Data exchange between STM32F411 and PC using USB

Note for using **CDC**(communication device class) or **VCP**(Virtual Com Port) to **exchange data between STM32 and PC via USB**.

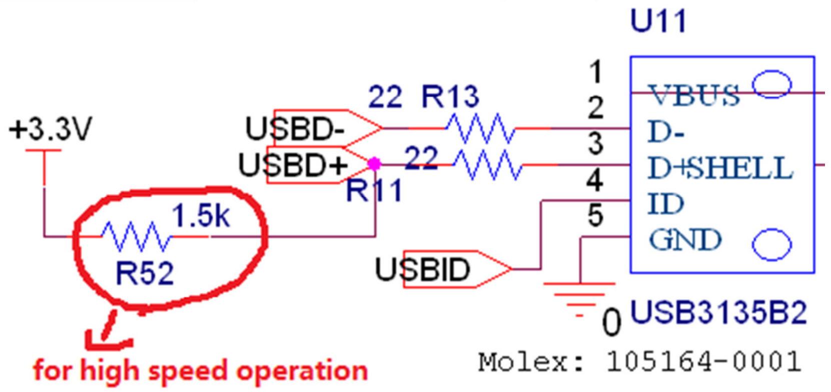
The sample code and related tools can be downloaded via:
https://bitbucket.org/rwmao/cdc_onstm32f411rc/downloads
https://bitbucket.org/rwmao/cdc_onstm32f411rc/src it is a repository. There is wiki too.

If you are not patient to read through, you can directly go to the last part:
Bugs or possible error you may face.

Development tools:
Keil 5.16a, FDP2.6 pack.
Cubemx v1.8pack.
Chip: STM32F411RC 64pins.

Implementation of hardware:

On STM32 side, USB must be correctly configured.



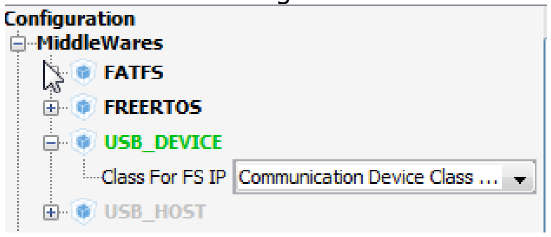
The d- and d+ datalines are simply connected to the STM32F411RC micro-controller (PA11 and PA12).
One of the most important thing is the 1.5k pull-up resistor to identify the devices itself as high speed USB.
Without the resistor, PC may not be able to detect the device at all.

On PC side, no need for any hardware implementation as long as USB port is available.
VCP driver is needed for PC to correctly access the data.
http://www.st.com/web/en/catalog/tools/PF257938
You also need a software to monitor the data sent through USB (virtual com port). The software can be found in the repository.

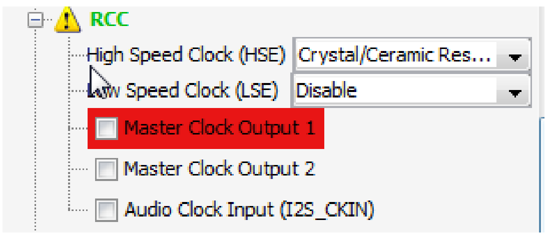
Software implementation.

1. Use STM32F4cubemx to generate the project.

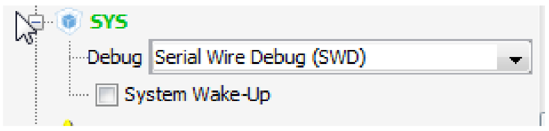
Not much to configure. Only select USB and I2C. Note I2C is not necessary for VCP. I used it for something else.



