STM32 Dark/Blue Pill Board Example

wqaxs36

16 Jul 2021 CPOL 3 min read

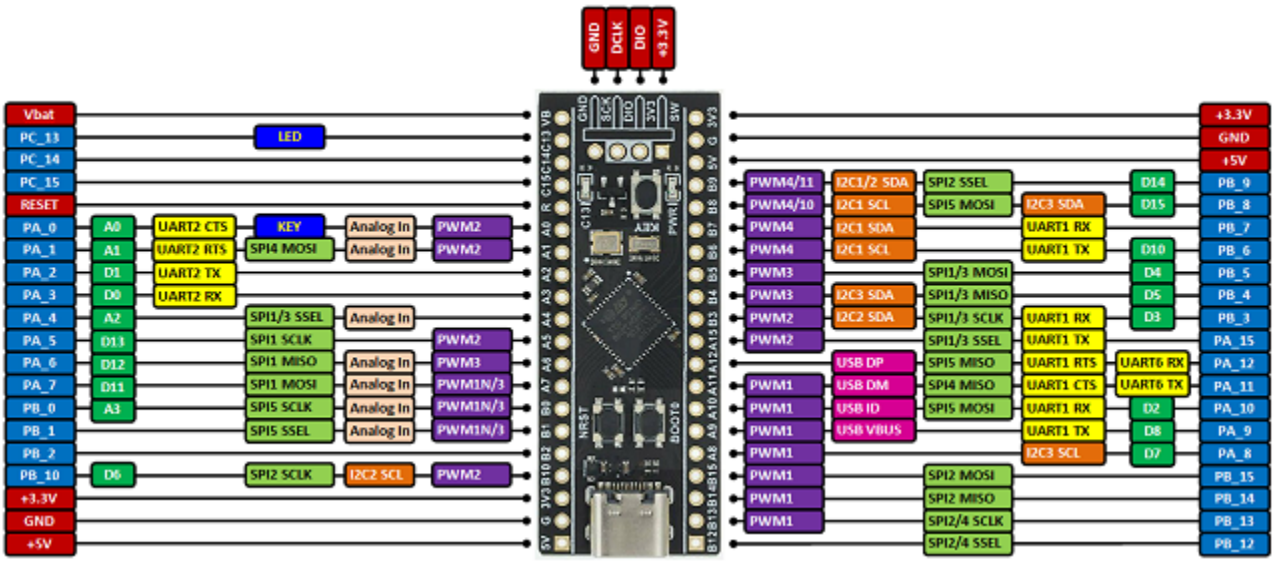
Making a HID device out of dark/blue pill board using STM32CubeIDE

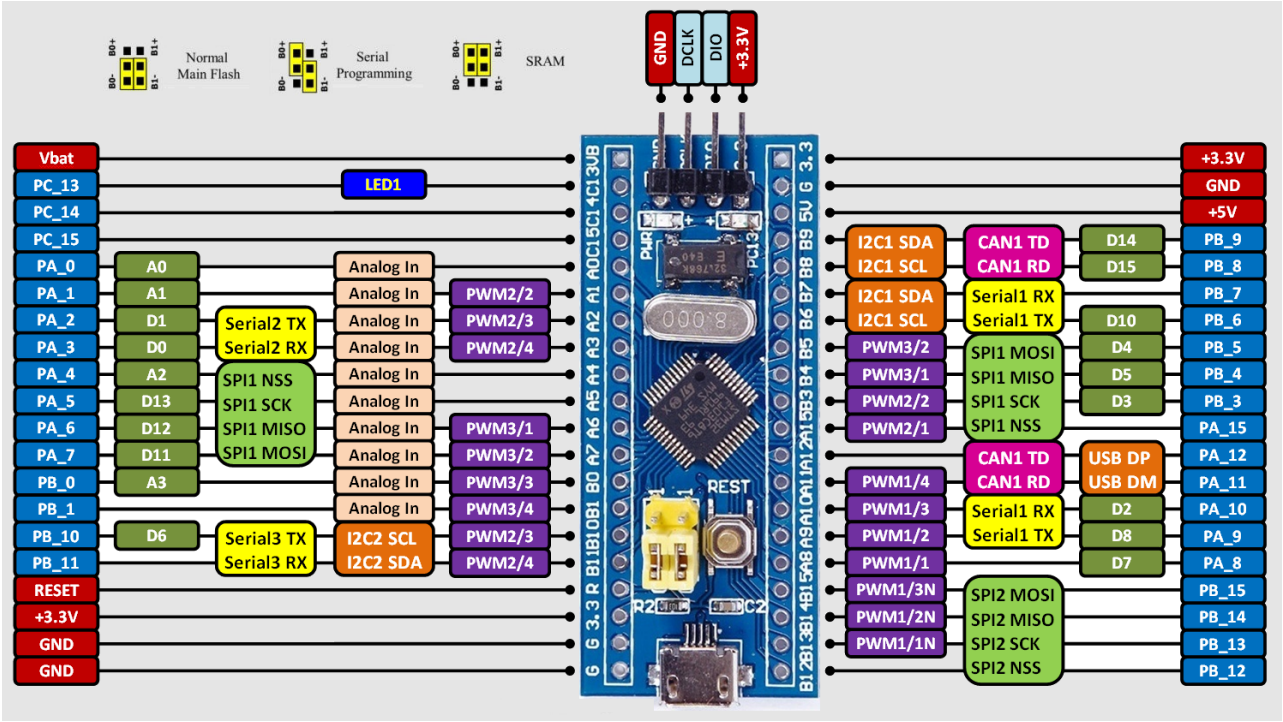
This article describes essential functions of the dark/blue pill API, bootloader and app programming/upload.

