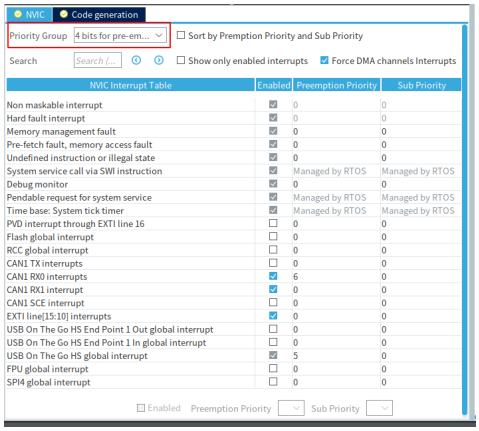
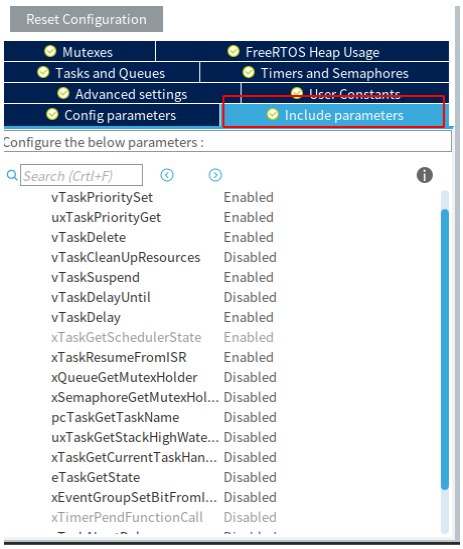
# 这里针对用Cube生成FreeRTOS的一些配置


使用cmsis版本1，因为版本1可以进行任务调度查询优先级进行指令上的优化。但是这只在CM3上有指令优化。


此时最大任务优先级不要超过32，因为cm3是32位的,优化指令是对1个32位寄存器进行操作的。

# 介绍一下CM3内核中断和FreeRTOS中断的记录


这个是include参数配置，即有一些函数默认不生成。

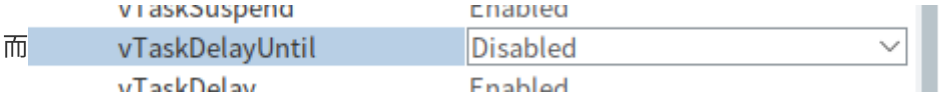
比如

然后查看

```
#if ( INCLUDE_vTaskDelayUntil == 1 )

    void vTaskDelayUntil( TickType_t * const pxPreviousWakeTime, const TickType_t xTimeIncrement )
    {
```

需要定义宏




而 这里没有打开。所以没有生成这个函数的实现，而只有定义。

---

## stm32优先级

在阐述一下中断优先级：中断优先级越低越优先

nvic是嵌套向量中断控制器，对于cm3-7???内核的都是通过一个8位的寄存器控制中断的优先级。但是stm32只使用其高4位，而低4位为0。所以stm32中只能有16个不同抢占优先级。而且这4个bits可以配置成不同的组合。如3位用做抢占优先级，1bit做副优先级。抢占优先级高的可以打断当前正在运行的低抢占优先级的中断——即嵌套中断。而同级别抢占优先级的中断到来，根据子优先级来决定先处理哪个中断。

| | | | |
|---|---|---|---|
| NVIC_PriorityGroup_0 | 0 级抢占优先级 | 0-15 级子优先级 | 0bit 用于抢占优先级<br>4bit 全用于子优先级 |
| NVIC_PriorityGroup_1 | 0-1 级抢占优先级 | 0-7 级子优先级 | 1bit 用于抢占优先级<br>3bit 用于子优先级 |
| NVIC_PriorityGroup_2 | 0-3 级抢占优先级 | 0-3 级子优先级 | 2bit 用于抢占优先级<br>2bit 用于子优先级 |
| NVIC_PriorityGroup_3 | 0-7 级抢占优先级 | 0-1 级子优先级 | 3bit 用于抢占优先级<br>1bit 用于子优先级 |
| NVIC_PriorityGroup_4 | 0-15 级抢占优先级 | 0 级子优先级 | 4bit 全用于抢占优先级<br>0bit 用于子优先级 |

