How to use STMicroelectronics classic USB device middleware with new STM32 families

B.Montanari
ST Employee

on 2023-10-26 08:13 AM - edited on 2024-01-03 09:04 AM by ADMIN **Laurids_PETERSEN**

Summary

This article presents a tutorial for importing and using the legacy STMicroelectronics USB middleware in the new lines of STM32 to implement a CDC class to open a virtual COM port. The STM32H5 series was selected for this tutorial, but the steps can be easily tailored to some other STM32 families, such as STM32U5.

For all the STM32 series released from 2021 onwards, the now classic, STMicroelectronics USB middleware is no longer offered as part of the STM32Cube native offer. But to maintain compatibility with legacy applications, this package is still available to be downloaded and manually included in your project via our GitHub page. You can download the package from the following links:

GitHub - STMicroelectronics - Middleware USB Device

GitHub - STMicroelectronics - Middleware USB Host

This tutorial was built using STM32CubeIDE 1.13.1, using the STM32CubeH5 HAL driver version 1.1.1 and using the NUCLEO-H503RB, which embeds an STM32H503RBT6 MCU. For further details about this board, refer to the device user manual.

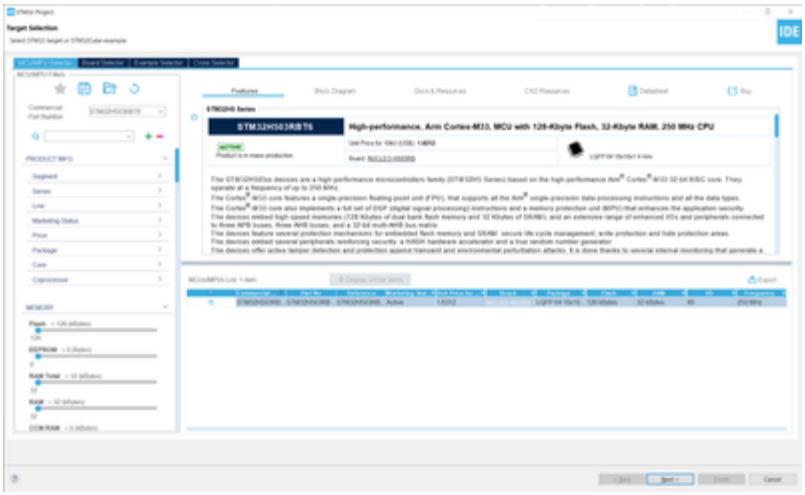
For a detailed explanation regarding the classic USB library, refer to to the USB wiki and the STM32 USB training:

ST Wiki - Introduction to USB with STM32

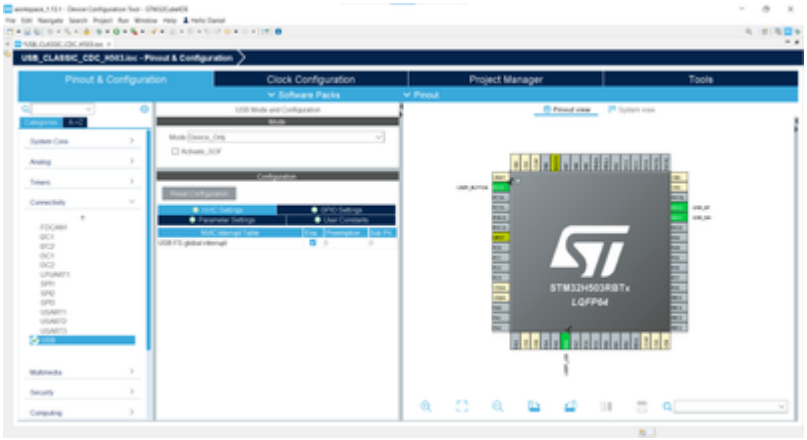
MOOC - STM32 USB Training

1. Development

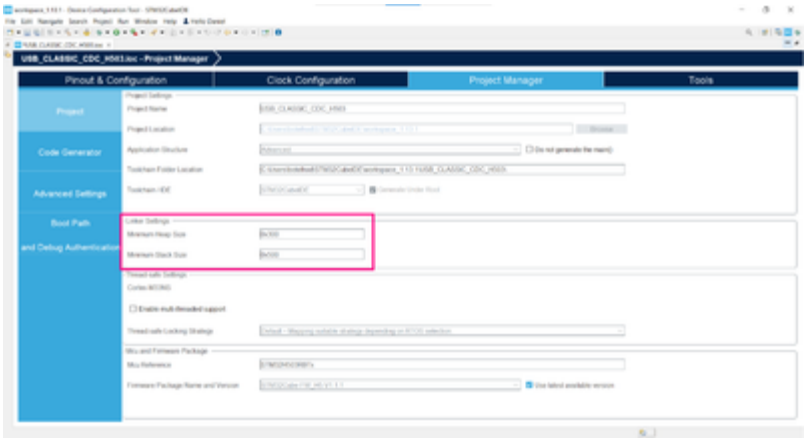
Start by creating a project for the STM32H503RB in the STM32CubeIDE.



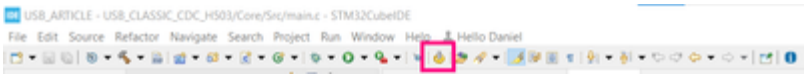
Once the project creation is done, enable the USB Peripheral in Device only mode and activate its interrupt in the NVIC Settings tab.



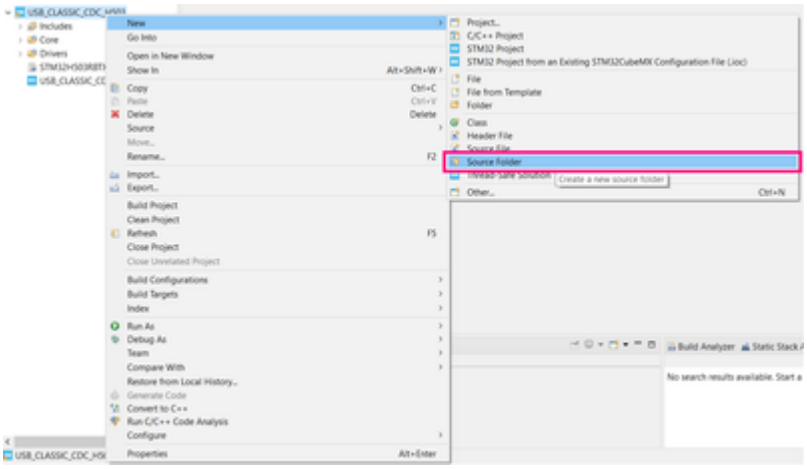
After that, increase the amount of heap and stack size of the project as indicated in the image below, this action can be done in the Project Manager tab:



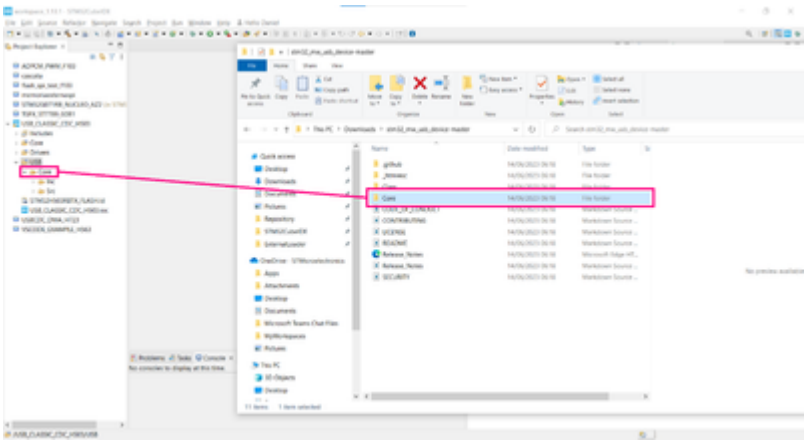
After doing that, generate the code by clicking on the highlighted icon, or just pressing the alt + K shortcut.



Once the code is generated, create a source folder in your project called USB.

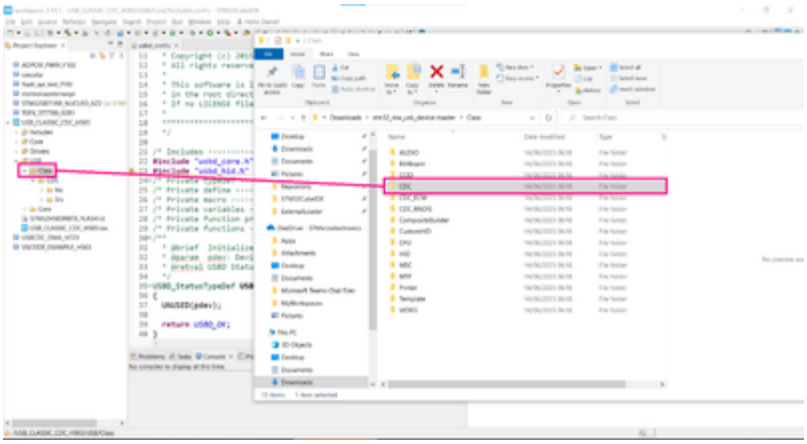


Download the USB Device package from the GitHub (the same link presented in the introduction of this article). Open the software pack folder and import the Core folder into the created ‘USB’ folder.