Background

First, you'd need STM32CubeIDE, STM32CubeProg, ST-LINK V2 device and STM32F103C8t6 board (blue pill) or ~~STM32F411CEU6 (black pill).~~

Performance comparison: Black pill > Blue/Red/Purple pill > Green pill > Arduino boards





Using the Code

Listing of Essential API's Functions

`GPIO_PIN_SET` and `GPIO_PIN_RESET` names come from SR Flip-flops (learnabout-electronics.org)

`HAL_GPIO_Init()` *// stm32f1xx_hal_gpio.c*

C++ Copy Code

```
void    HAL_GPIO_Init(GPIO_InitTypeDef *GPIO_Init);

#define LEDA_Pin    GPIO_PIN_10
#define LEDB_Pin    GPIO_PIN_11

GPIO_InitTypeDef GPIO_InitStruct = { 0 };
GPIO_InitStruct.Pin   = LEDA_Pin;
GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull  = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

GPIO_InitStruct.Pin  = LEDB_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
```

`HAL_GPIO_WritePin()` *// stm32f1xx_hal_gpio.c*

C++ Copy Code

```
void   HAL_GPIO_WritePin(uint16_t GPIO_Pin, GPIO_PinState PinState);
```

```
HAL_GPIO_WritePin(GPIOB, LEDA_Pin, GPIO_PIN_SET);       // Set HIGH pin
HAL_GPIO_WritePin(GPIOB, LEDB_Pin, GPIO_PIN_RESET);     // Set LOW pin
```

HAL_GPIO_ReadPin() *// stm32f1xx_hal_gpio.c*

C++ Copy Code

```
GPIO_PinState HAL_GPIO_ReadPin(uint16_t GPIO_Pin);

uint8_t LEDB_State = HAL_GPIO_ReadPin(GPIOB, LEDB_Pin);
HAL_GPIO_WritePin(LED_PinA, LEDB_State);
```

HAL_Delay() *// stm32f1xx_hal.c*

C++ Copy Code

```
void    HAL_Delay(uint32_t Delay);

HAL_Delay(1000);    // Wait 1s
```

CDC_Transmit_FS() *// usbd_cdc_if.c*

C++ Copy Code

```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len);

uint8_t* Data = "Hello World\n\r";
CDC_Transmit_FS(Data, 13);
```

CDC_Receive_FS() *// usbd_cdc_if.c*

This function is callback, thus you do not have to call this, the framework calls it for you, call `SerialReceive()` instead. The length of the buffer is 63 characters and I don't know why yet.

Modified version:

C++ Copy Code

```
static uint8_t  Buffer[64];
static uint32_t BufferLenght;

static int8_t CDC_Receive_FS(uint8_t* Data, uint32_t* Lenght)
{
    USBD_CDC_SetRxBuffer(&hUsbDeviceFS, Buffer);
    USBD_CDC_ReceivePacket(&hUsbDeviceFS);

    BufferLenght = *Lenght;
    memcpy(Buffer, Data, *Lenght);

    return (USBD_OK);
}
void    SerialReceive(uint8_t* Data, uint32_t* Lenght)
{
    *Lenght = BufferLenght;
    memcpy(Data, Buffer, BufferLenght);
}
```

Example of use:

C++ Copy Code

```
uint8_t  Buffer[64];
uint32_t BufferLenght;

SerialReceive(Buffer, &BufferLenght);
```

USBD_HID_SendReport() *// usbd_customhid.c*

`HID_Send()` is similar to `USBD_HID_SendReport()` but with one less parameter to handle.

C++ Copy Code

```
uint8_t USBD_CUSTOM_HID_SendReport(USBD_HandleTypeDef* pdev, uint8_t* report, uint16_t
len);
....

extern  USBD_HandleTypeDef hUsbDeviceFS;
void    HID_Send(uint8_t* Data, uint16_t Size)
{
    USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, Data, Size);
}
```

Example of use:

C++ Copy Code

```
#define REPORT_ID_SIZE  1
#define REPORT_SIZE     2

uint8_t ReportData[REPORT_ID_SIZE + REPORT_SIZE] = {0x02, 1, 2};
HID_Send(ReportData, sizeof(ReportData));
```

USBD_CUSTOM_HID_DataOut() *// usbd_customhid.c*

This function is callback thus you do not have to call this, the framework calls it for you, use `HID_Read()` instead.

Change `USBD_CUSTOMHID_OUTREPORT_BUF_SIZE` along what is in your HID Report Descriptor (`OUT_REPORT_COUNT`).

