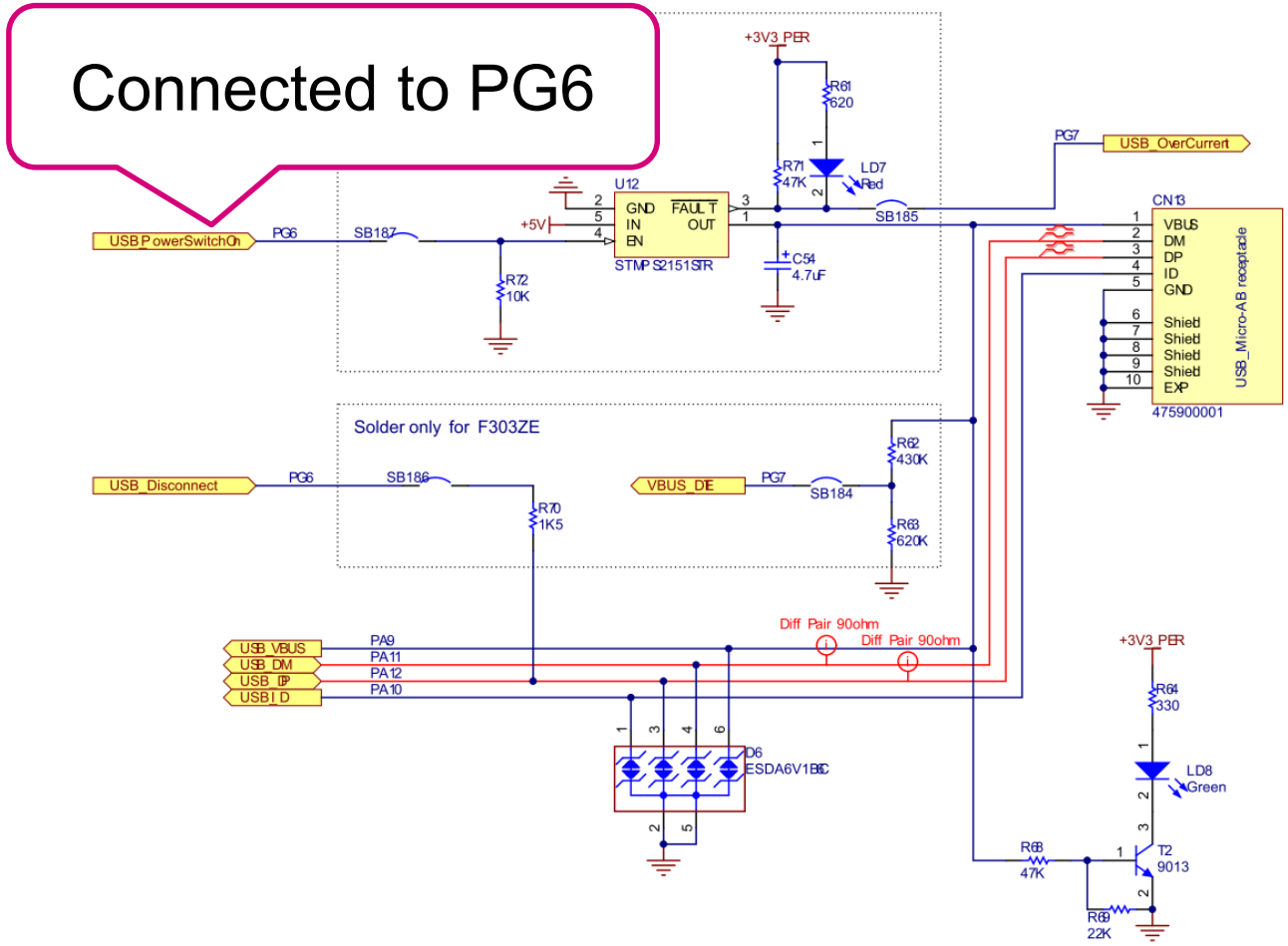




- For easier handling more convenient DMA implementation is not used

USB MSC DFU Host

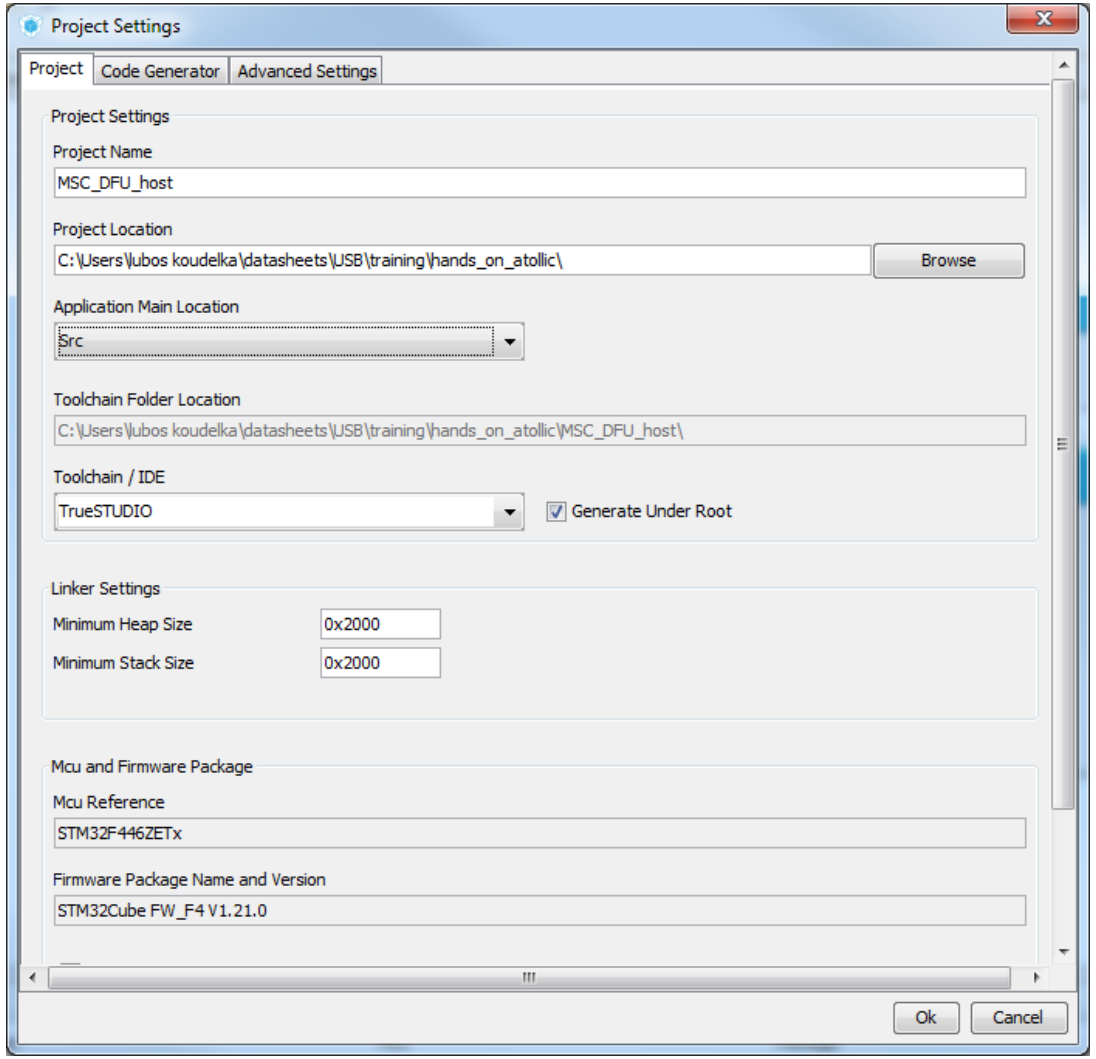
- HOST must also power the device -> we need to enable voltage regulator connected to VBUS line
- Set PG6 as GPIO output



life.augmented

USB MSC DFU 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 MSC DFU Host

- Work in similar way like MSC host lab – once MCS device is connected and enumerated, binary file from root of connected device with name "image.BIN" is written to flash memory area
- USBH_UserProcess callback store state of connected device into Appli_state variable
- If the Device is connected and enumerated, into Appli_state is stored APPLICATION_READY and we can commutate with device
  - Than on button press firmware is updated