但是stm32默认复位后是0组合，即无抢占。

## freertos中配置中断优先级

```
/* Cortex-M specific definitions. */
#ifdef __NVIC_PRIO_BITS
 /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
 #define configPRIO_BITS          __NVIC_PRIO_BITS
#else
 #define configPRIO_BITS          4
#endif
```

在freertosconfig.h中配置了以下宏。配置中断控制位是4bits和上面说的硬件情况一样。

以下的LIBRARY是表示配置的是用户接口的数值，实际写入寄存器的数值是这个数值<<4bits，因为stm32只用高四位。我们在32中配置写入的和freertos配置写入的如果不是直接填写寄存器也是填写原始的，然后实际写入的时候再<<4。

```
/* The lowest interrupt priority that can be used in a call to a "set priority"
function. */
#define configLIBRARY_LOWEST_INTERRUPT_PRIORITY   15
```

这个宏定义用来配置FreeRTOS用到的systick和pendsv(一个心跳、一个用于任务调度的上下文环境切换)。FreeRTOS吧这两个中断配成了系统中最低中断。

且FreeRTOS管理的中断也。

```
/* The highest interrupt priority that can be used by any interrupt service
routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
PRIORITY THAN THIS! (higher priorities are lower numeric values. */
#define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY 5
```

这个函数同样也是LIBRARY的数值，实际数值写入寄存器要<<4。这个数值用于表示可以在中断中调用FreeRTOS中断安全模式API(后缀ISR，在退出中断前需要进行任务调度一次)的最小中断优先级(最大优先中断)。即1-4优先级的中断内部是不许调用中断安全的FreeRTOSAPI的，RTOS会判断优先级不对劲后进入assert死循环。

```
* Interrupt priorities used by the kernel port layer itself.  These are generic
o all Cortex-M ports, and do not rely on any particular library functions. */
define configKERNEL_INTERRUPT_PRIORITY         ( configLIBRARY_LOWEST_INTERRUPT_PRIORITY << (8 - configPRIO_BITS) )
* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
ee http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
define configMAX_SYSCALL_INTERRUPT_PRIORITY    ( configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY << (8 - configPRIO_BITS) )
```

不带LIBRARY的，所以是直接写入寄存器的数值。第一个用来给FREERTOS内核用到的systick和pendsv的中断等级(15->240(即15<<4))。

第二个是configLibrary_max_syscall_intereup_priority <<4得到的，用来直接写入寄存器的。对于寄存器cm3内核寄存器basepri写入第二个宏

`configMAX_SYSCALL_INTERRUPT_PRIORITY`。就可以把大于等于这个数的中断都屏蔽掉。而写入0就可以关闭中断屏蔽而打开中断。

为何需要关闭中断，因为在RTOS内核许多东西调度的时候不希望被中断打断而改变什么东西，所以会关闭中断。由于关闭中断后中断需要在打开中断后延迟被处理，所以有些重要的中断必须不能被关闭。FreeRTOS能够实现这种功能的奥秘就在于FreeRTOS开关中断使用的是寄存器basepri，而像uCOS这种使用的是primask

| 名字 | 功能描述 |
| --- | --- |
| primask | 这是个只有 1 个 bit 的寄存器。 在它被置 1 后，就关掉所有可屏蔽的异常，只剩下 NMI 和硬 fault 可以响应。它的缺省值是 0，表示没有关中断。 |
| faultmask | 这是个只有 1 个 bit 的寄存器。当它置 1 时，只有 NMI 才能响应，所有其它的异常，甚至是硬 fault，也通通闭嘴。它的缺省值也是 0，表示没有关异常。 |
| basepri | 这个寄存器最多有 9 位(由表达优先级的位数决定)。它定义了被屏蔽优先级的阈值。当它被设成某个值后，所有优先级号大于等于此值的中断都被关(优先级号越大，优先级越低)。但若被设成 0，则不关闭任何中断， 0 也是缺省值。 |

在这里配置了中断优先级为5和6，因为在这两个中断回调中用到了 任务通知。

| | | | |
|---|---|---|---|
| CAN1 RX0 interrupts | ☑ | 6 | 0 |
| CAN1 RX1 interrupt | ☑ | 0 | 0 |
| CAN1 SCE interrupt | ☐ | 0 | 0 |
| EXTI line[15:10] interrupts | ☑ | 0 | 0 |
| USB On The Go HS End Point 1 Out global interrupt | ☐ | 0 | 0 |
| USB On The Go HS End Point 1 In global interrupt | ☐ | 0 | 0 |
| USB On The Go HS global interrupt | ☑ | 5 | 0 |
| FPU global interrupt | ☐ | 0 | 0 |