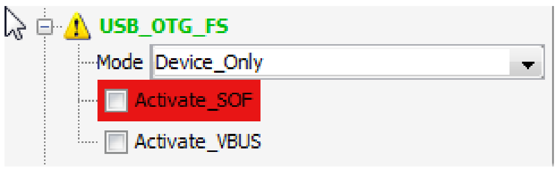
Configuration
- MiddleWares
  - FATFS
  - FREERTOS
  - USB_DEVICE
    - Class For FS IP: Communication Device Class ...
  - USB_HOST



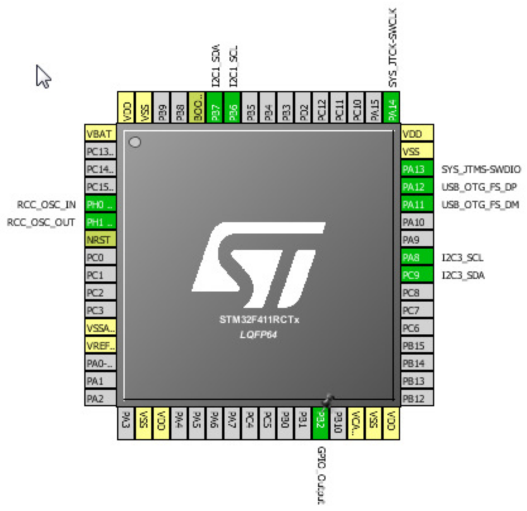
- RCC
  - High Speed Clock (HSE): Crystal/Ceramic Res...
  - Low Speed Clock (LSE): Disable
  - ☐ Master Clock Output 1
  - ☐ Master Clock Output 2
  - ☐ Audio Clock Input (I2S_CKIN)