USB MSC DFU 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 MSC Host

- If the Device is connected and enumerated, into Appli_state is stored APPLICATION_READY state and we can commutate with device
  - Storage mount and debug UART message print is added in usb_host.c

```
static void USBH_UserProcess(USBH_HandleTypeDef *phost, uint8_t id) {
/* USER CODE BEGIN 1 */
switch (id) {
case HOST_USER_SELECT_CONFIGURATION:
break;

case HOST_USER_DISCONNECTION:
Appli_state = APPLICATION_DISCONNECT;
break;

case HOST_USER_CLASS_ACTIVE:
Appli_state = APPLICATION_READY;
uart_length=sprintf(uart_tx_buffer, "application ready \n");
HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
if(f_mount(&USBH_fatfs, USBHPath, 0) != FR_OK)
    {
  uart_length=sprintf(uart_tx_buffer, "f_mount fail \n");
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
    }
break;

case HOST_USER_CONNECTION:
Appli_state = APPLICATION_START;
break;
default:
break;
}
/* USER CODE END 1 */
}
```

Device can communicate

Device not connected

life.augmented

USB MSC DFU Host

- Add files flash_if.c and flash_if.h for operation with internal flash memory

flash_if.c

flash_if.h

- In flash_if.h start address for user application is defined
  - Binary application to be flashed need to start on this address

```
#define APPLICATION_ADDRESS        (uint32_t)0x08008000
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
/* USER CODE BEGIN 0 */
#include "ff.h"
static void COMMAND_ProgramFlashMemory(void);
FATFS USBH_fatfs;
FIL MyFile;
FRESULT res;
uint32_t bytesWritten;
uint8_t rtext[200];
uint8_t wtext[] = "USB Host Library : Mass Storage Example";
uint8_t name[10];//name of the file
uint16_t counter=0;
uint32_t i=0;
static int32_t uart_length=0;
extern char USBHPath [];  /* USBH logical drive path */
extern UART_HandleTypeDef huart3;
uint8_t uart_tx_buffer[100];
#define DOWNLOAD_FILENAME          "0:image.BIN"
FIL MyFileR;                    /* File object for download operation */
FILINFO MyFileInfo;             /* File object information */
#define BUFFER_SIZE        ((uint16_t)512*64)
static uint32_t TmpReadSize = 0x00;
static uint32_t RamAddress = 0x00;
static __IO uint32_t LastPGAddress = APPLICATION_ADDRESS;
static uint8_t RAM_Buf[BUFFER_SIZE] = { 0x00 };
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
void userFunction(void) {
if (Appli_state == APPLICATION_READY) {
uart_length=sprintf(uart_tx_buffer, "Press and release user button to start FW update \n");
HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
while(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13)!=GPIO_PIN_SET){}
while(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13)==GPIO_PIN_SET){}
  /* Open the binary file to be downloaded */
  if (f_open(&MyFileR, DOWNLOAD_FILENAME, FA_READ) != FR_OK)
  {
    /*read size of binary file*/
    f_stat(DOWNLOAD_FILENAME,&MyFileInfo);
    /* The binary file is not available: Turn LED1, LED2 and LED4 On and Toggle
     * LED3 in infinite loop */
  uart_length=sprintf(uart_tx_buffer, "The binary file is not available \n");
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
  while(1){}
  }
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
 if (MyFileInfo.fsize > USER_FLASH_SIZE)
  {
    /* No available Flash memory size for the binary file: Turn LED4 On and
     * Toggle LED3 in infinite loop */
   uart_length=sprintf(uart_tx_buffer, "No available Flash memory size for the
binary file \n");
   HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
   while(1){}
   }

   /* Download On Going: Turn LED4 On */
   uart_length=sprintf(uart_tx_buffer, "Download On Going \n");
   HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
   FLASH_If_FlashUnlock();
   /* Erase FLASH sectors to download image */
   if (FLASH_If_EraseSectors(APPLICATION_ADDRESS) != 0x00)
   {
     /* Flash erase error: Turn LED4 On and Toggle LED2 and LED3 in
     * infinite loop */
   uart_length=sprintf(uart_tx_buffer, "Flash erase error\n");
   HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
   while(1){}
   }
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
/* Program flash memory */
  COMMAND_ProgramFlashMemory();

  /* Download Done: Turn LED4 Off and LED2 On */
  uart_length=sprintf(uart_tx_buffer, "Download Done\n");
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);

  /* Close file */
  f_close(&MyFileR);
  uart_length=sprintf(uart_tx_buffer, "Application going to be reset in 5
seconds\n");
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
  HAL_Delay(5000);
  NVIC_SystemReset();
}
}
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
static void COMMAND_ProgramFlashMemory(void)
{
  uint32_t programcounter = 0x00;
  uint8_t readflag = TRUE;
  uint16_t bytesread;

  /* RAM Address Initialization */
  RamAddress = (uint32_t) & RAM_Buf;
  /* Erase address init */
  LastPGAddress = APPLICATION_ADDRESS;
  /* While file still contain data */
  while ((readflag == TRUE))
  {
    /* Read maximum 512 Kbyte from the selected file */
    f_read(&MyFileR, RAM_Buf, BUFFER_SIZE, (void *)&bytesread);

    /* Temp variable */
    TmpReadSize = bytesread;
    /* The read data < "BUFFER_SIZE" Kbyte */
    if (TmpReadSize < BUFFER_SIZE)
    {
      readflag = FALSE;
    }
```