Modified version:

C++ Shrink ▲ Copy Code

```
uint8_t     OutputData[USBD_CUSTOMHID_OUTREPORT_BUF_SIZE];
uint8_t     OutputSize;

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[0],
                                                              hhid->Report_buf[1]);

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

    OutputSize = USBD_CUSTOMHID_OUTREPORT_BUF_SIZE;
    memcpy(OutputData, hhid->Report_buf, OutputSize);

    return USBD_OK;
}
```

```
uint8_t     HID_Read(uint8_t* Data, uint16_t* Size)
{
    if (OutputSize == 0)
        return 0;

    memcpy(Data + 1, OutputData, OutputSize);
    *Size = OutputSize;

    OutputSize = 0;

    return 1;
}
```

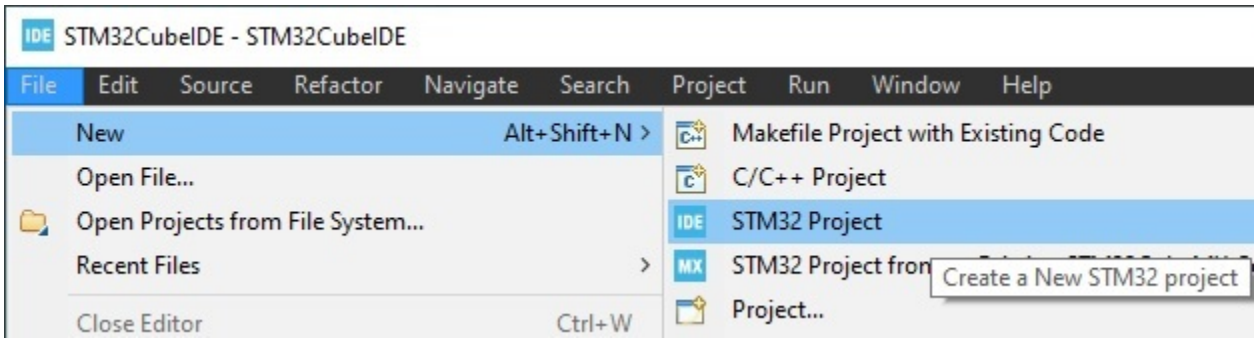
Example of use:

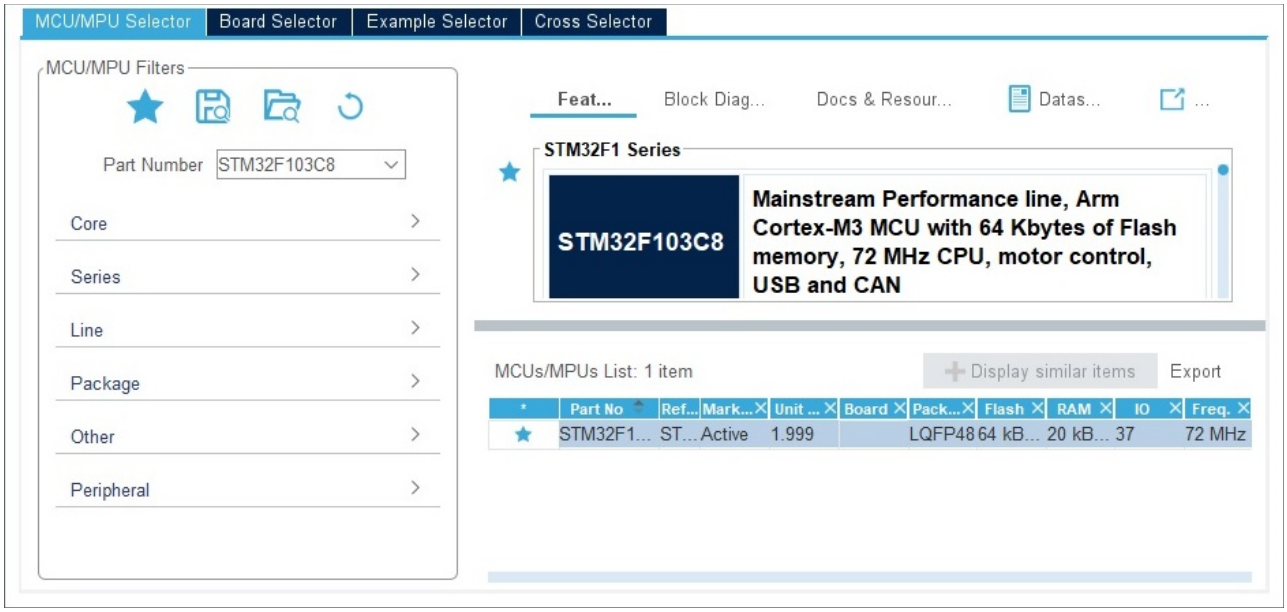
C++ Copy Code

```
#define REPORT_ID_SIZE  1
#define REPORT_SIZE     2

uint8_t  ReportData[REPORT_ID_SIZE + REPORT_SIZE] = {0x02, 1, 2};
uint16_t DataSize;

HID_Read(ReportData, &DataSize);
HID_Send(ReportData, REPORT_ID + REPORT_SIZE);
```