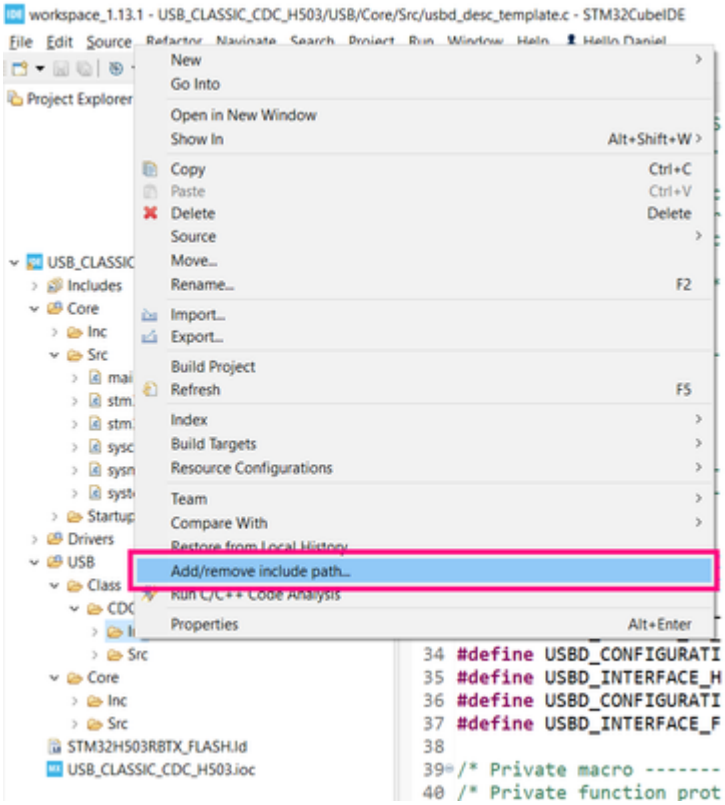


Now, create another folder called ‘Class’ and import the desired classes to your project, in this case, the CDC is the only one selected.

Feedback



Add the path to the include reference. Right click in the ‘Inc’ folder inside the USB/Class /CDC/.. path, then select `Add/remove include path…` option, and in the pop-up menu, press OK. Perform the same steps for the USB/Core/Inc as well.



Make sure to rename the USB/Core/Inc/usbd_conf_template.h and USB/Core /Src/usbd_conf_template.c files to usbd_conf.h/.c. After doing so, open the usbd_conf.h. In this file, we need to replace the HAL driver header file name according to the MCU Family we are using. In this case, the STM32H503 is being used, so the include is as follows:

Feedback

```
/* Includes ------------------------------------------------------
#include "stm32h5xx.h"  /* replace 'stm32xxx' with your HAL driver he
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

In the next step, you should change the defines according to your application. In this example, we set the defines as follows (the other defines will not be used and can be deleted):

```
/** @defgroup USBD_CONF_Exported_Defines
  * @{
  */
#define USBD_MAX_NUM_INTERFACES                     1U
#define USBD_MAX_NUM_CONFIGURATION                  1U
#define USBD_MAX_STR_DESC_SIZ                       0x100U
#define USBD_SELF_POWERED                           1U
#define USBD_DEBUG_LEVEL                            0U
/* #define USBD_USER_REGISTER_CALLBACK                 1U */
/* ECM, RNDIS, DFU Class Config */
#define USBD_SUPPORT_USER_STRING_DESC               0U
/* BillBoard Class Config */
#define USBD_CLASS_USER_STRING_DESC                 0U
#define USBD_CLASS_BOS_ENABLED                      0U
#define USB_BB_MAX_NUM_ALT_MODE                     0x2U
/* CDC Class Config */
#define USBD_CDC_INTERVAL                           2000U
```

After that, the usbd_conf.h file can be saved and closed. Now, open the usbd_conf.c file. In this file, we need to link the HAL PCD drivers with the middleware. First, add the used class include header files:

```
/* Includes ------------------------------------------------------------------
#include "main.h"
#include "usbd_core.h"
#include "usbd_cdc.h" /* Include class header file */
```

The second step is to import the PCD_HandleTypeDef to this file:

```
/* Private variables ---------------------------------------------------------
extern PCD_HandleTypeDef hpcd_USB_DRD_FS;
```

The next one is to create a function prototype, which will be implemented later in this file:

```
/* Private function prototypes -----------------------------------
static USBD_StatusTypeDef USBD_Get_USB_Status(HAL_StatusTypeDef hal_s
```

Then, we need to implement the PCD Callbacks and populate as follows:

```
/* Private functions ---------------------------------------------
void HAL_PCD_SetupStageCallback(PCD_HandleTypeDef *hpcd)
{
  USBD_LL_SetupStage((USBD_HandleTypeDef*)hpcd->pData, (uint8_t *)hpc
}

void HAL_PCD_DataOutStageCallback(PCD_HandleTypeDef *hpcd, uint8_t ep
{
  USBD_LL_DataOutStage((USBD_HandleTypeDef*)hpcd->pData, epnum, hpcd-
}

void HAL_PCD_DataInStageCallback(PCD_HandleTypeDef *hpcd, uint8_t epn
{
  USBD_LL_DataInStage((USBD_HandleTypeDef*)hpcd->pData, epnum, hpcd->
}

void HAL_PCD_SOFCallback(PCD_HandleTypeDef *hpcd)
{
  USBD_LL_SOF((USBD_HandleTypeDef*)hpcd->pData);
}

void HAL_PCD_ResetCallback(PCD_HandleTypeDef *hpcd)
{
  USBD_SpeedTypeDef speed = USBD_SPEED_FULL;
  if ( hpcd->Init.speed != PCD_SPEED_FULL)
  {
    Error_Handler();
  }
    /* Set Speed. */
  USBD_LL_SetSpeed((USBD_HandleTypeDef*)hpcd->pData, speed);
```

View more

Feedback

Finally, implement the code to perform the driver link for the STM32H503:

```
USBD_StatusTypeDef USBD_LL_Init(USBD_HandleTypeDef *pdev)
{
      pdev->pData  = &hpcd_USB_DRD_FS;
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x00 , PC
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x80 , PC
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_IN_EP
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_OUT_E
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_CMD_E
      return USBD_OK;
}

USBD_StatusTypeDef USBD_LL_DeInit(USBD_HandleTypeDef *pdev)
{
      HAL_StatusTypeDef hal_status;
      hal_status = HAL_PCD_DeInit(pdev->pData);
      return USBD_Get_USB_Status(hal_status);
}

USBD_StatusTypeDef USBD_LL_Start(USBD_HandleTypeDef *pdev)
{
      HAL_StatusTypeDef hal_status;
      hal_status = HAL_PCD_Start(pdev->pData);
      return  USBD_Get_USB_Status(hal_status);
}

USBD_StatusTypeDef USBD_LL_Stop(USBD_HandleTypeDef *pdev)
{
      HAL_StatusTypeDef hal_status;
      hal_status = HAL_PCD_Stop(pdev->pData);
      return USBD_Get_USB_Status(hal_status);
```

View more

Feedback

Doing that, all the needed changes in the usbd_conf files are done and we can move forward to the usbd_desc files. So, rename the USB/Core/Src/usbd_desc_template.c and USB/Core/Inc/usbd_desc_template.h to usbd_desc.h/.c. Then, let us start by opening the usbd_desc.c file. Here, we need to start changing some defines for the descriptor:

```
1  /* Private define ------------------------------------------------
2  #define USBD_VID                     0x0483
3  #define USBD_PID                     22336  /* Replace '0xaaaa' with
4  #define USBD_LANGID_STRING           1033   /* Replace '0xbbb' with
5  #define USBD_MANUFACTURER_STRING     "STMicroelectronics" /* Add you
6  #define USBD_PRODUCT_HS_STRING       "STM32 Virtual ComPort" /* Add
7  #define USBD_PRODUCT_FS_STRING       "STM32 Virtual ComPort" /* Add
8  #define USBD_CONFIGURATION_HS_STRING "CDC Config" /* Add your config
9  #define USBD_INTERFACE_HS_STRING     "CDC Interface" /* Add your Int
10 #define USBD_CONFIGURATION_FS_STRING "CDC Config" /* Add your config
11 #define USBD_INTERFACE_FS_STRING     "CDC Interface" /* Add your Int
```

And change the descriptor settings, so the VCOM class can work properly (the lines 12 and 13 are the only ones that change from default):