USB MSC DFU Host

- Adapt user function to FW update of user flash area – file usb_host.c

```
 /* Program flash memory */
    for (programcounter = 0; programcounter < TmpReadSize; programcounter
+= 4)
    {
      /* Write word into flash memory */
      if (FLASH_If_Write((LastPGAddress + programcounter),
                         *(uint32_t *) (RamAddress + programcounter)) !=
0x00)
      {
        /* Flash programming error: Turn LED2 On and Toggle LED3 in
infinite
         * loop */
  uart_length=sprintf(uart_tx_buffer, "Flash programming error\n");
  HAL_UART_Transmit(&huart3, uart_tx_buffer,(uint16_t)uart_length, 1000);
  while(1){}
      }
    }
    /* Update last programmed address value */
    LastPGAddress += TmpReadSize;
  }
}
/* USER CODE END 0 */
```

USB MSC DFU Host


```
#include "flash_if.h"
```

- To main.c add

```
uint32_t JumpAddress;
pFunction Jump_To_Application;
```

- Function to jump into updatable user code if button is pressed

```
/* USER CODE BEGIN 2 */
 if (HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) != GPIO_PIN_RESET)
    {
      /* Check Vector Table: Test if user code is programmed starting from address
      "APPLICATION_ADDRESS" */
      if ((((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0xFF000000 ) == 0x20000000) || \
        (((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0xFF000000 ) == 0x10000000)){
        /* Jump to user application */
        JumpAddress = *(__IO uint32_t*) (APPLICATION_ADDRESS + 4);
        Jump_To_Application = (pFunction) JumpAddress;
        /* Initialize user application's Stack Pointer */
        __set_MSP(*(__IO uint32_t*) APPLICATION_ADDRESS);
        Jump_To_Application();
      }
    }
/* USER CODE END 2 */
```

USB MSC DFU Host

- To usb_host.h add

```
#include "flash_if.h"
```

- To main.c add

```
uint32_t JumpAddress;
pFunction Jump_To_Application;
```

- In while function again add userFunction(), which is here handling communication with MSC device and FW upgrade

```
while (1)
 {
 /* USER CODE END WHILE */
   MX_USB_HOST_Process();
 /* USER CODE BEGIN 3 */
   userFunction();
 }
```