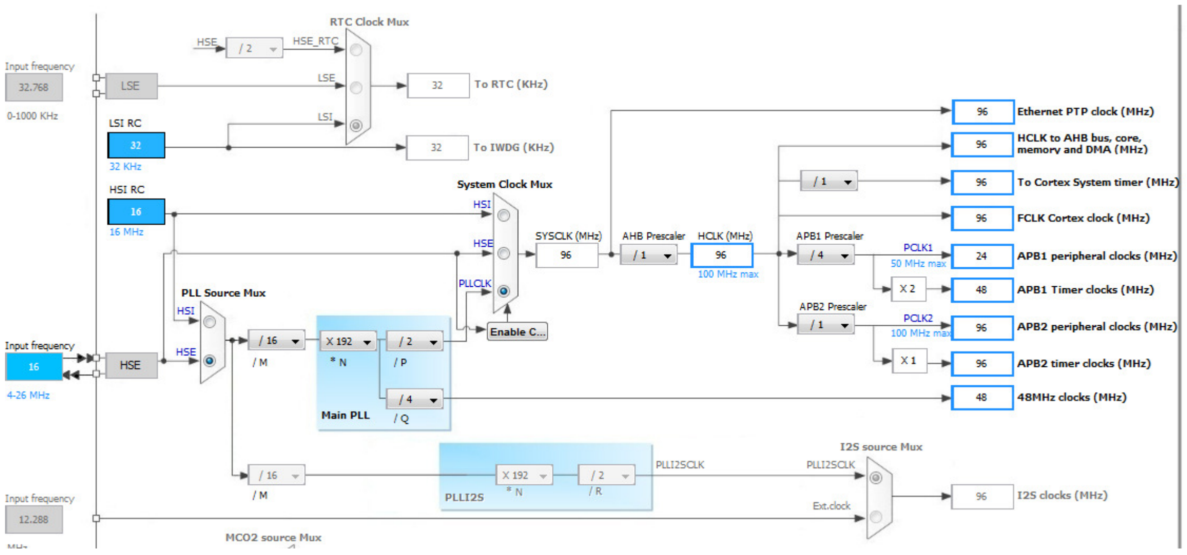


- SYS
  - Debug: Serial Wire Debug (SWD)
  - ☐ System Wake-Up



- USB_OTG_FS
  - Mode: Device_Only
  - ☐ Activate_SOF
  - ☐ Activate_VBUS





2

2. Implement the USB TX function.

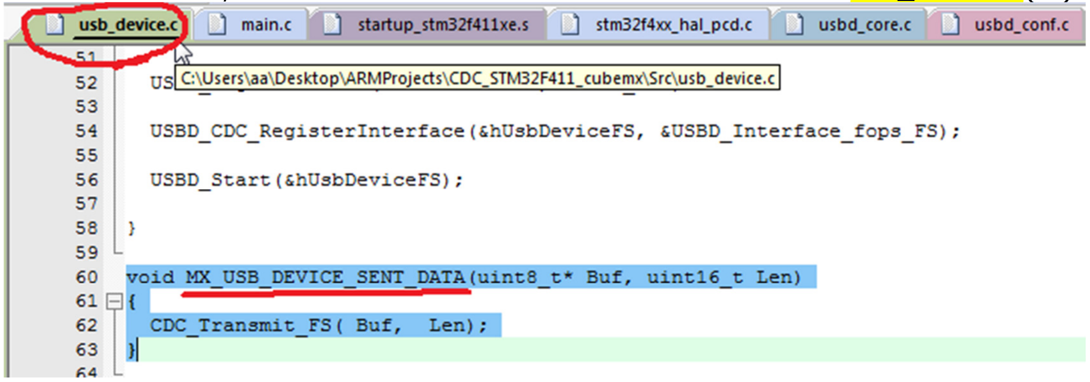
(1). Implement the data sending function, `CDC_Transmit_FS`. in file:**usbd_cdc_if.c**

```
/**

  * @brief  CDC_Transmit_FS

  *         Data send over USB IN endpoint are sent over CDC interface

  *         through this function.

  *         @note

  * @param  Buf: Buffer of data to be send

  * @param  Len: Number of data to be send (in bytes)

  * @retval Result of the operation: USBD_OK if all operations are OK
else USBD_FAIL or USBD_BUSY

  */

uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)

{

      //I revised the code to send long strings exceeding
APP_TX_DATA_SIZE. rwmao


  uint8_t result = USBD_OK;

  if ( hUsbDevice_0 == NULL ) return USBD_FAIL;

  USBD_CDC_HandleTypeDef *pCDC =

          (USBD_CDC_HandleTypeDef *)hUsbDevice_0->pClassData;


  if ( pCDC->TxState != 0 ) return USBD_BUSY;


   /* USER CODE BEGIN 8 */

      if (Len > APP_TX_DATA_SIZE)

      {

            int offset;
		for (offset = 0; offset < Len; offset++)
		{
			int todo = MIN(APP_TX_DATA_SIZE,
						Len - offset);
			result = CDC_Transmit_FS(Buf + offset, todo);
			if ( ( result != USBD_OK ) && ( result != USBD_BUSY )
) {
				/* Error:  Break out now */
				return result;
			}
		}
		return USBD_OK;
	}

	pCDC = (USBD_CDC_HandleTypeDef *)hUsbDevice_0->pClassData;
	/* TODO:  Consider a timeout in the following wait loop. */
	while(pCDC->TxState) { } //Wait for previous transfer to complete

	int i;
	for ( i = 0; i < Len; i++ ) {
		UserTxBufferFS[i] =  Buf[i];
	}
	USBD_CDC_SetTxBuffer(hUsbDevice_0, &UserTxBufferFS[0], Len);
	result = USBD_CDC_TransmitPacket(hUsbDevice_0);

	/* USER CODE END 8 */
	return result;
}
```

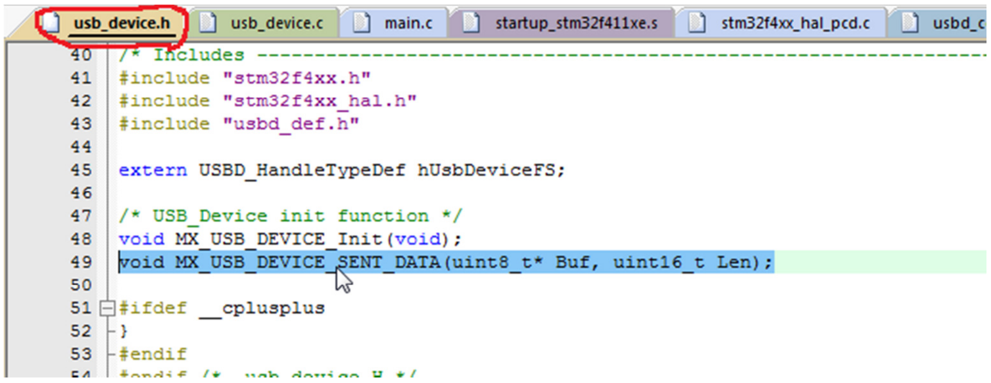
You can change the buffer size too.

```
  /* USER CODE BEGIN 1 */
/* Define size for the receive and transmit buffer over CDC */
/* It's up to user to redefine and/or remove those define */
#define APP_RX_DATA_SIZE  64
#define APP_TX_DATA_SIZE  64
  /* USER CODE END 1 */
```

Now the function to send data is ready.
For convenience, we define a function to call the subroutine in usb_device.c(.h).