```
__ALIGN_BEGIN uint8_t USBD_DeviceDesc[USB_LEN_DEV_DESC] __ALIGN_END =
{
  0x12,                       /* bLength */
  USB_DESC_TYPE_DEVICE,       /* bDescriptorType */
#if ((USBD_LPM_ENABLED == 1) || (USBD_CLASS_BOS_ENABLED == 1))
  0x01,                       /*bcdUSB */     /* changed to USB versi
                                               in order to support BOS
#else
  0x00,                       /* bcdUSB */
#endif /* (USBD_LPM_ENABLED == 1) || (USBD_CLASS_BOS_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),           /* idVendor */
  HIBYTE(USBD_PID),           /* idVendor */
  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 */
}; /* USB_DeviceDescriptor */
```

You can change the template resource names to the most convenient name according to your implementation. For example, changing the function:

```
uint8_t *USBD_Class_DeviceDescriptor(USBD_SpeedTypeDef speed, uint16_
```

Feedback

to:

```
uint8_t *USBD_CDC_DeviceDescriptor(USBD_SpeedTypeDef speed, uint16_t
```

Make sure to update all the references related to these resources to avoid compiling warnings and errors.

Doing that, the editions in the usbd_desc.c are done, save this file and open the usbd_desc.h. In this file, edit the USBD_DescriptorsTypeDef exporting line to reflect the variable declared in the usbd_desc.c file. Since we did not change the resources name in this file, to make it easier to be followed, the standard name is used:

```
/* Exported functions ------------------------------------------
extern USBD_DescriptorsTypeDef Class_Desc; /* Replace 'XXX_Desc' with
```

You can save and close both usbd_desc.h/.c files and move to the next one. In the following step, we need to rename the USB/Class/CDC/Inc/usbd_cdc_if_template.h and USB/Class/CDC/Src/usbd_cdc_if_template.c to usbd_cdc_if.h/.c. In the usbd_cdc_if.h file, add the following defines:

```
/* Exported constants --------------------------------------------
#define APP_RX_DATA_SIZE  512
#define APP_TX_DATA_SIZE  512
```

Add the exported function to transmit data through the VCP:

```
/* Exported functions --------------------------------------------
uint8_t TEMPLATE_Transmit(uint8_t* Buf, uint16_t Len);
```

Now, you can save and close this file to open the usbd_cdc_if.c. In this file, declare the buffer variables:

```
/* Create buffer for reception and transmission           */
/* It's up to user to redefine and/or remove those define */
/** Received data over USB are stored in this buffer      */
uint8_t UserRxBufferFS[APP_RX_DATA_SIZE];
/** Data to send over USB CDC are stored in this buffer   */
uint8_t UserTxBufferFS[APP_TX_DATA_SIZE];
extern USBD_HandleTypeDef hUsbDeviceFS;
```

Then populate the following functions as shown below:

```
static int8_t TEMPLATE_Init(void)
{
      USBD_CDC_SetTxBuffer(&hUsbDeviceFS, UserTxBufferFS, 0);
      USBD_CDC_SetRxBuffer(&hUsbDeviceFS, UserRxBufferFS);
      return (0);
}

static int8_t TEMPLATE_Receive(uint8_t *Buf, uint32_t *Len)
{
      USBD_CDC_ReceivePacket(&hUsbDeviceFS);
      return (USBD_OK);
}
```

Finally, you can add the transmit function as:

```
uint8_t TEMPLATE_Transmit(uint8_t* Buf, uint16_t Len)
{
  uint8_t result = USBD_OK;
  USBD_CDC_HandleTypeDef *hcdc = (USBD_CDC_HandleTypeDef*)hUsbDeviceF
  if (hcdc->TxState != 0){
    return
             USBD_BUSY;
  }
  USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
  result = USBD_CDC_TransmitPacket(&hUsbDeviceFS);
  return result;
}
```

There are no restrictions to change the function names and remove the TEMPLATE portion of it, but this action is up to the developer. You can save and close this file, and move to the main.c file. This article uses the comment section's as guidance for the needed code to be added.

In this file, include the following headers:

```
/* Private includes ----------------------------------------------
/* USER CODE BEGIN Includes */
#include "usbd_core.h"
#include "usbd_cdc_if.h"
/* USER CODE END Includes */
```

Then, add these variables:

```
/* USER CODE BEGIN PV */
USBD_HandleTypeDef hUsbDeviceFS;
extern USBD_DescriptorsTypeDef Class_Desc;
/* USER CODE END PV */
```

And finally, add the following code into the MX_USB_PCD_Init function (note that each code has a user code section):

```
/* USER CODE BEGIN USB_Init 0 */
hpcd_USB_DRD_FS.pData = &hUsbDeviceFS;
/* USER CODE END USB_Init 0 */

/* USER CODE BEGIN USB_Init 2 */
if(USBD_Init(&hUsbDeviceFS, &Class_Desc, 0) != USBD_OK)
      Error_Handler();

if (USBD_RegisterClass(&hUsbDeviceFS, &USBD_CDC) != USBD_OK)
      Error_Handler();

if(USBD_CDC_RegisterInterface(&hUsbDeviceFS, &USBD_CDC_Template_fops)
      Error_Handler();

if(USBD_Start(&hUsbDeviceFS) != USBD_OK)
      Error_Handler();
/* USER CODE END USB_Init 2 */
```

Now, all the needed implementations are done. The next steps are just to create an example to check the middleware functionality.

To check the functionality, add the following code into the main.c file:

```
/* USER CODE BEGIN 1 */
uint8_t TxMessageBuffer[] = "MY USB IS WORKING! \r\n";
/* USER CODE END 1 */

/* USER CODE BEGIN 2 */
while(hUsbDeviceFS.pClassData == NULL);
/* USER CODE END 2 */

/* Infinite loop */
/* Infinite loop */
/* USER CODE BEGIN WHILE */

while (1)
{
      TEMPLATE_Transmit(TxMessageBuffer, sizeof(TxMessageBuffer));
      HAL_Delay(500);
   /* USER CODE END WHILE */

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

In the USB/Class/CDC/Src/usbd_cdc_if.c file, add the following code:

```
static int8_t TEMPLATE_Receive(uint8_t *Buf, uint32_t *Len)
{
      if(Buf[0] == '1')
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
      else if(Buf[0] == '0')
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
      USBD_CDC_ReceivePacket(&hUsbDeviceFS);
      return (USBD_OK);
}
```

Congratulations on reaching here! Now, you have all the necessary settings and demo code to test the example we have presented here. Now, let us see the results!

2. Results

The implemented code shows a basic transmit and receive data through the USB Virtual COM Port. At the beginning the TxMessageBuffer is created and filled with a message to be transmitted.

```
/* USER CODE BEGIN 1 */
uint8_t TxMessageBuffer[] = "MY USB IS WORKING! \r\n";
/* USER CODE END 1 */
```

Feedback

After that, we wait for the USB to be connected and set, polling the pClassData into the

hUsbDeviceFS.

```
/* USER CODE BEGIN 2 */
while(hUsbDeviceFS.pClassData == NULL);
/* USER CODE END 2 */
```

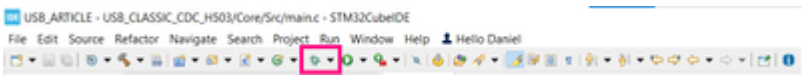
Once the USB is connected and set, we start to transmit the message on every 500 ms, as the code below:

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
      TEMPLATE_Transmit(TxMessageBuffer, sizeof(TxMessageBuffer));
      HAL_Delay(500);
/* USER CODE END WHILE */

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

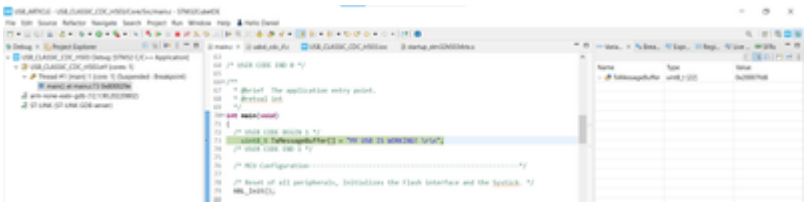
Finally, we use the received data to control the status of the User LED available in the Nucleo board, connected to the PA5, according to its User Manual.

```
static int8_t TEMPLATE_Receive(uint8_t *Buf, uint32_t *Len)
{
	if(Buf[0] == '1')
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
	else if(Buf[0] == '0')
		HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

	USBD_CDC_ReceivePacket(&hUsbDeviceFS);
	return (USBD_OK);
}
```

Now that you know how the demo works, let us try it. So, build (you should get zero errors and zero warnings) then click on the Debug button and wait for the debug session to be opened. Your STLINK might request to be updated. You can do so and then issue to debug again once it is completed.