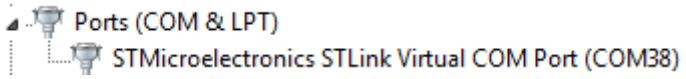
USB MSC DFU Host

- Function to jump into updatable user code if button is pressed

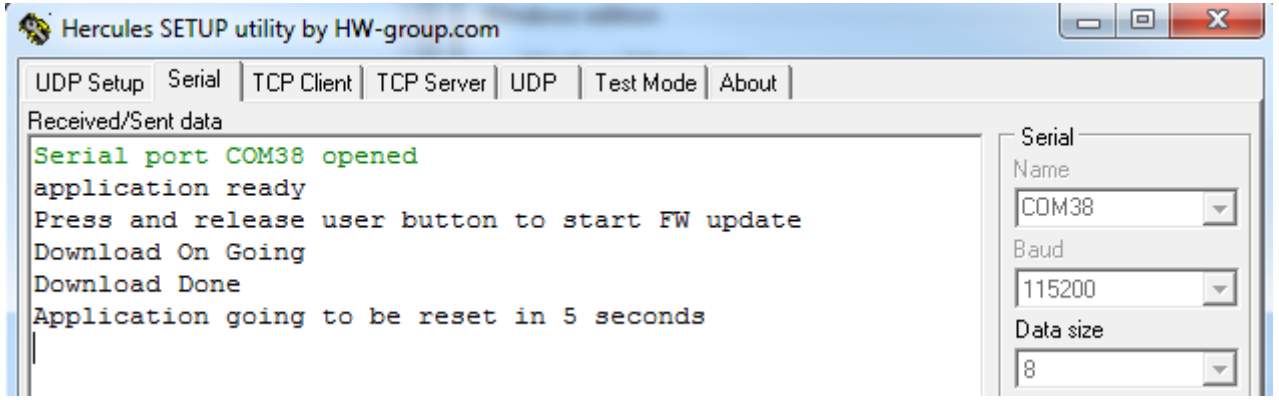
```
/* USER CODE BEGIN 2 */
 if (HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) != GPIO_PIN_RESET)
    {
      /* Check Vector Table: Test if user code is programmed starting from address
      "APPLICATION_ADDRESS" */
      if ((((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0xFF000000 ) == 0x20000000) || \
        (((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0xFF000000 ) == 0x10000000)){
        /* Jump to user application */
        JumpAddress = *(__IO uint32_t*) (APPLICATION_ADDRESS + 4);
        Jump_To_Application = (pFunction) JumpAddress;
        /* Initialize user application's Stack Pointer */
        __set_MSP(*(__IO uint32_t*) APPLICATION_ADDRESS);
        Jump_To_Application();
      }
    }
/* USER CODE END 2 */
```

USB MSC DFU Host

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

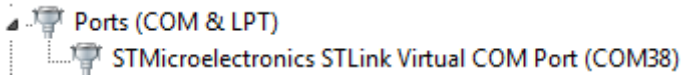


- Debug output with more instructions can be displayed in any COM port terminal application