Bootloader Programming

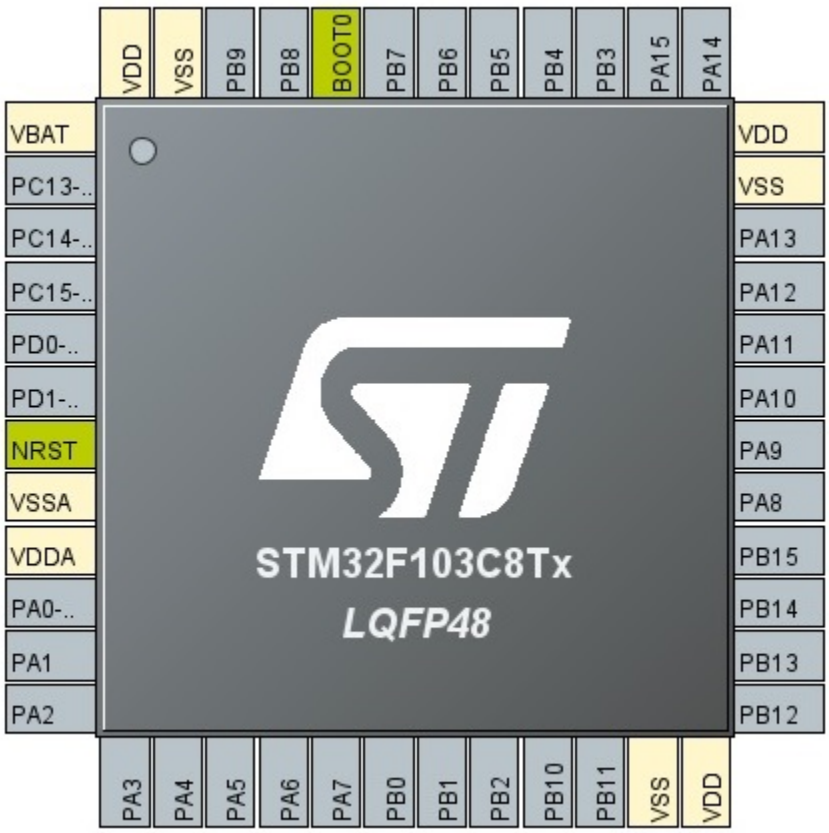
After installing STM32CubeIDE, open it and create a new project.





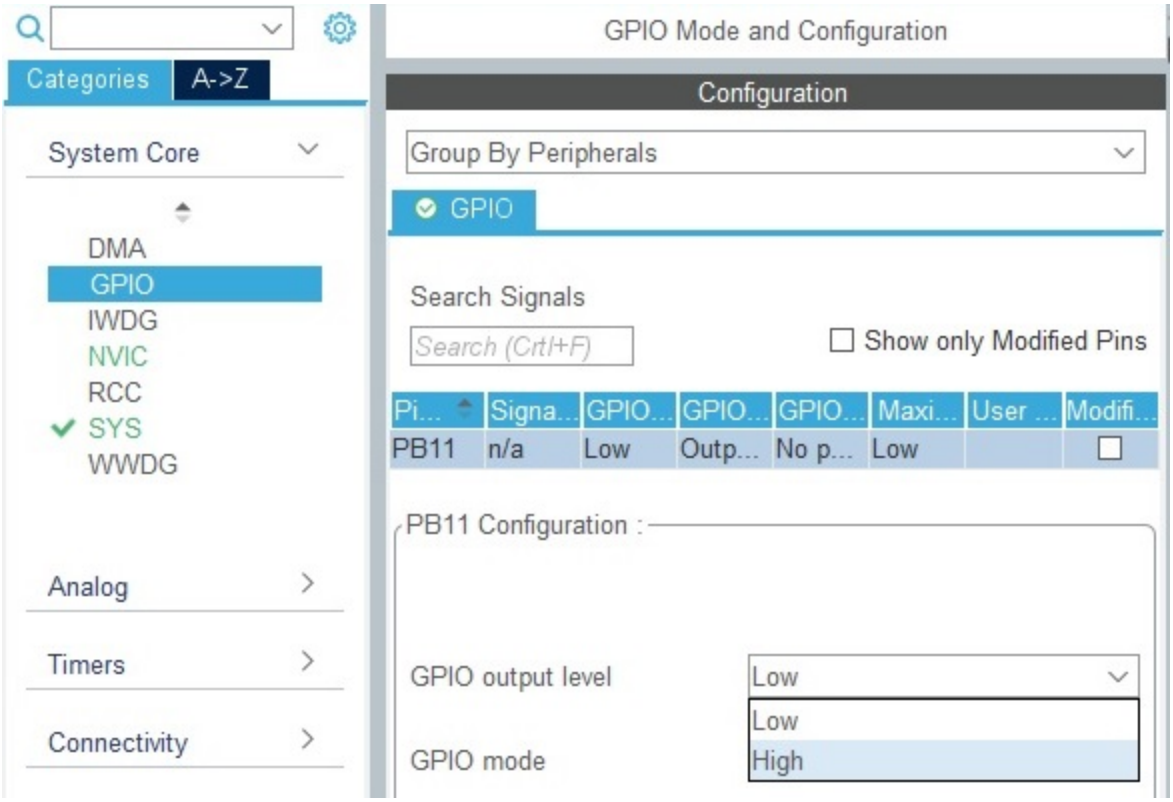
Next -> **Next** -> **Finish**

You are now seeing the MCU chosen:



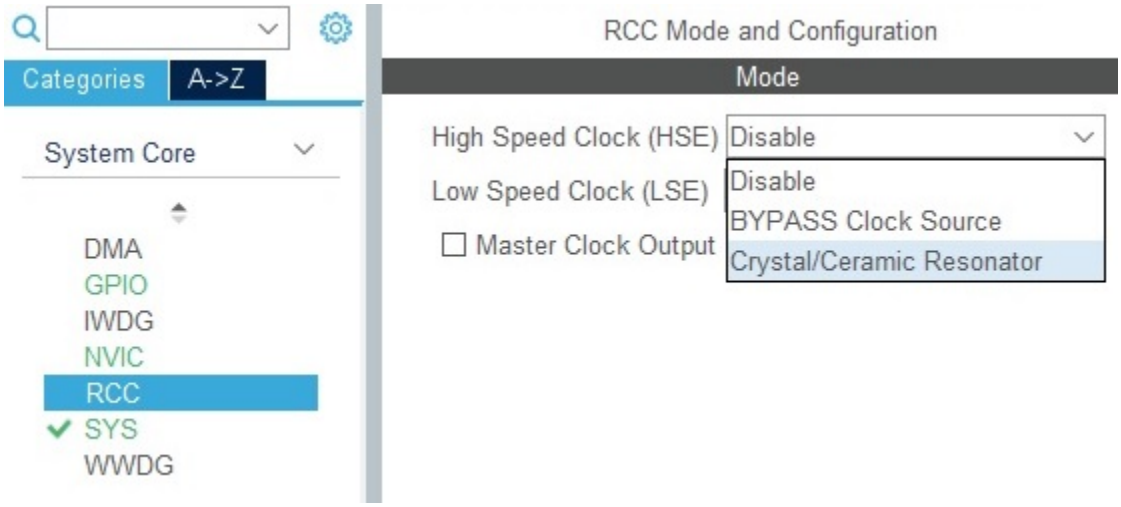
To be able to keep an eye on normal/programming mode, a LED can help that out.

Right click on **PB11** and choose **GPIO_Output**, then set it at high level.

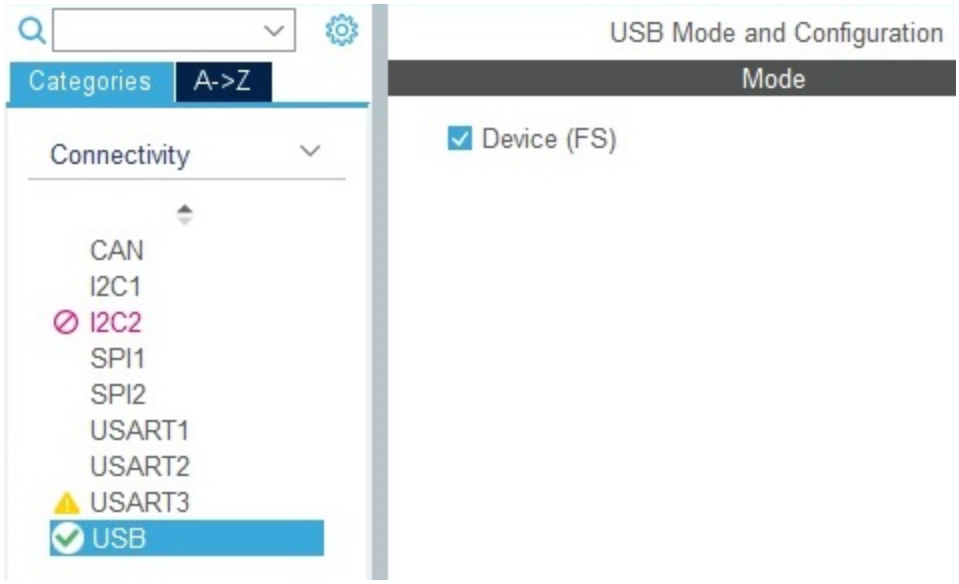


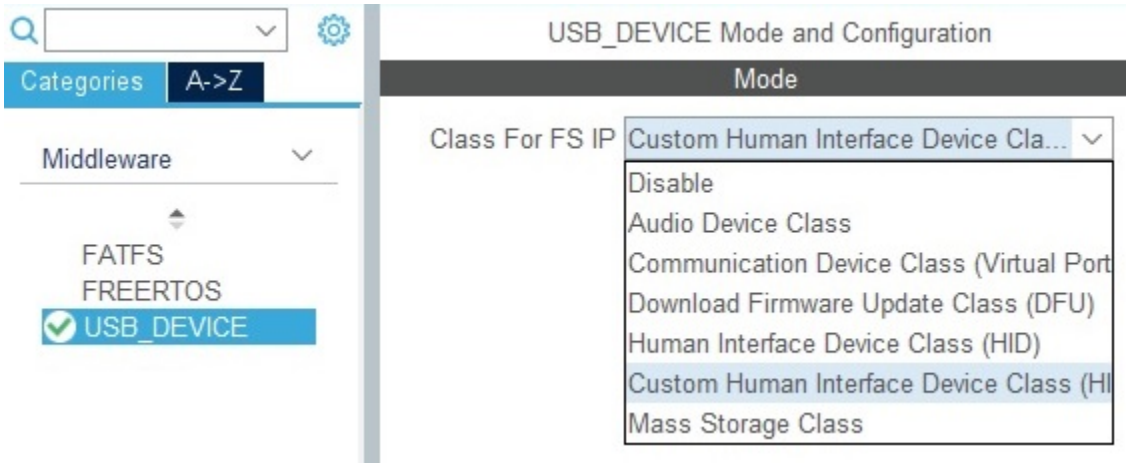
Then attach a LED to B11 pin and ground it.