Wait until the code reaches the breakpoint in the first line of the main function:

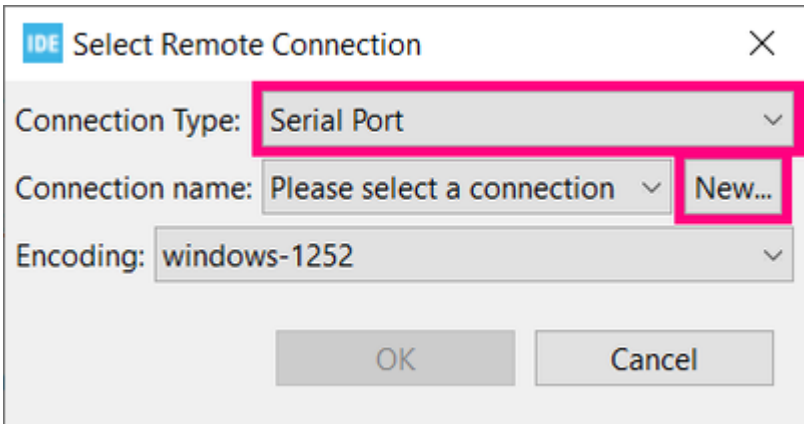


Then click on Resume, or press F8 to let the application start to run:

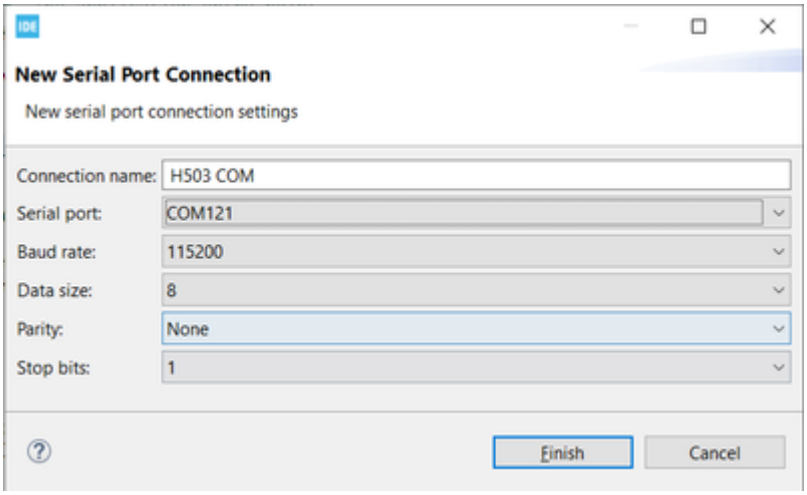
Feedback

Let us open a terminal in the STM32CubeIDE to see the application running. For that, click o New Shell console:

In the opened menu, select Serial Port for the Connection Type, then click on New…

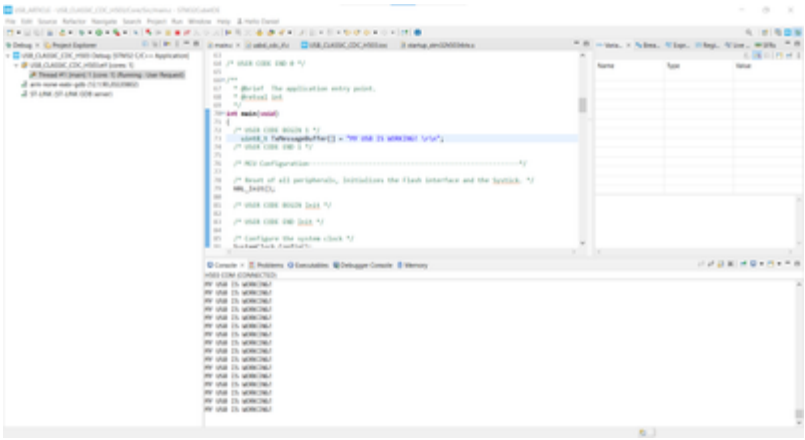


In the new menu, create a connection for your board. Make sure to select the proper Serial Port, then click Finish and finally OK:



Doing that, a console with a Virtual COM Port terminal opens and we can see our application running, printing the message on every 500 ms. If you type ‘1’ the User LED should be turned ON and ‘0’ will turn it OFF.

Feedback



3. Conclusion

That concludes our article. Now you have the needed information to implement the STMicroelectronics classic USB device middleware. The steps for doing that in another MCU Family or for other classes are very similar. If you face any difficulties while developing, refer to our demonstration projects available in our GitHub or contact us through the Community or by the On-line Support channel.

We hope the this article was helpful!

4. Relevant links:

Here are some relevant links that can help you in your developments using our ST peripheral:

ST Wiki - Introduction to USB with STM32

MOOC - STM32 USB Training

GitHub - STMicroelectronics - Middleware USB Device

GitHub - STMicroelectronics - Middleware USB Host

STMicroelectronics · GitHub

NUCLEO-H503RB User Manual

STM32H503RB - High-performance, Arm Cortex-M33, MCU with 128-Kbyte Flash, 32-Kbyte RAM, 250 MHz CPU - STMicroelectronics

Embedded Software **STM32 MCU Products** **STM32CubeIDE**

3 Kudos

COMMENTS

gbm
Lead II

2023-10-26 09:59 AM

1. The USB_CDC_ReceivePacket() call will trigger reception of a single USB packet of up to 64 bytes, There is no point in declaring the input buffer of 512 bytes, as no more than 64 bytes will ever be used. And the transmit buffer will probably never be used at all.

2. Transmit function called from main() performs multiphase logic operations on endpoint control registers. Similar operations are performed in USB interrupt service routine. If USB interrupt is triggered and serviced during the execution of Transmit, it will lead to USB stack lockup because of incorrect setting of endpoint control register. This is an old, well-known failure in ST USB stack, present in it from the very beginning and never addressed/corrected by ST. The simplest way to prevent this failure is to call Transmit only from an ISR of the same priority as USB interrupt.

0 Kudos

Robmar
Senior III

2023-11-09 05:33 AM

The USB support seems worse than the older version, there seems to be no CdcVcp_CtrlLines_t structure for com port handshake line simulation anymore.

Is there a fix?

I really do not want to be forced to use Microsoft RTOS just for USB support.

Feedback

What's the best way to get USB audio and VCom on an H743 using ST libraries, or should we just give up and try TinyUSB?

1 Kudo

B.Montanari
ST Employee

2023-11-09 06:49 AM

Hi @Robmar ,

The STM32H743 series supports both stacks, USBX and our classic USB Middleware. They are both integrated in the STM32CubeMX and IDE, so nothing has changed for this series, the newer ones, such as the H5, are the ones that have changed this, where they only includes the USBX natively now and this article was meant to show how to use the now classic USB Middelware with them.

As for the control line support, you can manually add it into your code, same process as usual, in the usbd_cdc_if.c the condition CDC_SET_CONTROL_LINE_STATE: is present in the TEMPLATE_Control function, so you can add your code there, something like this:

```
case CDC_SET_CONTROL_LINE_STATE:
{
    // we get wValue here as buffer
    uint16_t wValue = *(uint16_t*)&(pbuf[0]);
    cdcvcp_ctrllines.dtr = (wValue & 0x01)?1:0;
    cdcvcp_ctrllines.rts = (wValue & 0x02)?1:0;
    break;
}
```

Assuming you created the structure, which can be done in the usbd_cdc_if.h:

Feedback

```
/* USER CODE BEGIN EXPORTED_TYPES */
 typedef struct {
     uint16_t
     dtr:1,
     rts:1;
 } CdcVcp_CtrlLines_t;

extern __IO CdcVcp_CtrlLines_t  cdcvcp_ctrllines;
/* USER CODE END EXPORTED_TYPES */
```

There is no need to use ThreadX (Microsoft's RTOS) for USBX support, we will launch an article showing how to use the USBX in bare metal mode soon, so stay tuned.

For Audio and VCOM, you need to implement a Composite class, we currently don't have any examples ready for this particular combination, but we do have for VCOM + HID using the classic USB stack. So my suggestion is to get familiarized with the composite class and audio examples and try to change the HID to the Audio. Assuming you have the HAL driver installed with the STM32CubeMX in your PC, the examples will be in your repository>

C:\Users\%username%\STM32Cube\Repository\STM32Cube_FW_H7_V1.11.1\Projects\STM32H743I-EVAL\Applications\USB_Device\DualCore_Standalone

and

C:\Users\%username%\STM32Cube\Repository\STM32Cube_FW_H7_V1.11.1\Projects\STM32H743I-EVAL\Applications\USB_Device\Audio_Standalone

Hope this was helpful, but in case you need more details, I do recommend issuing an on line support ticket (Online Support (st.com)), so one of our FAEs can help you addressing the particular implementation and specific questions.

0 Kudos

gbm
Lead II

2023-11-09 07:19 AM

Assuming that you created the structure as in the second code fragment above, the first one should rather look like:

```
case CDC_SET_CONTROL_LINE_STATE:
  {
      // we get wValue here as buffer
      uint16_t wValue = *(uint16_t*)&(pbuf[0]);
      *(uint16_t *)&cdcvcp_ctrllines = wValue;
      break;
  }
```

and assuming that the declarations for the USB control request packet and ControlLineState were just slightly smarter than they are in the current version of the USB stack,it could be just single, simple assignment without any typecasts. 😉

Unfortunately I believe that the code above is completely incorrect, since the ControlLineState is sent in the setup packet wValue field which is at offset 2 (NOT at offset 0) of a setup packet, and the code above suggests that it is received as data packet.