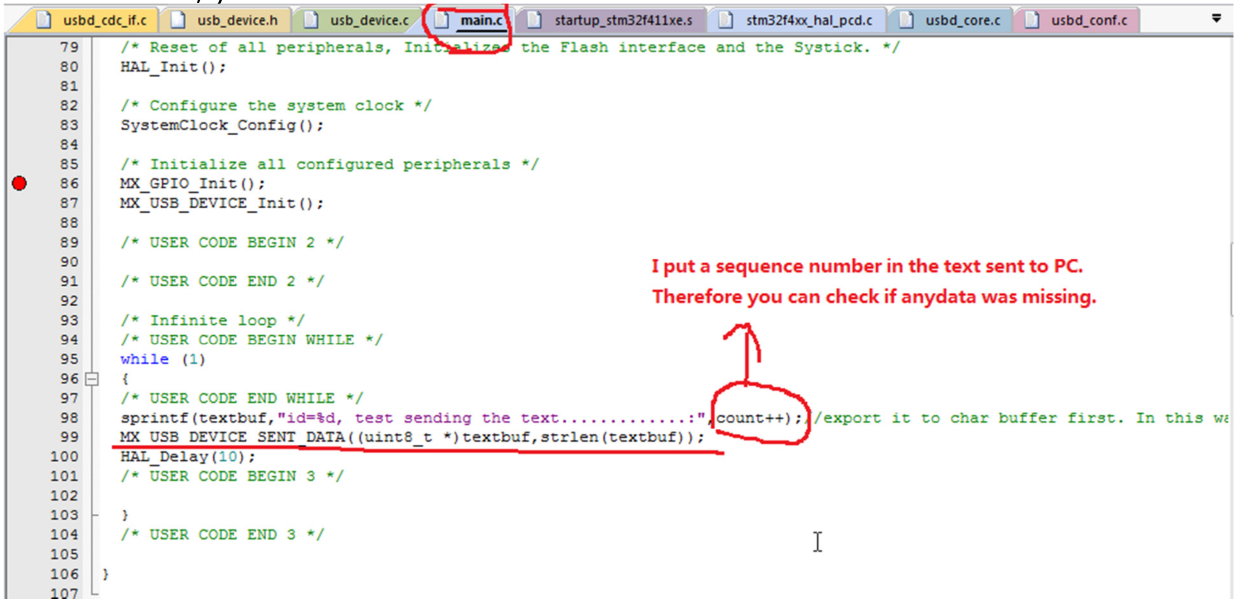


```
  US C:\Users\aa\Desktop\ARMProjects\CDC_STM32F411_cubemx\Src\usb_device.c

  USBD_CDC_RegisterInterface(&hUsbDeviceFS, &USBD_Interface_fops_FS);

  USBD_Start(&hUsbDeviceFS);

}

void MX_USB_DEVICE_SENT_DATA(uint8_t* Buf, uint16_t Len)
{
  CDC_Transmit_FS( Buf,  Len);
}
```