Then we have to setup a clock device



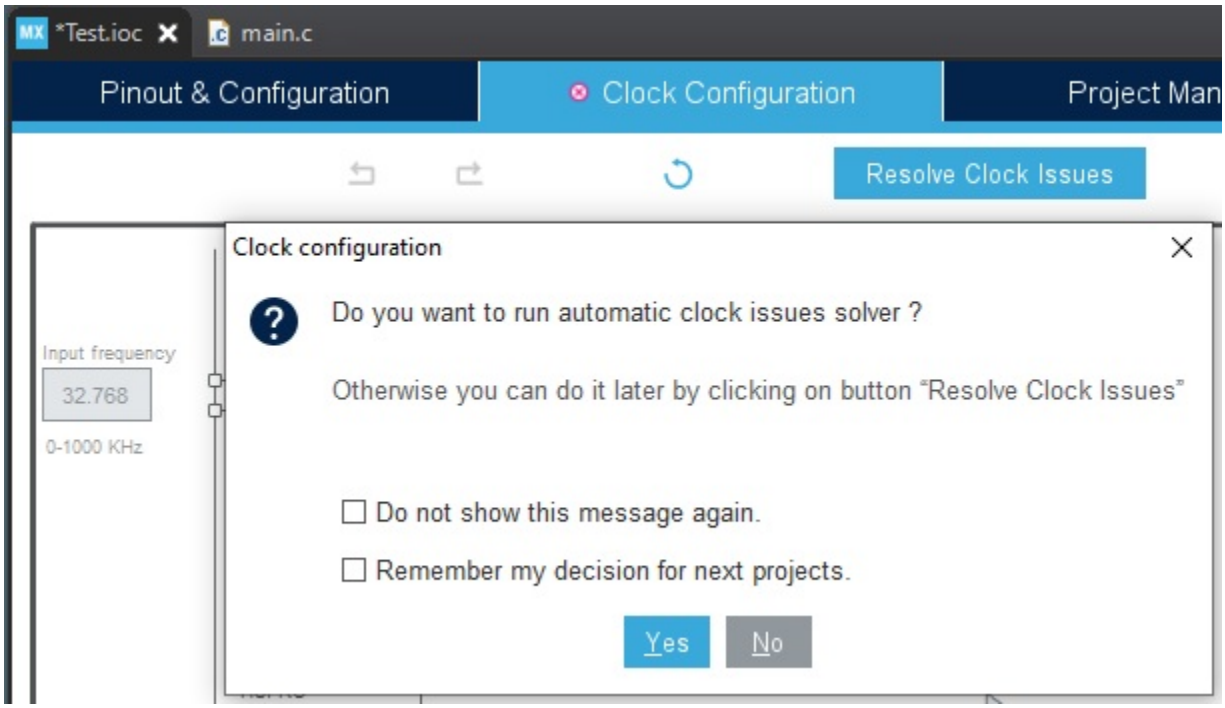
After that, we can setup the USB mode:





We choose Custom HID class rather HID class since Custom HID class offers the possibility to read HID packet, not the other one.

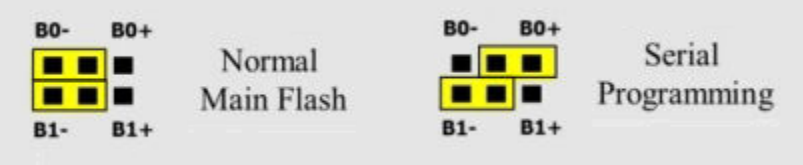
After that, we have to configure the clock of the MCU as the final step to make everything work together.



Just click **Yes**, **Ctrl-S** and wait the code to generate.

Then, right click on the hammer icon and choose **Release**.

Now that the **.bin* been made, prepare your board by selecting the Programming mode and press **Reset** pushbutton onboard.

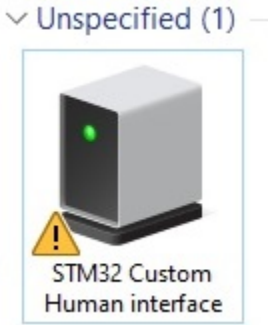


Open **STM32CubeProgrammer** - Click on **Connect** button - Click on **Open File**.

Select the **.bin* inside *Release* folder of your project - Click on **Download**.

Put back the normal mode - Unplug the STLink - Re/plug the USB cable of your board to the PC.

At the end, you should see this and it's normal.



Device/App Programming

First, you would need to modify some data to prepare the device.

In *\USB_DEVICE\App\usbd_desc.c*, set the new data:

C++ Copy Code

```
#define USBD_VID                        0x0001
#define USBD_PID_FS                     0x0001
#define USBD_LANGID_STRING              1033
#define USBD_MANUFACTURER_STRING        "Manufacturer Name"
#define USBD_PRODUCT_STRING_FS          "Product Name"
#define USBD_CONFIGURATION_STRING_FS    "Configuration Name"
#define USBD_INTERFACE_STRING_FS        "Interface Name"
```

In the same directory go to *usbd_custom_hid_if.c and* replace `CUSTOM_HID_ReportDesc_FS[]` by:

C++ Shrink ▲ Copy Code

```
enum HID_Helper
{
    REPORT_ID_SIZE   = 1,   // 1 byte added to each HID packets

    IN_REPORT_SIZE   = 8,   // Bit size of one HID packet
    IN_REPORT_COUNT  = 2,   // Number of HID packets

    OUT_REPORT_SIZE  = 8,   // Bit size of one HID packet
    OUT_REPORT_COUNT = 2    // Number of HID packets
```

```
};

// USBD_CUSTOM_HID_REPORT_DESC_SIZE is 33
__ALIGN_BEGIN static uint8_t CUSTOM_HID_ReportDesc_FS[USBD_CUSTOM_HID_REPORT_DESC_SIZE]
__ALIGN_END =
{
    0x06, 0x00, 0xFF,           //  USAGE_PAGE (Vendor Specific)
    0x09, 0x01,                 //  USAGE (1)
    0xA1, 0xFF,                 //  COLLECTION (Vendor Specific)

    0x15, 0x00,                 //      LOGICAL_MINIMUM (0)
    0x26, 0xFF, 0x00,           //      LOGICAL_MAXIMUM (255 possibilities = 2 ^ 8 bits)

    0x85, 0x02,                 //      REPORT_ID (2)
    0x75, IN_REPORT_SIZE,       //
    0x95, IN_REPORT_COUNT,      //
    0x09, 0x00,                 //      USAGE (0)
    0x81, 0x00,                 //      INPUT (Data,Ary,Abs)

    0x85, 0x01,                 //      REPORT_ID (1)
    0x75, OUT_REPORT_SIZE,      //
    0x95, OUT_REPORT_COUNT,     //
    0x09, 0x00,                 //      USAGE (0)
    0x91, 0x00,                 //      OUTPUT (Data,Ary,Abs)

    0xC0                        //  END_COLLECTION
};
```

To learn more about HID Report Descriptor, go to Human Interface Devices (HID) Information | USB-IF

In *\Middlewares\ST\STM32_USB_Device_Library\Class\CustomHID\Src\usbd_customhid.c*, add the code below before `USBD_CUSTOM_HID_SendReport()`.

C++ Copy Code

```
uint8_t USBD_CUSTOM_HID_SendReport(USBD_HandleTypeDef* pdev, uint8_t* report, uint16_t len);
....

extern  USBD_HandleTypeDef hUsbDeviceFS;
void    HID_Send(uint8_t* Data, uint16_t Size)
{
    USBD_CUSTOM_HID_SendReport(&hUsbDeviceFS, Data, Size);
}
```

Update `USBD_CUSTOM_HID_DataOut()` with this:

C++ Shrink ▲ Copy Code

```
uint8_t     OutputData[USBD_CUSTOMHID_OUTREPORT_BUF_SIZE];
uint8_t     OutputSize;

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[0],
                                                              hhid->Report_buf[1]);

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

```
    OutputSize = USBD_CUSTOMHID_OUTREPORT_BUF_SIZE;
    memcpy(OutputData, hhid->Report_buf, OutputSize);

    return USBD_OK;
}
uint8_t     HID_Read(uint8_t* Data, uint16_t* Size)
{
    if (OutputSize == 0)
        return 0;

    memcpy(Data + 1, OutputData, OutputSize);
    *Size = OutputSize;

    OutputSize = 0;