In my USB stack, I have this:

```
usbd->cdc_data[funidx].ControlLineState = req->wValue.w;
```


Robmar
Senior III

2023-11-09 08:18 AM

Thanks @B.Montanari for the response, I was told to use USBX we had to install MS RTOS, so I hope you are correct, that would be good news.

In our commercial product with F407 we have the composite audio and CDC (VCom port) working perfectly, but now that we have designed a new board with H743 find utter USB chaos, which has made our lives at work very stressful.

We are also very disappointed that in 2023 STM have lost composite devices and there is no UAC 2.0 for higher audio baud rates. How can this be possible, even the TinyUSB project supports UAC 2.0!

I can't tell you how unpleasant this has been, our shipment date to clients is wrecked, we will fail to deliver for Christmas holidays and a lot of clients will be disappointed.

How do you suggest we get audio and vcom port working on the H743VIT6 as quickly as possible, and is there a good reference manual for USB that we can use (without MS ROTS!)

1 Kudo

Feedback

Robmar
Senior III

2023-11-09 08:36 AM

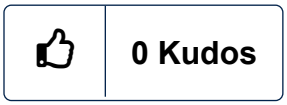
@B.Montanari I have looked at the DualCore example code you mention, it states in the readme file:-

"This is a typical application on how to use the STM32H7xx USB OTG Device peripheral,

where STM32 is
enumerated as a CDC device in the High Speed mode, and also as a HID device in the Full Speed mode,
using the native PC Host HID/CDC drivers to which the STM32H743I-EVAL board is connected."

How does this impact our H743VIT6 board which does not implement the faster (full speed) USB mode? This sample is made for high speed and the HID device on full speed, how can we test that on our device, and will this even work on the H743VIT6?

Our original F407 composite USB code, we did try to drop that in but there seem to be differences in the H743 HAL. Do you know if we can easily resolve those changes, i.e. fastest path to composite is to upgrade our F407 USB code?



B.Montanari
ST Employee

2023-11-09 09:07 AM

Hi @Robmar ,

Let me break down the points:

Control Line support: I believe you just misinterpreted this, Ripa and I said the same thing, it is not supported by default, but you can manually add it into your code. @gbm even made his own proposition on how to do so, thanks btw.

USBX in Stand Alone Mode: this capability was added in the December's 2022 release as part of the X-CUBE-AZRTOS-H7 package release, if you go to the STMicroelectronics/x-cube-azrtos-h7: X-CUBE-AZRTOS-H7 (Azure RTOS Software Expansion for STM32Cube) provides a full integration of Microsoft Azure RTOS in the STM32Cube environment for the STM32H7 series of microcontrollers. (github.com), you can see the examples currently available in bare metal mode.

You have already an FAE assigned to your case, so my suggestion is to continue the

conversation via the OLS platform to solve the Composite implementation.

0 Kudos

Robmar
Senior III

2023-11-09 09:17 AM

I've been communicating via OLS about the lack of USB support since July 30, that's three and a half months now, progress is glacially slow, we have a business to run, frankly dealing with ST is the road to failure.

I am very not impressed.

1 Kudo

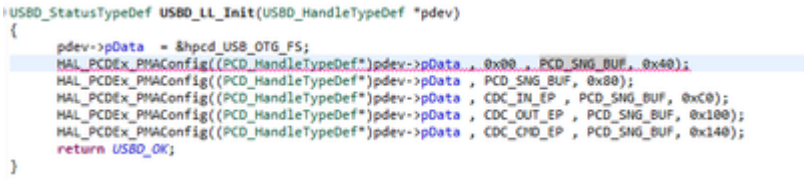
AkaPaDa_18
Associate III

2023-11-14 08:06 AM

Hello @B.Montanari

Thank you for this article,
I'm trying to do the same using the STWIN.box (STM32U585) but I'm getting errors in "usbdd_conf.c" as shown in the picture.
error displayed: 'PCD_SNG_BUF' undeclared (first use in this function)

What's the problem? Thanks in advance



```
USBD_StatusTypeDef USBD_LL_Init(USBD_HandleTypeDef *pdev)
{
      pdev->pData  = &hpcd_USB_OTG_FS;
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , 0x00 , PCD_SNG_BUF, 0x40);
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , PCD_SNG_BUF, 0x80);
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_IN_EP , PCD_SNG_BUF, 0xC0);
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_OUT_EP , PCD_SNG_BUF, 0x100);
      HAL_PCDEx_PMAConfig((PCD_HandleTypeDef*)pdev->pData , CDC_CMD_EP , PCD_SNG_BUF, 0x140);
      return USBD_OK;
}
```

Feedback

1 Kudo

Andy Tsai
Associate II

2023-11-17 06:01 PM

Hi @B.Montanari

Thank you for your help.

I have created a nucleo-h503 project and follow your descriptions, then every thing compiled/burn OK, but when I plug it into a win11 machine, it shows unknown device. I have checked again and nothing wrong. Could you create you project in Github and share it for comparison ? it will help a lot, thank you !

1 Kudo

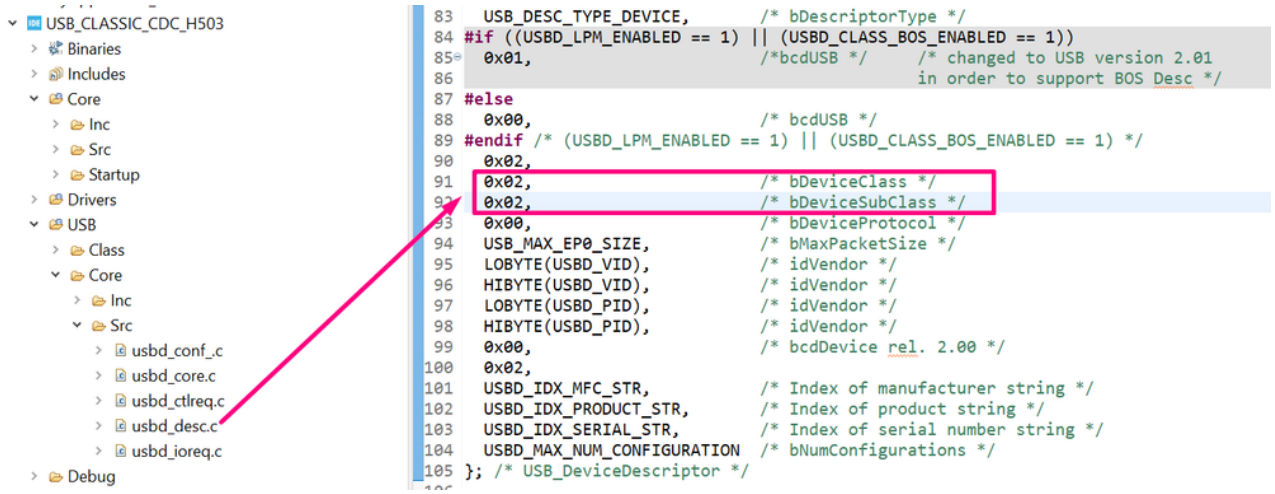
B.Montanari
ST Employee

2023-11-21 09:55 AM

Hi @Andy Tsai ,

I failed to add one step to allow the Windows recognize the driver properly. I'll update the article, but if you go to the usbd_desc.c, please change the descriptor to be like this:

Feedback



I've added the code in my personal github as well, but I'll upload this to our hotspot as soon as possible.

Again, sorry for the missing step and thanks for reporting the problem.