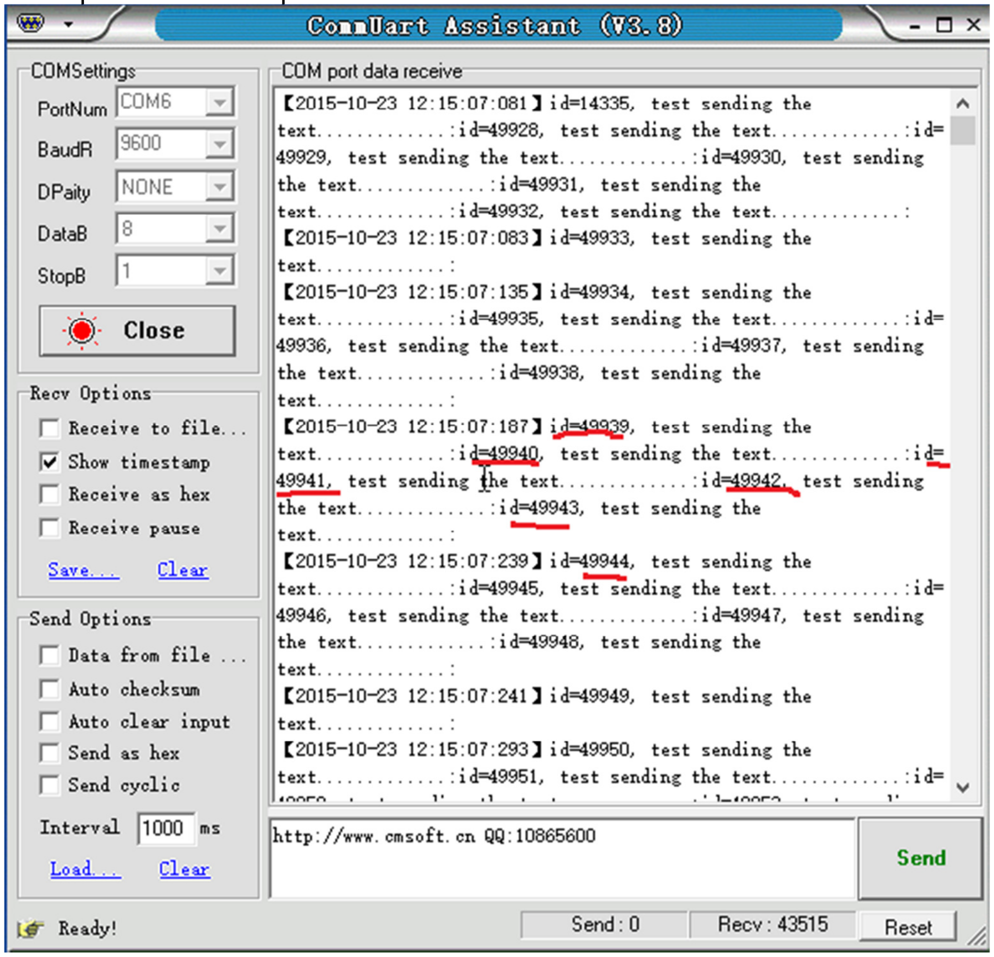
```
/* Includes ------------------------------------------------------------------
#include "stm32f4xx.h"
#include "stm32f4xx_hal.h"
#include "usbd_def.h"

extern USBD_HandleTypeDef hUsbDeviceFS;

/* USB_Device init function */
void MX_USB_DEVICE_Init(void);
void MX_USB_DEVICE_SENT_DATA(uint8_t* Buf, uint16_t Len);

#ifdef __cplusplus
}
#endif
```

Now in main.c, you can call the subroutine to send data.



```
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USB_DEVICE_Init();

  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  sprintf(textbuf,"id=%d, test sending the text............:",count++);//export it to char buffer first. In this wa
  MX_USB_DEVICE_SENT_DATA((uint8_t *)textbuf,strlen(textbuf));
  HAL_Delay(10);
  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}
```

A snapshot of the captured data is:


ComUart Assistant (V3.8)

COMSettings
PortNum COM6
BaudR 9600
DPaity NONE
DataB 8
StopB 1
Close

Recv Options
- [ ] Receive to file...
- [x] Show timestamp
- [ ] Receive as hex
- [ ] Receive pause

Save... Clear

Send Options
- [ ] Data from file ...
- [ ] Auto checksum
- [ ] Auto clear input
- [ ] Send as hex
- [ ] Send cyclic