## FreeRTOS线程优先级

线程优先级是数值越高越优先。

这里configMAX_PRIORITIES表示任务优先级的最大值。且是不可以在cube中更改的。

以下是CMSISv2配置下的参数，

```
#define configENABLE_FPU                         1
#define configENABLE_MPU                         0

#define configUSE_PREEMPTION                     1
#define configSUPPORT_STATIC_ALLOCATION          1
#define configSUPPORT_DYNAMIC_ALLOCATION         1
#define configUSE_IDLE_HOOK                      0
#define configUSE_TICK_HOOK                      0
#define configCPU_CLOCK_HZ                       ( SystemCoreClock )
#define configTICK_RATE_HZ                       ((TickType_t)1000)
#define configMAX_PRIORITIES                     ( 56 )
#define configMINIMAL_STACK_SIZE                 ((uint16_t)128)
#define configTOTAL_HEAP_SIZE                    ((size_t)50000)
#define configMAX_TASK_NAME_LEN                  ( 16 )
#define configUSE_TRACE_FACILITY                 1
#define configUSE_16_BIT_TICKS                   0
#define configUSE_MUTEXES                        1
#define configQUEUE_REGISTRY_SIZE                8
#define configUSE_RECURSIVE_MUTEXES              1
#define configUSE_COUNTING_SEMAPHORES            1
```

而CMSISV1可以在cube中配置任务优先级最大值。


用于可以使用的任务优先级是**0～configMAX_PRIORITIES-1**，空闲任务优先级是0。且不要超过32，否则无法使用调度的硬件指令加速。

## 可视化插件

如果IDE是基于Eclipse的如CSS  STM32CUBEIDE这样的。可以下载插件https://www.taterli.com/2711/来可视化任务调度。

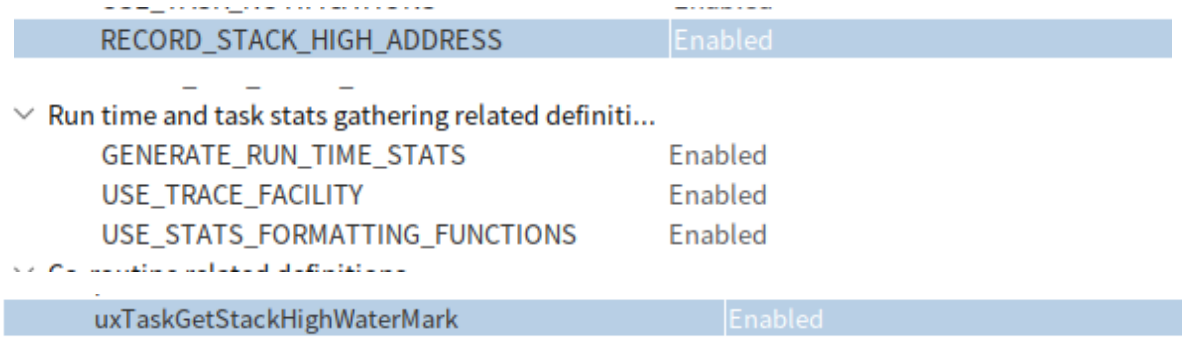
在cubeide中的help -install new software 添加

http://freescale.com/lgfiles/updates/Eclipse/KDS

http://www.highintegritysystems.com/StateViewer/

并参考一下几个链接进行使用

https://mcuoneclipse.com/2017/03/18/better-freertos-debugging-in-eclipse/




https://www.cnblogs.com/seifguo/p/9480935.html

## CPU效率

```
//启用运行时间统计功能
#define configGENERATE_RUN_TIME_STATS           1
//启用可视化跟踪调试
#define configUSE_TRACE_FACILITY                1
/* 与宏 configUSE_TRACE_FACILITY 同时为 1 时会编译下面 3 个函数
 * prvWriteNameToBuffer()
 * vTaskList(),
 * vTaskGetRunTimeStats()
*/
#define configUSE_STATS_FORMATTING_FUNCTIONS    1

extern volatile uint32_t CPU_RunTime;

#define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()      (CPU_RunTime = 0ul)
#define portGET_RUN_TIME_COUNTER_VALUE()               CPU_RunTime
```