Best Regards

0 Kudos

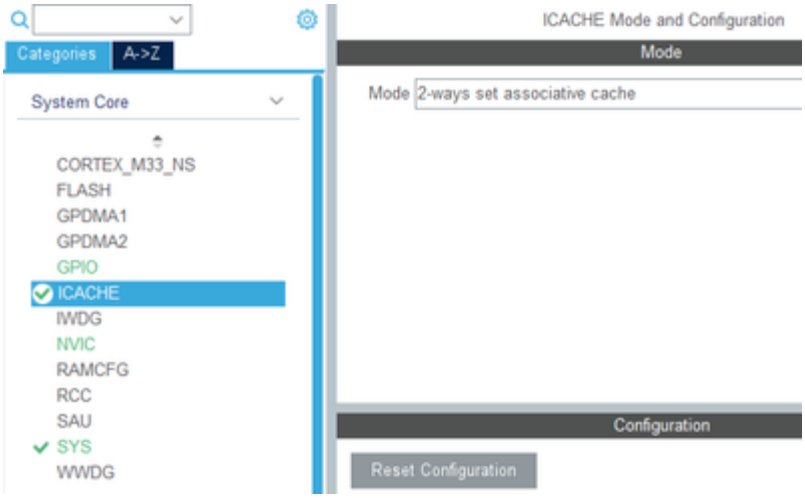
Andy Tsai
Associate II

2023-11-23 04:16 PM

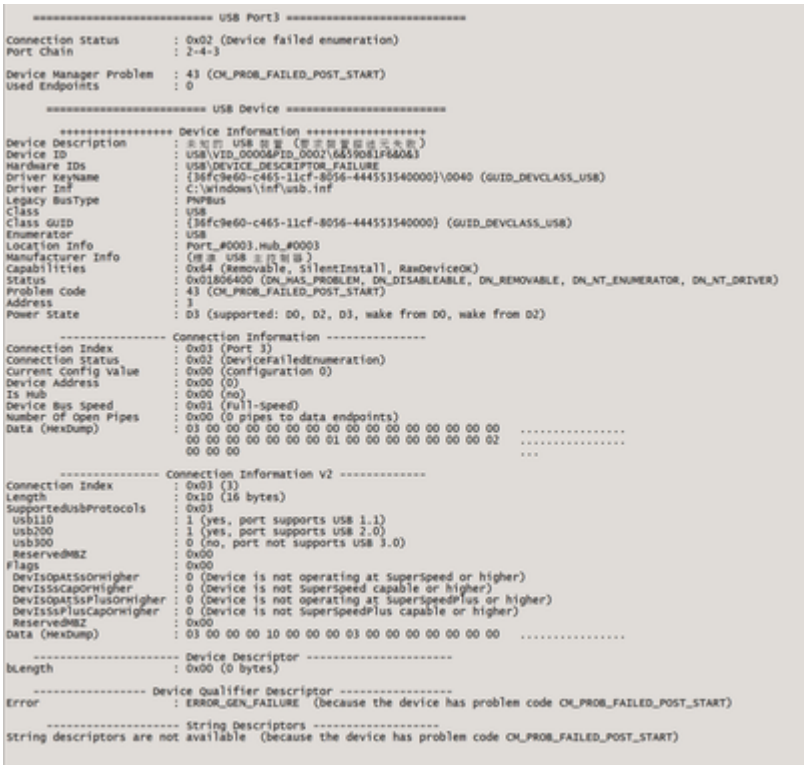
Hi @B.Montanari ,

Thank you for your code. I clone your project and it runs perfectly. Then I try to figure out why my code has issues there. I modify the bDeviceClass & bDeviceSubClass as you mentioned. But it still not work. After many tries, finally I found it's the ICACHE issue.

Feedback



In your code, if I enable the ICACHE in CubeMX and re-generate the project, the USB device enumeration will be failed. Using "USB Device Tree Viewer.exe V3.8.6", we can see:



```
    ============================ USB Port3 ============================

Connection Status        : 0x02 (Device failed enumeration)
Port Chain               : 2-4-3

Device Manager Problem   : 43 (CM_PROB_FAILED_POST_START)
Used Endpoints           : 0

      ======================== USB Device ========================

         ++++++++++++++++++ Device Information ++++++++++++++++++
Device Description       : ▒▒▒▒ USB ▒▒▒▒ (▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒)
Device ID                : USB\VID_0000&PID_0002\6&39081F6&0&3
Hardware IDs             : USB\DEVICE_DESCRIPTOR_FAILURE
Driver KeyName           : {36fc9e60-c465-11cf-8056-444553540000}\0040 (GUID_DEVCLASS_USB)
Driver Inf               : C:\Windows\inf\usb.inf
Legacy BusType           : PNPBus
Class                    : USB
Class GUID               : {36fc9e60-c465-11cf-8056-444553540000} (GUID_DEVCLASS_USB)
Enumerator               : USB
Location Info            : Port_#0003.Hub_#0003
Manufacturer Info        : (▒▒▒▒ USB ▒▒▒▒▒▒▒▒)
Capabilities             : 0xb4 (Removable, SilentInstall, RawDeviceOK)
Status                   : 0x01806400 (DN_HAS_PROBLEM, DN_DISABLEABLE, DN_REMOVABLE, DN_NT_ENUMERATOR, DN_NT_DRIVER)
Problem Code             : 43 (CM_PROB_FAILED_POST_START)
Address                  : 3
Power State              : D3 (supported: D0, D2, D3, wake from D0, wake from D2)

         ---------------- Connection Information ----------------
Connection Index         : 0x03 (Port 3)
Connection Status        : 0x02 (DeviceFailedEnumeration)
Current Config Value     : 0x00 (Configuration 0)
Device Address           : 0x00 (0)
Is Hub                   : 0x00 (no)
Device Bus Speed         : 0x01 (Full-Speed)
Number Of Open Pipes     : 0x00 (0 pipes to data endpoints)
Data (HexDump)           : 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
                           00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 02   ................
                           00 00 00                                         ...

         ---------------- Connection Information V2 -------------
Connection Index         : 0x03 (3)
Length                   : 0x10 (16 bytes)
SupportedUsbProtocols    : 0x03
 Usb110                  : 1 (yes, port supports USB 1.1)
 Usb200                  : 1 (yes, port supports USB 2.0)
 Usb300                  : 0 (no, port not supports USB 3.0)
 ReservedMBZ             : 0x00
Flags                    : 0x00
 DevIsOpAtSsOrHigher     : 0 (Device is not operating at SuperSpeed or higher)
 DevIsSsCapOrHigher      : 0 (Device is not SuperSpeed capable or higher)
 DevIsOpAtSsPlusOrHigher : 0 (Device is not operating at SuperSpeedPlus or higher)
 DevIsSsPlusCapOrHigher  : 0 (Device is not SuperSpeedPlus capable or higher)
 ReservedMBZ             : 0x00
Data (HexDump)           : 03 00 00 00 10 00 00 00 03 00 00 00 00 00 00 00   ................

    ---------------------- Device Descriptor ----------------------
bLength                  : 0x00 (0 bytes)

    ------------------ Device Qualifier Descriptor ------------------
Error                    : ERROR_GEN_FAILURE  (because the device has problem code CM_PROB_FAILED_POST_START)

      -------------------- String Descriptors --------------------
String descriptors are not available  (because the device has problem code CM_PROB_FAILED_POST_START)
```

Then if I disable the ICACHE and regenerate, and everything goes back.

My code was generated by CubeMX using nucleo-h503, the default MCU running under 250MHz and the ICACHE was default enabled, but your code was generate by CubeMX by bare metal STM32H503, it's 32MHz and ICACHE disabled, that's why my code always failed even followed your instructions.

Feedback

So my question is how to make the CDC run with the ICACHE enabled ?

1 Kudo

gbm
Lead II

2023-11-24 06:26 AM

It's not about ICACHE, it's probably about timing.

I am trying to port my USB device stack to H503, which is (when it comes to USB device) basically the same as G0B1. Similar problem, still no solution.

0 Kudos

Robmar
Senior III

2023-11-24 07:02 AM

We can't wait another decade for ST to not fix the USB situation, so has anyone tried TinyUSB, that even has UAC 2.0 support, and a larger following, and several people have integrated it into H7 MCUs.

1 Kudo

Feedback

Andy Tsai
Associate II

2023-11-24 04:33 PM

Using @B.Montanari 's project and enable the ICACHE, We get the code of main() in main.c:

```
86  /* Configure the system clock */
87  SystemClock_Config();
88
89  /* USER CODE BEGIN SysInit */
90
91  /* USER CODE END SysInit */
92
93  /* Initialize all configured peripherals */
94  MX_GPIO_Init();
95  MX_MEMORYMAP_Init();
96  MX_USB_PCD_Init();
97  // MX_ICACHE_Init();  // Original was here, it's failed.
98  /* USER CODE BEGIN 2 */
99  while(hUsbDeviceFS.pClassData == NULL);
100 /* USER CODE END 2 */
101 MX_ICACHE_Init();   // Now moved here, it's work.
102
103 /* Infinite loop */
104 /* USER CODE BEGIN WHILE */
105 while (1)
106 {
107   TEMPLATE_Transmit(TxMessageBuffer, sizeof(TxMessageBuffer));
108   HAL_Delay(500);
109   /* USER CODE END WHILE */
110
111   /* USER CODE BEGIN 3 */
112 }
113 /* USER CODE END 3 */
```

The MX_ICACHE_Init() generated by the CubeMX was in line#97, in this case, the USB enumeration failed. Now I move the line manually to line#101, then the USB works. I don't know why.

0 Kudos

Feedback

gbm
Lead II

2023-11-26 06:57 AM

Impressed by the discussion in this thread I did some experiments with H503 USB device at various HCLK frequencies. My code, same as STM32G0 USB code, works without

problem on H503 up to 196 MHz, with ICACHE on or off. At 240/250 MHz the code works ok if ICACHE is off. The USB enumeration/configuration fails with ICACHE on. I did some experiments with turning on the cache at various stages of USB configuration. If ICACHE is turned on after the payload data communication is established, everything works. If ICACHE is turned on during or at the end of configuration process (right after finalizing the endpoint configuration), it fails.