Interval 1000 ms

Load... Clear

COM port data receive

```
【2015-10-23 12:15:07:081】id=14335, test sending the
text............:id=49928, test sending the text............:id=
49929, test sending the text............:id=49930, test sending
the text............:id=49931, test sending the
text............:id=49932, test sending the text............:
【2015-10-23 12:15:07:083】id=49933, test sending the
text............:
【2015-10-23 12:15:07:135】id=49934, test sending the
text............:id=49935, test sending the text............:id=
49936, test sending the text............:id=49937, test sending
the text............:id=49938, test sending the
text............:
【2015-10-23 12:15:07:187】id=49939, test sending the
text............:id=49940, test sending the text............:id=
49941, test sending the text............:id=49942, test sending
the text............:id=49943, test sending the
text............:
【2015-10-23 12:15:07:239】id=49944, test sending the
text............:id=49945, test sending the text............:id=
49946, test sending the text............:id=49947, test sending
the text............:id=49948, test sending the
text............:
【2015-10-23 12:15:07:241】id=49949, test sending the
text............:
【2015-10-23 12:15:07:293】id=49950, test sending the
text............:id=49951, test sending the text............:id=
```

http://www.cmsoft.cn QQ:10865600

Send

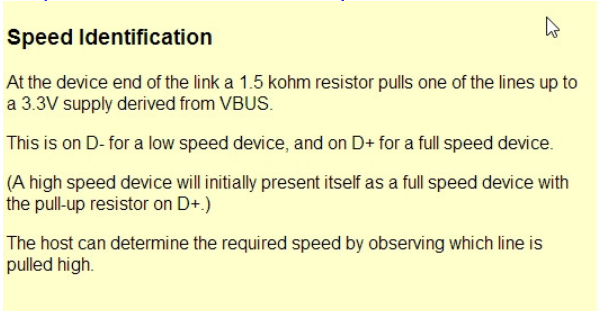
Ready! Send : 0 Recv : 43515 Reset

----

3. Bugs or possible error you may face.

(1). PC doesn't response at all when you plug in the usb cable.

This is the most headache you may face. There are lots of possibility.
One of them is the missing pull-up resistor. Look at the first page. You may need to connect a 1.5kohm pull up resistor.
The idea is the device needs to be correctly identified.
http://www.usbmadesimple.co.uk/ums_3.htm

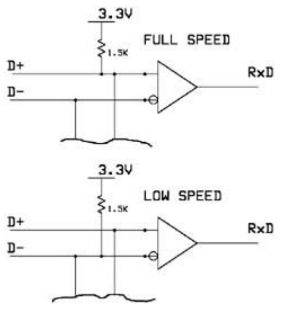

Speed Identification

At the device end of the link a 1.5 kohm resistor pulls one of the lines up to a 3.3V supply derived from VBUS.

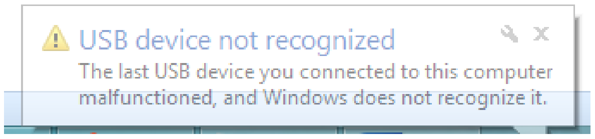
This is on D- for a low speed device, and on D+ for a full speed device.

(A high speed device will initially present itself as a full speed device with the pull-up resistor on D+.)