其中最上面2个 为1的宏定义可以在cube中勾选。

```
/* USER CODE BEGIN Defines */
 extern volatile uint32_t CPU_RunTime;
 #define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS() (CPU_RunTime = 0ul)
 #define portGET_RUN_TIME_COUNTER_VALUE() CPU_RunTime
/* USER CODE END Defines */
```

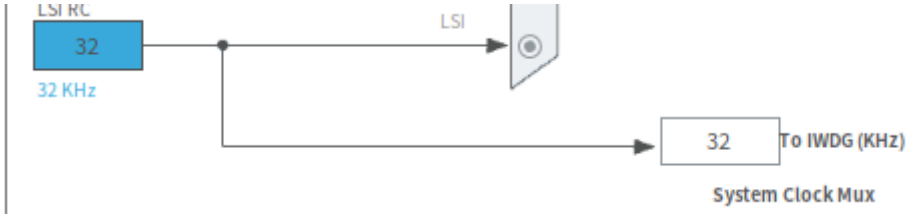
然后在_it.c中实现一个定时器回调来累计时间。用这个周期为FreeTOS时基1/10的定时器来给FreeRTOS计时

```
volatile uint32_t CPU_RunTime = 0UL;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim==&htim6)
  {
    CPU_RunTime++;
  }
}
```

## 线程看门狗

cube配IWDG。

然后在pid这种 单片机飞了 电机就会爆炸的地方加上 线程看门狗。




是32KHz的一个计时器。这样就是 32K/(4*4.095K)~2Hz


为了配置成20ms内线程不喂狗就复位可以配成

32K/(4*160)~50Hz  32k/50

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file        : usbd_cdc_if.h
 * @version     : v1.0_Cube
 * @brief       : Header for usbd_cdc_if.c file.
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under Ultimate Liberty license
 * SLA0044, the "License"; You may not use this file except in compliance with
 * the License. You may obtain a copy of the License at:
 *                      www.st.com/SLA0044
 *
 ******************************************************************************
 */
```

```
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __USBD_CDC_IF_H__
#define __USBD_CDC_IF_H__

#ifdef __cplusplus
 extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc.h"

/* USER CODE BEGIN INCLUDE */
#include "ringbuf.h"
/* USER CODE END INCLUDE */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
  * @brief For Usb device.
  * @{
  */

/** @defgroup USBD_CDC_IF USBD_CDC_IF
  * @brief Usb VCP device module
  * @{
  */

/** @defgroup USBD_CDC_IF_Exported_Defines USBD_CDC_IF_Exported_Defines
  * @brief Defines.
  * @{
  */
/* USER CODE BEGIN EXPORTED_DEFINES */
/* 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  100
#define APP_TX_DATA_SIZE  100

/* USER CODE END EXPORTED_DEFINES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Types USBD_CDC_IF_Exported_Types
  * @brief Types.
  * @{
  */

/* USER CODE BEGIN EXPORTED_TYPES */
typedef struct
{
   uint8_t data[APP_RX_DATA_SIZE];
}USBFrame;
/* USER CODE END EXPORTED_TYPES */

/**
  * @}
  */
```

```
/** @defgroup USBD_CDC_IF_Exported_Macros USBD_CDC_IF_Exported_Macros
  * @brief Aliases.
  * @{
  */

/* USER CODE BEGIN EXPORTED_MACRO */
extern RingBuf_t USBRxRingBuf;
/* USER CODE END EXPORTED_MACRO */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
  * @brief Public variables.
  * @{
  */

/** CDC Interface callback. */
extern USBD_CDC_ItfTypeDef USBD_Interface_fops_HS;

/* USER CODE BEGIN EXPORTED_VARIABLES */

/* USER CODE END EXPORTED_VARIABLES */

/**
  * @}
  */

/** @defgroup USBD_CDC_IF_Exported_FunctionsPrototype
USBD_CDC_IF_Exported_FunctionsPrototype
  * @brief Public functions declaration.
  * @{
  */

uint8_t CDC_Transmit_HS(uint8_t* Buf, uint16_t Len);