It looks like there is some undocumented timing quirk with the USB device on H503.

EsaT
Associate

2023-11-28 04:04 AM

Hello @B.Montanari

I am trying to do this with STM32U5 MCU.

1) First problem is with CubeMX. STM32U5 does not have USB peripheral, it has USB_OTG_FS.
--> I'll active it to Device_only mode

2) Next problem:
extern PCD_HandleTypeDef hpcd_USB_DRD_FS;
--> Generated code does not have hpcd_USB_DRD_FS but it has hpcd_USB_OTG_FS. Should that be used?

3) Main.c
STM32U5 version does not contain MX_USB_PCD_Init function.
--> Should I do similar changes to MX_USB_OTG_FS_PCD_Init function in usb_otg.c?

4) Compilation error:
USB/Core/Src/usbd_conf.c: In function 'USBD_LL_Init':
USB/Core/Src/usbd_conf.c:83:3: warning: implicit declaration of function 'HAL_PCDEx_PMAConfig'; did you mean 'HAL_PWREx_PVMConfig'? [-Wimplicit-

function-declaration]
83 | HAL_PCDEx_PMAConfig((PCD_HandleTypeDef *)pdev->pData, 0x00,
PCD_SNG_BUF, 0x40);
| ^~~~~~~~~~~~~~~~~~
| HAL_PWREx_PVMConfig
USB/Core/Src/usbd_conf.c:83:63: error: 'PCD_SNG_BUF' undeclared (first use in this function)
83 | HAL_PCDEx_PMAConfig((PCD_HandleTypeDef *)pdev->pData, 0x00,
PCD_SNG_BUF, 0x40);
| ^~~~~~~~~~

Both of those are behind USB_DRD_FS which is not defined with stm32u575xx.h.
--> How to continue?

5) Any estimates when you will launch article showing how to use the USBX in bare metal mode?

Robmar
Senior III

2023-11-28 04:15 AM

ST put near zero effort into supporting the USB driver, which hasn't really been changed since pre 2015.

The TinyUSB GitHub driver is far more advanced, supports composite devices and UAC 2.0 for fast audio streaming. It also has a very active group, and a number of users using ST processors.

ST support people have promised to have the USB driver updated by next year, but I wouldn't hold my breath.

0 Kudos

B.Montanari
ST Employee

2023-11-28 04:43 AM

Hi @EsaT,

Hope you are doing well.

I'm not sure which STM32U5 series you are using, but I believe it is the STM32U575 or U585, based on the USB OTG comment. The STM32U545 series has USB device only peripheral and the steps are similar to the ones shown in this article, but if my assumption is correct, then we have 2 USB Device classes examples available that you can use: HID and CDC> stm32u5-classic-coremw-apps/Projects/NUCLEO-U575ZI-Q/Applications /USB_Device at main · STMicroelectronics/stm32u5-classic-coremw-apps (github.com), they are based on the NUCLEO-U575ZI-Q board.

As for the USBX stand alone article, I'll push it to be released by next week.

Hope this was helpful.

Feedback

Best Regards

0 Kudos

EsaT
Associate

2023-11-28 04:54 AM

Thanks@B.Montanari for quick answer!

Your assumption is correct, I am using STM32U575.

Will USBX article work with STM32U575?

We are planning to use HID profile and FreeRTOS. Which stack should we use, old one or USBX?

0 Kudos

B.Montanari
ST Employee

2023-11-28 06:46 AM

Hi @EsaT ,

Given the selected RTOS, I'd suggest using FreeRTOS with classic USB MW.

The USBX article was written for the STM32H7 family, mostly because the stand alone USBX is currently available for a few series and the STM32U5 is not among them yet - I've made the comment due to the H7 specific question we had previously, but considering your chosen STM32, you have 3 options readily available:

1. Stand Alone classic USB

Feedback

2. FreeRTOS + classic USB

3. AzureRTOS (ThreadX) + USBX

Microsoft announced, just a week ago, that AzureRTOS and its modules went open source with Eclipse> Introducing Eclipse ThreadX | Life at Eclipse (eclipse-foundation.blog). I have yet to understand the specifics, since the Stand Alone version was made specifically for ST and the previous license would restrict us from doing the USBX + FreeRTOS combination. Regardless of that, this combination is not implemented in the STM32CubeMX/IDE, so I'd avoid using this path for now.

Best Regards

0 Kudos

EsaT
Associate

2023-12-07 12:38 AM

Hi @B.Montanari

I have merged https://github.com/STMicroelectronics/stm32u5-classic-coremw-apps/tree/main/Projects/NUCLEO-U575ZI-Q/Applications/USB_Device/HID_Standalone to Cube generated code to run in our board.

I have managed to compile and run everything, but Windows doesn't recognize my device. I assume there is something vital what I have missed or not merged correctly.

I have debugged that CAD_StateMachine_SNK stays in USBPD_CAD_STATE_DETACHED state.

UCPD1_IRQHandler is triggered once during USB initialization. But when I insert USB cable to Windows, so IRQs are triggered.

USB connections in our board are:
USBDM -> PA11
USBDP -> PA12

Feedback

VBUS -> PA9 (with 33k / 82k resistors)
Board is self powered, we need somekind of vbus detection. Right?
How should PA9 be configured?
Should we connect VBUS also to ADC like your NUCLEO144_Q?

0 Kudos

gbm
Lead II

2024-01-07 12:59 AM

H503 USB+cache problem solved:

The origin of the problem is H503 errata 2.15.1 related to USB peripheral. Incorrect Rx data size is read from the endpoint descriptor in PMA if it is read too early. Turning ICACHE on speeds up the code execution significantly and causes the RxSize read from PMA before it's update by the peripheral. The solution is to reorder the actions in Out endpoint handling ISR and/or add some delay before reading the Rx size from PMA. The current xxx_hal_pcd.c uses a delay loop which solves the problem. I applied something similar in my USB stack and now it works at 240 MHz with cache enabled. In my code, I moved reading the RxSize just before the data copy loop and got the correct operation with optimization turned off (-O0). The problem is also affected by compiler optimization settings - with higher optimization levels longer delay must be added since the code before reading the RxSize executes faster. The final solution required adding the delay loop to ensure the correct operation with -O2 and -O3 optimization.

1 Kudo

oeliks
Senior

2024-01-07 03:39 AM

Thanks 🙂

0 Kudos

randyr2024
Associate

2024-01-07 06:22 AM

B.Montanari thank you for creating this project.

I also got the error:

error displayed: 'PCD_SNG_BUF' undeclared (first use in this function)

I went through every step in your article.

I dont think it's repairable. I must be missing some secret setting somewhere. I notice the HALs dont even include USB in them in my working project.

Your steps totally locked my project from working, I was forced to restore from backup.

For now i have decided to skip the shiny nice USB-C connector on my U5A5ZJ and only use the 100baud microSD which works with Uart just fine. I have optimized communication with my robot main cpu and i am able to send 50 commands per second so i'm ok with that.

Feedback

Plus i noticed the azure rtos documentation is poor. When i enable USBx the "threads" folder gets immediately populated with over 165 files! most irrelevant stuff. and the usb driver stuff inside of USBx ... i spent some time trying to find it and it seems hidden in some abstraction layers or something (typical microsoft visual studio lack of documentation style)

0 Kudos

MKing
Associate III

2024-01-16 02:45 AM

This example will create a "camera interface" in the windows "device manager" by USB. It is only to have a basic project to make the enumeration. There is no camera connected on my STM32F103.

I use: STM32CubeIDE 1.14.0 CubeMX 6.10.0 STM32F103CBTx / USB FullSpeed

mkoenig334455/USB_Video_STM32F103CBTx: STM32F103CBTx example USB and VIDEO class (github.com)

0 Kudos

Feedback

JerryK
Associate

2024-01-16 02:06 PM

Great work @B.Montanari , I am porting to a H563 but I seem to be missing something

with the USBD_CDC_Init.

This function is where pClassData is assigned to a Non-NULL value, but it is never called from the initialization routines. This causes me to hang up at while(hUsbDeviceFS.pClassData == NULL); in main.c .

I realize USBD_CDC_Init is set up as a call back and is magically called somewhere in the depths of the init routines. I have set a breakpoint in USBD_CDC_Init and it is never hit.

Any suggestions? Help is appreciated.

EDIT: I am using the NUCLEO-H563ZI (Nucleo-144) as my hardware platform.

0 Kudos

NikhilP

Associate II

2024-01-24 11:51 AM

Hello @B.Montanari I have followed all the step you mentioned here for STM32H563 series controller my code is getting stuck at while(hUsbDeviceFS.pClassData == NULL); Statement. Basically i am trying to open the com port on my Laptop and then send and receive data, but even after the MX_USB_PCD_Init(); My Laptop is unable to detect the USB of Microcontroller Please suggest What might be Wrong. When i am using the Azure Rtos My Laptop is able to detect the ST Controller as shown in image below:

When i am following all the Steps here i am unable to detect the USB. Please Suggest some solution.