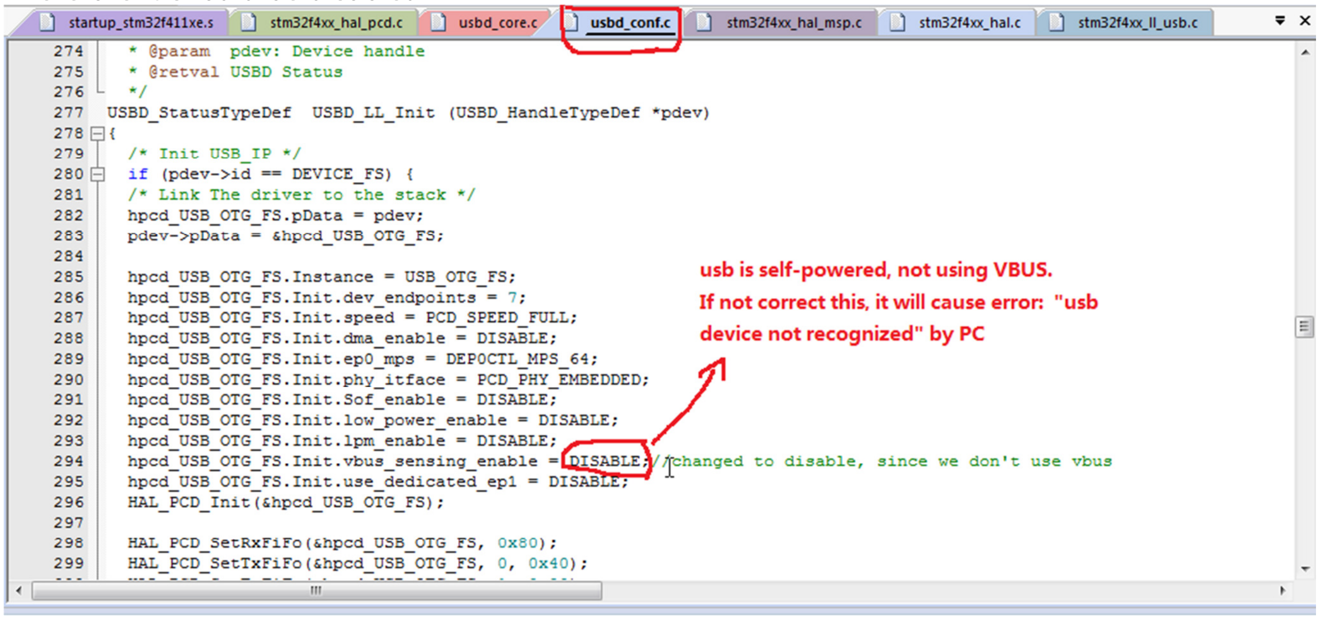
The host can determine the required speed by observing which line is pulled high.



(2). PC response, but it prompts "USB device not recognized"


USB device not recognized
The last USB device you connected to this computer malfunctioned, and Windows does not recognize it.

The problem is that my own board uses self-powered usb. VBus is not used.
Therefore it should be disabled.

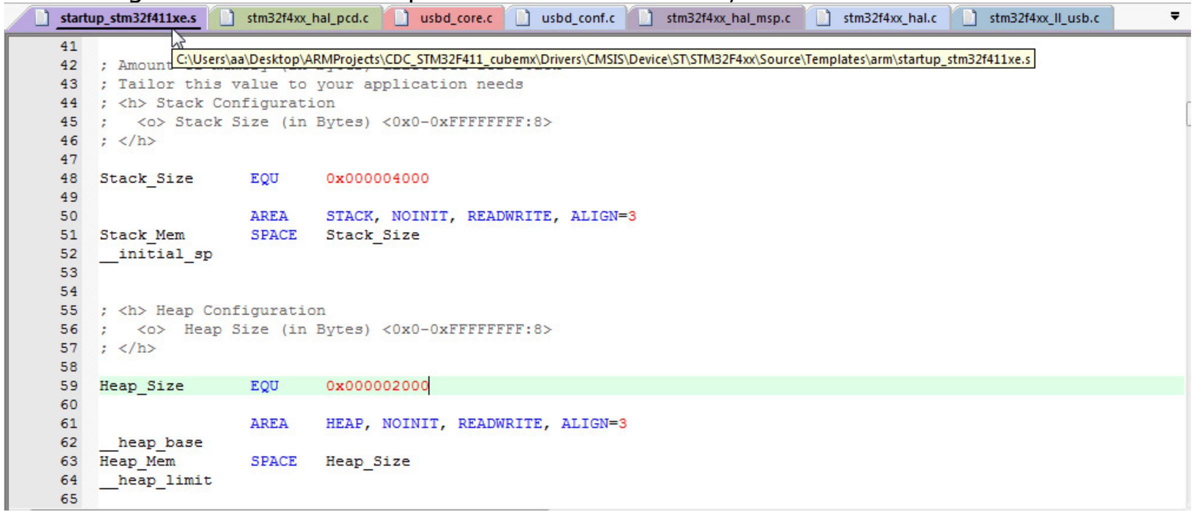


```
  * @param  pdev: Device handle
  * @retval USBD Status
  */
USBD_StatusTypeDef  USBD_LL_Init (USBD_HandleTypeDef *pdev)
{
  /* Init USB_IP */
  if (pdev->id == DEVICE_FS) {
  /* Link The driver to the stack */
  hpcd_USB_OTG_FS.pData = pdev;
  pdev->pData = &hpcd_USB_OTG_FS;

  hpcd_USB_OTG_FS.Instance = USB_OTG_FS;
  hpcd_USB_OTG_FS.Init.dev_endpoints = 7;
  hpcd_USB_OTG_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_OTG_FS.Init.dma_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.ep0_mps = DEP0CTL_MPS_64;
  hpcd_USB_OTG_FS.Init.phy_itface = PCD_PHY_EMBEDDED;
  hpcd_USB_OTG_FS.Init.Sof_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.vbus_sensing_enable = DISABLE;//changed to disable, since we don't use vbus
  hpcd_USB_OTG_FS.Init.use_dedicated_ep1 = DISABLE;
  HAL_PCD_Init(&hpcd_USB_OTG_FS);

  HAL_PCD_SetRxFiFo(&hpcd_USB_OTG_FS, 0x80);
  HAL_PCD_SetTxFiFo(&hpcd_USB_OTG_FS, 0, 0x40);
```