/* USER CODE BEGIN EXPORTED_FUNCTIONS */

/* USER CODE END EXPORTED_FUNCTIONS */

/**
  * @}
  */

/**
  * @}
  */

/**
  * @}
  */

#ifdef __cplusplus
}
#endif
```

```
#endif /* __USBD_CDC_IF_H__ */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file         : usbd_cdc_if.c
 * @version      : v1.0_Cube
 * @brief        : Usb device for Virtual Com Port.
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under Ultimate Liberty license
 * SLA0044, the "License"; You may not use this file except in compliance with
 * the License. You may obtain a copy of the License at:
 *                      www.st.com/SLA0044
 *
 ******************************************************************************
 */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "usbd_cdc_if.h"

/* USER CODE BEGIN INCLUDE */
#include "FreeRTOS.h"
#include "task.h"
/* USER CODE END INCLUDE */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/** @addtogroup STM32_USB_OTG_DEVICE_LIBRARY
 * @brief Usb device library.
 * @{
 */

/** @addtogroup USBD_CDC_IF
 * @{
 */

/** @defgroup USBD_CDC_IF_Private_TypesDefinitions USBD_CDC_IF_Private_TypesDefinitions
 * @brief Private types.
 * @{
 */
```

```
/* USER CODE BEGIN PRIVATE_TYPES */

/* USER CODE END PRIVATE_TYPES */

/**
 * @}
 */

/** @defgroup USBD_CDC_IF_Private_Defines USBD_CDC_IF_Private_Defines
 * @brief Private defines.
 * @{
 */

/* USER CODE BEGIN PRIVATE_DEFINES */
/* USER CODE END PRIVATE_DEFINES */

/**
 * @}
 */

/** @defgroup USBD_CDC_IF_Private_Macros USBD_CDC_IF_Private_Macros
 * @brief Private macros.
 * @{
 */

/* USER CODE BEGIN PRIVATE_MACRO */

/* USER CODE END PRIVATE_MACRO */

/**
 * @}
 */

/** @defgroup USBD_CDC_IF_Private_Variables USBD_CDC_IF_Private_Variables
 * @brief Private 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 UserRxBufferHS[APP_RX_DATA_SIZE];

/** Data to send over USB CDC are stored in this buffer   */
uint8_t UserTxBufferHS[APP_TX_DATA_SIZE];

/* USER CODE BEGIN PRIVATE_VARIABLES */
#define USB_RINGBUF_LENGTH 100
USBFrame USBRxBuf[USB_RINGBUF_LENGTH];
RingBuf_t USBRxRingBuf;
/* USER CODE END PRIVATE_VARIABLES */

/**
 * @}
 */

/** @defgroup USBD_CDC_IF_Exported_Variables USBD_CDC_IF_Exported_Variables
```

```
 * @brief Public variables.
 * @{
 */

extern USBD_HandleTypeDef hUsbDeviceHS;

/* USER CODE BEGIN EXPORTED_VARIABLES */
extern TaskHandle_t usb_process_packet_thread_Handle;
/* USER CODE END EXPORTED_VARIABLES */

/**
 * @}
 */

/** @defgroup USBD_CDC_IF_Private_FunctionPrototypes USBD_CDC_IF_Private_FunctionPrototypes
 * @brief Private functions declaration.
 * @{
 */

static int8_t CDC_Init_HS(void);
static int8_t CDC_DeInit_HS(void);
static int8_t CDC_Control_HS(uint8_t cmd, uint8_t *pbuf, uint16_t length);
static int8_t CDC_Receive_HS(uint8_t *pbuf, uint32_t *Len);
static int8_t CDC_TransmitCplt_HS(uint8_t *pbuf, uint32_t *Len, uint8_t epnum);

/* USER CODE BEGIN PRIVATE_FUNCTIONS_DECLARATION */

/* USER CODE END PRIVATE_FUNCTIONS_DECLARATION */

/**
 * @}
 */

USBD_CDC_ItfTypeDef USBD_Interface_fops_HS = { CDC_Init_HS, CDC_DeInit_HS,
      CDC_Control_HS, CDC_Receive_HS, CDC_TransmitCplt_HS };

/* Private functions ---------------------------------------------------------*/

/**
 * @brief  Initializes the CDC media low layer over the USB HS IP
 * @retval USBD_OK if all operations are OK else USBD_FAIL
 */
static int8_t CDC_Init_HS(void) {
  /* USER CODE BEGIN 8 */
  RingBuf_Creat(&USBRxRingBuf, USBRxBuf, USB_RINGBUF_LENGTH,
        sizeof(USBFrame));
  /* Set Application Buffers */
  USBD_CDC_SetTxBuffer(&hUsbDeviceHS, UserTxBufferHS, 0);
  USBD_CDC_SetRxBuffer(&hUsbDeviceHS, UserRxBufferHS);
  return (USBD_OK);
  /* USER CODE END 8 */
}