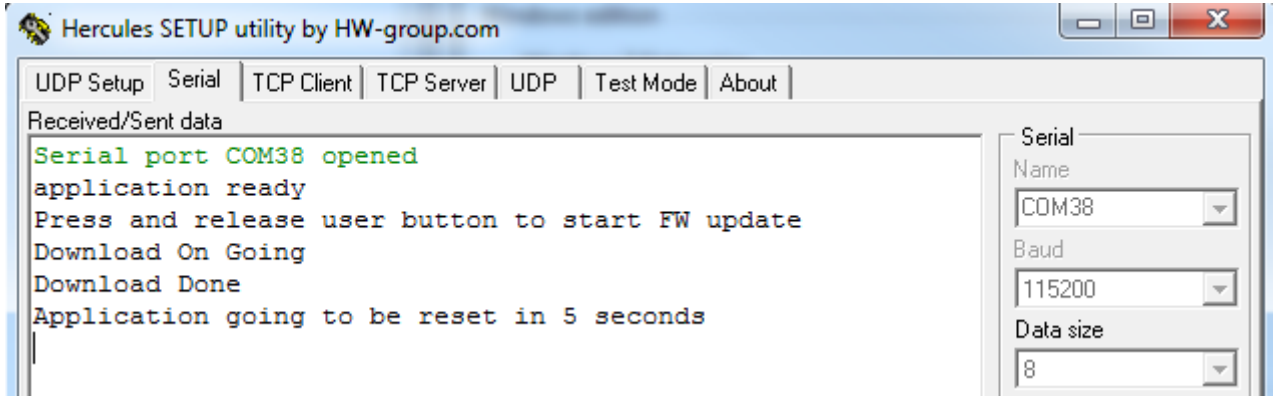


USB MSC DFU Host

- 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



USB MSC DFU Host

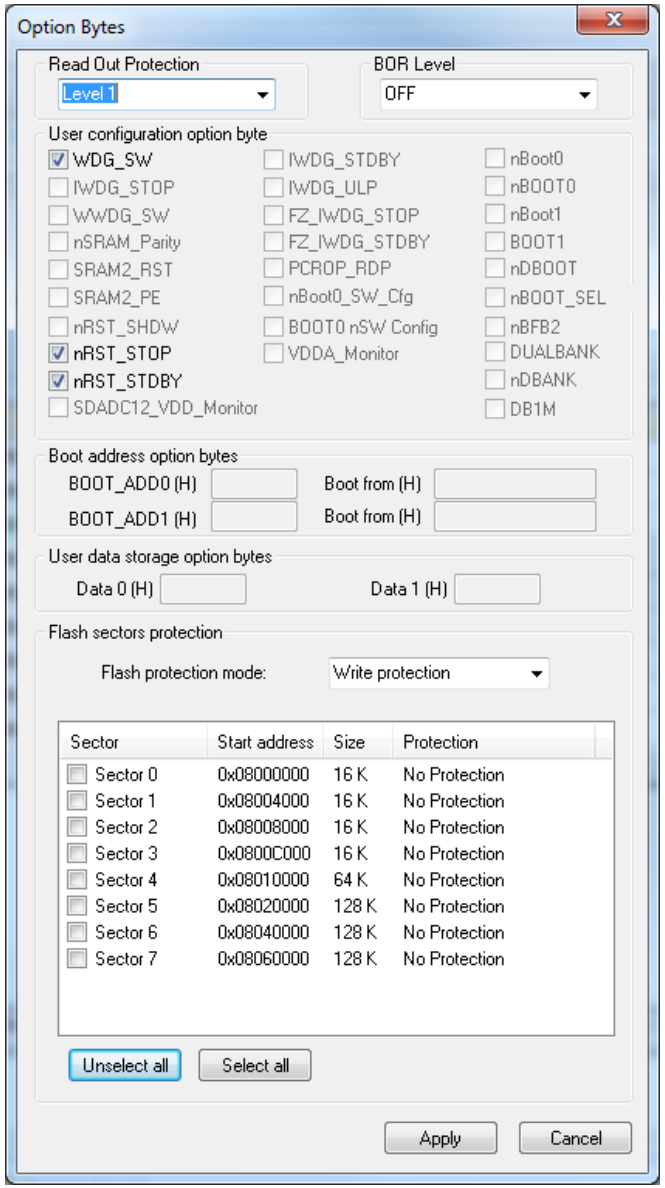
- Now is the project functional and we can test it with simple LED project used already for DFU device
- In this example we need to generate binary file output
  - It's default setting in SW4STM32 project Properties -> C/C++ Build -> Settings -> Build Steps

```
arm-none-eabi-objcopy -O binary "${BuildArtifactFileBaseName}.elf"
"${BuildArtifactFileBaseName}.bin" && arm-none-eabi-size
"${BuildArtifactFileName}"
```

- Rebuild the project, .bin file is created in project structure
  - If you have already created .hex file, you can convert it using ST-link utility
- Copy the binary file to the root folder of the MSC device, rename to "image.bin"

USB MSC DFU Host

- With user DFU bootloader FW update is possible also with enabled read out protection (RDP=1)
- Perform mass erase of the device and load again the host application
  - This step is only to demonstrate, that the code is load to clear area and the application is not loaded from previous update
- Using ST-ink utility set RDP to 1
  - Target-> option bytes
- Application user bootloader is still accessible as it was without Read Out Protection



life.augmented