Feedback

0 Kudos

JerryK
Associate

2024-01-24 12:56 PM

@NikhilP and all,

The technical team helped me out with this very issue.....just a couple days ago. I hope it is okay for me to post the solution.

First, enable VDDUS in HAL_PCD_MspInit()

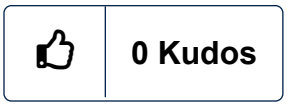
```
void HAL_PCD_MspInit(PCD_HandleTypeDef* hpcd)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
  if(hpcd->Instance==USB_DRD_FS)
  {
  /* USER CODE BEGIN USB_DRD_FS_MspInit 0 */
        HAL_PWREx_EnableVddUSB();   //<----------------------------------
  /* USER CODE END USB_DRD_FS_MspInit 0 */
```

Then in your main.c you will need to enable CRS (clock Recovery system)

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
```

Feedback

```
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  RCC_CRSInitTypeDef RCC_CRSInitStruct = {0};     //<--------------------------


  ...


  /** Enable the CRS APB clock */                 //<-------------
  __HAL_RCC_CRS_CLK_ENABLE();                     //<-------------


  /** Configures CRS  */                          //<-------------
  RCC_CRSInitStruct.Prescaler = RCC_CRS_SYNC_DIV1;            //<-------------
  RCC_CRSInitStruct.Source = RCC_CRS_SYNC_SOURCE_USB;         //<-------------
  RCC_CRSInitStruct.Polarity = RCC_CRS_SYNC_POLARITY_RISING;  //<-------------
  RCC_CRSInitStruct.ReloadValue =
__HAL_RCC_CRS_RELOADVALUE_CALCULATE(48000000,1000);  //<-----
  RCC_CRSInitStruct.ErrorLimitValue = 34;                     //<-------------
  RCC_CRSInitStruct.HSI48CalibrationValue = 32;               //<-------------


  HAL_RCCEx_CRSConfig(&RCC_CRSInitStruct);                    //<-------------
}
```

NikhilP
Associate II

2024-01-25 10:09 AM

Hello @JerryK I Tried Doing this but still the code gets stuck at the statement while(hUsbDeviceFS.pClassData == NULL);. I am unable to figure is going wrong with the code. Also after the MX_USB_PCD_Init(); function the USB Port should be detected by the Computer but it does not detect it. Also if i comment the while statement and execute the TEMPLATE_Transmit(TxMessageBuffer, sizeof(TxMessageBuffer)); Code The Code goes in Hardfault Handler.

0 Kudos

ACand.3
Associate II

2024-01-29 05:19 PM

hey Everyone,

I am also new to the Stm32U5 platform, I am currently using the stm32u545 nucleo board. And I also get stuck while(hUsbDeviceFS.pClassData == NULL) when I debug.

But if I flash the code without debugging, my terminal does say USB is working.

But as soon as I start debugging, I get a usb error on windows and the code gets stuck in that while loop

0 Kudos

Feedback

Greg Horler
Associate III

2024-02-01 03:57 AM

Hi there.

I have tried to use this tutorial to port the classic USB_CDC to a stm32h5xx, specifically a Nucleo H563Z1. I have this library on all my other work and I DO NOT use an RTOS.

I too get stuck in a hardfault error.

My code executes the following, but the USB is not enumerated, I would expect it to be.

```
MX_GPIO_Init();

MX_USB_PCD_Init();

//MX_ICACHE_Init();

/* USER CODE BEGIN 2 */

while(hUsbDeviceFS.pClassData == NULL);
```

The call to

```
TEMPLATE_Transmit(TxMessageBuffer, sizeof(TxMessageBuffer));
```

results in a hardfault error.

The tutorial is on balance well written, (thankyou @B.Montanari) but there are times when the instructions become vague. Would it be possible @B.Montanari to include the ammended files (six of them?) for your h503 example so that I and others can compare with our template edits, please.

It is clear from this post that there are lots of users who would like to migrate the classic USB_CDC library to newer devices, especially those with an M33 core. I don't understand why ST has not included this library in the release?

In the mean-time, if anyone gets this port working, please would you share the details. Many thanks

0 Kudos

JerryK
Associate

2024-02-01 10:20 AM

Attached is a working build for USB CDC for the NUCLEO-H563ZI.

It was built using STMCubeIDE version 1.14.1

I used terraterm to connect to the USB on the NUCLEO-144 PCB.

I have also included the CUBEMX .ioc file.

This is provided without warrantee or guarantee. Do your own due diligence. Don't expect the edited code to fall within the boundaries that CUBEMX requires for updatable code.

Cheers!

J

USB_Only_for_Nucleo144_v0001.7z

1 Kudo

Greg Horler
Associate III

2024-02-02 04:17 AM

@JerryK Many thanks for providing this project, I am pleased to confirm it works as per the tutorial of @B.Montanari .

I will now compare my attempt with yours to try and establish where I have gone wrong.

Thanks again😁

0 Kudos

NikhilP
Associate II

2024-02-02 04:30 PM

@JerryK I Tried the code and it was working Properly. Thanks.

0 Kudos

NikhilP
Associate II

2024-02-08 01:19 PM

@B.Montanari @JerryK Can you explain how does
this while(hUsbDeviceFS.pClassData == NULL); line work

I wanted to Understand Exactly how is it filled . I wanted to Know what exactly triggers the initialization of this variable and how it is filled.

0 Kudos

Feedback

NikhilP
Associate II

2024-02-16 10:46 AM

@B.Montanari @JerryK Can you also post the implementation of USB host in Similar way?. There is no documentation available which tells how to implement USB host with this Classic Library which you have used.

1 Kudo

Robmar
Senior III

2024-02-17 02:41 AM

Isn't it appalling that STM don't have a working composite USB driver and examples for their MCUs!

Developers have been asking for this for over a decade!

Cédric the ST Paris manager promised me their would be a new driver in Jan 2024, and only silence!

1 Kudo

Feedback

EL_HACHIMI
Associate II

2024-02-18 03:32 PM

Hello,

I tried the same procedure on my custom made Board based on STM32F303ZET, but I

get this message while trying to debug.
Please can someone explain to me what does this mean ?

0 Kudos

v_dimitrov
Associate

2024-02-20 01:59 AM

@gbm Would you elaborate (or even better - provide example code) where you inserted the delay loop? I've tried doing it before

```
1 | hcdc->RxLength = USBD_LL_GetRxDataSize(pdev, epnum);
```

in the USBD_CDC_DataOut function (usdb_cdc.c file), but to no avail.

@JerryK , @NikhilP , @ACand.3 , What you're fighting is exactly the bug gbm is speaking about, although JerryK circumvented the problem just by disabling the ICache resulting in slower code execution.
I implemented a delay (needs >= 8 ms with HAL) before the ICache init and placed it after the
hUsbDeviceFS.pClassData == NULL line which also solves the problem, but with the ICache working properly.

Feedback

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USB_PCD_Init();
/* USER CODE BEGIN 2 */
while(hUsbDeviceFS.pClassData == NULL);
/* USER CODE END 2 */
HAL_Delay(8); // Minimum needed delay
MX_ICACHE_Init();
```

Thanks again to @gbm for the troubleshooting and tracing the problem to the errata. Surely saved me lots of trouble and probably weeks worth of time.

1 Kudo

gbm
Lead II

2024-02-20 12:34 PM

I am not sure which delay you refer to. The one mentioned in errata is needed in low-level routines. Detecting the CDC VCP connection is a different story.

The delay loop fixing the rxcount problem is already present in ST PCD_GET_EP_RX_CNT inline function, defined in stm32h5xx_hal_pcd.h file (same for U5).

In my USB stack I solved the RX count problem in a different way, using the value of 1023 to initialize the rxcount field of buffer descriptor, then waiting for the value to be not equal to 1023. In theory at least, however, 1023 might be a valid data size for isochronous endpoint.

My USB stuff is here:

Feedback

https://github.com/gbm-ii/gbmUSBdevice

You may also look at my method of detecting the VCP connection. It is still "work in progress" so some things need to be fixed but it works reliably for me.

Version history

Last update:
2024-01-03 09:04 AM

Updated by:
ADMIN **Laurids_PETERSEN**

Contributors

 **D.Botelho**

 **B.Montanari**

 **Laurids_PETERSEN**

Feedback

life.augmented

About STMicroelectronics

Who we are

Investor relations

Sustainability

Innovation & technology

Careers

Blog

Compliance, ethics & privacy

Ethics and compliance

ST ethics hotline

Privacy portal

Follow us

Connect with us

Contact ST offices

Find sales offices & distributors

Community

Newsroom

Events & trainings

Browse

Shortcuts

Sitemap

Feedback