    return 1;
}
```

Now we're ready to start programming the App.

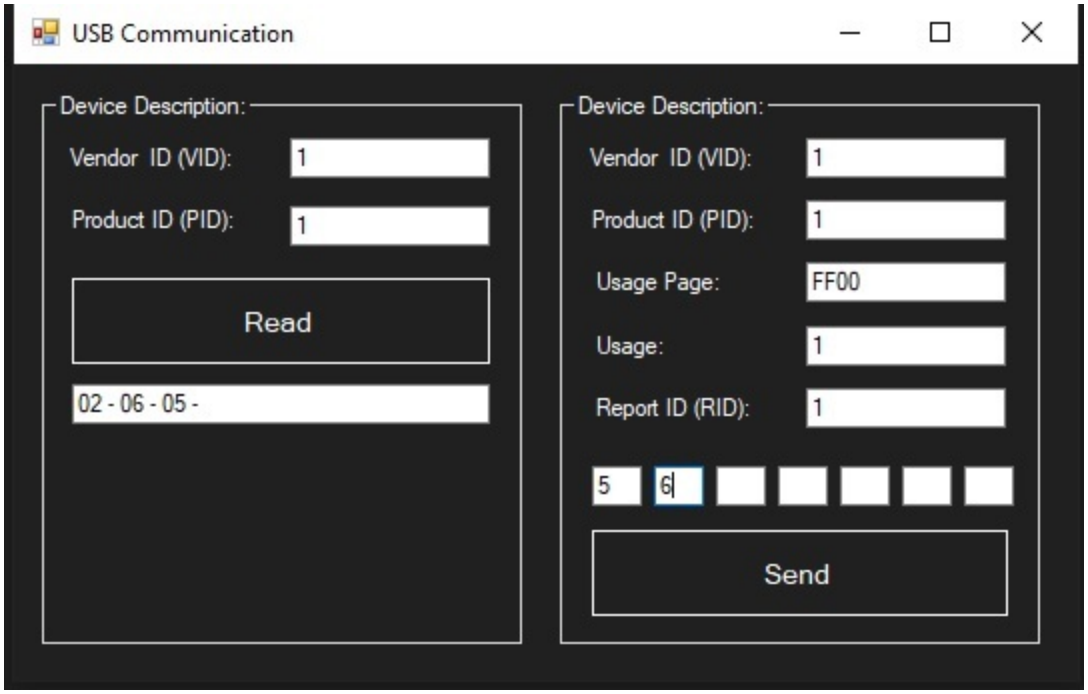
Go to *Core**Src**main.c* and paste inside the `main()`:

C++ Copy Code

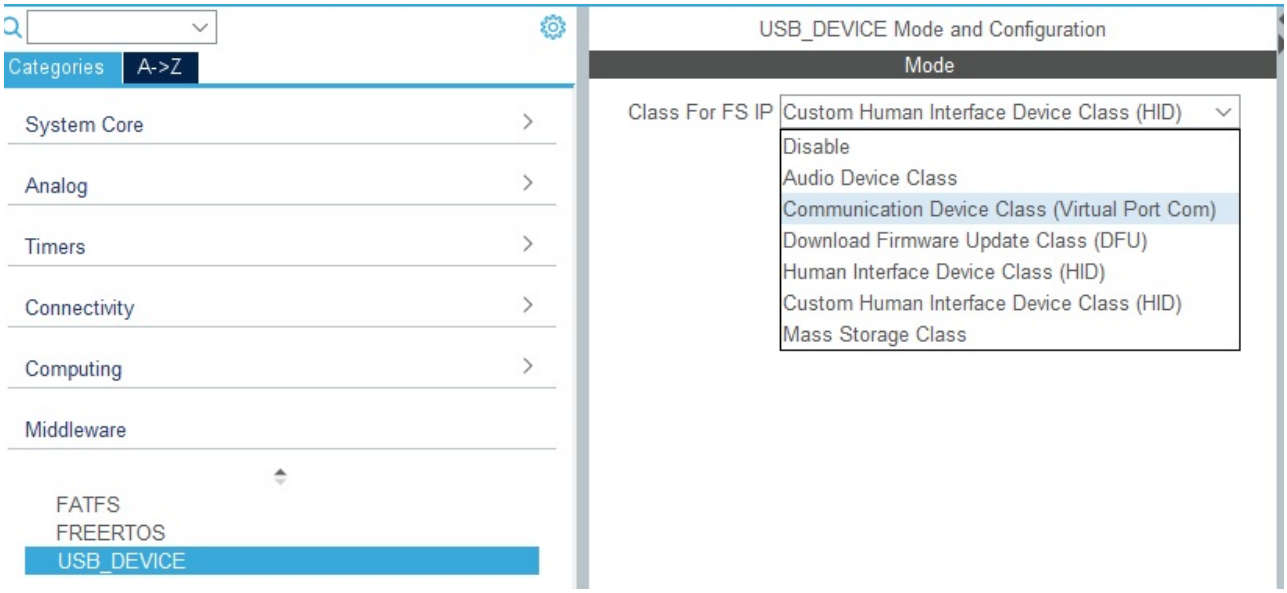
```
int     main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_USB_DEVICE_Init();

    #define REPORT_ID_SIZE  1
    #define REPORT_SIZE     2

    uint8_t  ReportData[REPORT_ID_SIZE + REPORT_SIZE] = {0x02, 1, 2};
    uint16_t DataSize;

    while (1)
    {
        HID_Read(ReportData, &DataSize);
        HID_Send(ReportData, REPORT_ID_SIZE + REPORT_SIZE);

        HAL_Delay(500);
    }
}
```

Now build and upload this firmware to your board and to interact with this firmware, you would need two tools.

A USB sniffer like GitHub - djpnewton/busdog and/or a HID comm tool How to HID Protocol - CodeProject.



Apparently, we cannot use Serial and HID mode in the same time with STM boards, thus you would need to select CDC in `USB_DEVICE` to be able to use it.



Double click on a tab file (e.g., *main.c*) in the IDE to fullscreen the current file.

Press **Ctrl+Shift+W** to restore the workspace.

`STM32CubeProg` has an interesting feature called **Full Chip Erase** (tab below), useful if your chip has weird behavior.

For black pill tuto, I will wait receiving the board next month but it's roughly the same way of programming process.

FS means Full-Speed for USB spec:

Copy Code

```
LS  =  Low Speed  =   1.5 Mb
FS  = Full Speed  =  12   Mb
HS  = High Speed  = 480   Mb
SS  = SuperSpeed  =   5   Gb
```

```
SS+ = SuperSpeed+ =  10   Gb
SS+ = SuperSpeed+ =  20   Gb
SS+ = SuperSpeed+ =  40   Gb
```

Points of Interest

Programming with STM boards is much more accessible and has more possibilities than with Arduino boards.

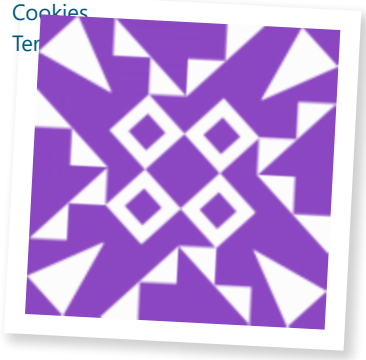
History

- 16th July, 2021 : Initial version

License

Share

About the Author



wqaxs36
France

No Biography provided

Comments and Discussions

You must Sign In to use this message board.

Search Comments

Spacing Relaxed Layout Normal Per page 25 Update

First Prev Next

My vote of 5	Ștefan-Mihai MOGA	16-Jul-21 2:04
Re: My vote of 5	wqaxs36	16-Jul-21 2:25

Last Visit: 19-Jul-21 1:16 Last Update: 19-Jul-21 7:19 Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.