usb is self-powered, not using VBUS.
If not correct this, it will cause error: "usb device not recognized" by PC

(3).USB is recognized by PC, but reported as unknown or "This device cannot start"

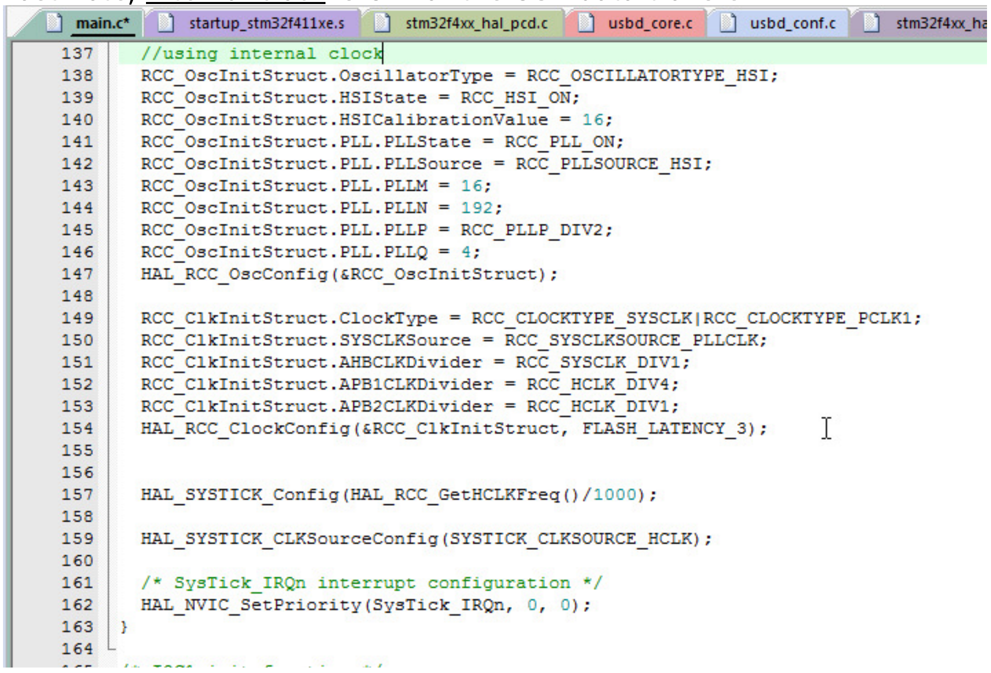
This problem was discussed in the forum as link:
This one is easy to correct. Go to startup file
and change stacksize and heap size from 400-> 4000, 200->2000.



C:\Users\aa\Desktop\ARMProjects\CDC_STM32F411_cubemx\Drivers\CMSIS\Device\ST\STM32F4xx\Source\Templates\arm\startup_stm32f411xe.s

```

; Amount
; Tailor this value to your application needs
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU     0x000004000

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp


; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU     0x000002000

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit

```

Last note, internal clock is OK for the USB data transfer.



```
  //using internal clock
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 192;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  HAL_RCC_OscConfig(&RCC_OscInitStruct);

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_SYSCLK|RCC_CLOCKTYPE_PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3);


  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
```

Dr. RongWei Mao
2015-10-23