/**
 * @brief  DeInitializes the CDC media low layer
 * @param  None
 * @retval USBD_OK if all operations are OK else USBD_FAIL
 */
```

```
static int8_t CDC_DeInit_HS(void) {
  /* USER CODE BEGIN 9 */
  return (USBD_OK);
  /* USER CODE END 9 */
}

/**
 * @brief  Manage the CDC class requests
 * @param  cmd: Command code
 * @param  pbuf: Buffer containing command data (request parameters)
 * @param  length: Number of data to be sent (in bytes)
 * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
 */
static int8_t CDC_Control_HS(uint8_t cmd, uint8_t *pbuf, uint16_t length) {
  /* USER CODE BEGIN 10 */
  switch (cmd) {
  case CDC_SEND_ENCAPSULATED_COMMAND:

    break;

  case CDC_GET_ENCAPSULATED_RESPONSE:

    break;

  case CDC_SET_COMM_FEATURE:

    break;

  case CDC_GET_COMM_FEATURE:

    break;

  case CDC_CLEAR_COMM_FEATURE:

    break;

    /*******************************************************************************/
    /* Line Coding Structure                                                       */
    /*-----------------------------------------------------------------------------*/
    /* Offset | Field      | Size | Value  | Description                          */
    /* 0    | dwDTERate   |   4  | Number |Data terminal rate, in bits per second*/
    /* 4    | bCharFormat |   1  | Number | Stop bits                            */
    /*                               0 - 1 Stop bit                                */
    /*                               1 - 1.5 Stop bits                             */
    /*                               2 - 2 Stop bits                               */
    /* 5    | bParityType |  1   | Number | Parity                               */
    /*                               0 - None                                      */
    /*                               1 - Odd                                       */
    /*                               2 - Even                                      */
    /*                               3 - Mark                                      */
    /*                               4 - Space                                     */
    /* 6    | bDataBits  |   1   | Number Data bits (5, 6, 7, 8 or 16).          */
    /*******************************************************************************/
  case CDC_SET_LINE_CODING:

    break;

  case CDC_GET_LINE_CODING:
```

```
    break;

  case CDC_SET_CONTROL_LINE_STATE:

    break;

  case CDC_SEND_BREAK:

    break;

  default:
    break;
  }

  return (USBD_OK);
  /* USER CODE END 10 */
}

/**
 * @brief  Data received over USB OUT endpoint are sent over CDC interface
 *       through this function.
 *
 *       @note
 *       This function will issue a NAK packet on any OUT packet received on
 *       USB endpoint until exiting this function. If you exit this function
 *       before transfer is complete on CDC interface (ie. using DMA controller)
 *       it will result in receiving more data while previous ones are still
 *       not sent.
 *
 * @param  Buf: Buffer of data to be received
 * @param  Len: Number of data received (in bytes)
 * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
 */
static int8_t CDC_Receive_HS(uint8_t *Buf, uint32_t *Len) {
  /* USER CODE BEGIN 11 */

  USBFrame *frameptr;
  RingBuf_Write(&USBRxRingBuf, (void**) &frameptr);

  USBD_CDC_SetRxBuffer(&hUsbDeviceHS, frameptr->data);
  USBD_CDC_ReceivePacket(&hUsbDeviceHS);
  BaseType_t xHigherPriorityTaskWoken = pdFALSE;
  //https://www.cnblogs.com/w-smile/p/11333950.html
  //外设中断优先级不能小于 5 否则   不能调用rtos ISRapi .
  vTaskNotifyGiveFromISR(usb_process_packet_thread_Handle,
      &xHigherPriorityTaskWoken);
  portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
  return (USBD_OK);
  /* USER CODE END 11 */
}

/**
 * @brief  Data to send over USB IN endpoint are sent over CDC interface
 *       through this function.
 * @param  Buf: Buffer of data to be sent
 * @param  Len: Number of data to be sent (in bytes)
 * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL or USBD_BUSY
```

```
 */
uint8_t CDC_Transmit_HS(uint8_t *Buf, uint16_t Len) {
  uint8_t result = USBD_OK;
  /* USER CODE BEGIN 12 */
  USBD_CDC_HandleTypeDef *hcdc =
       (USBD_CDC_HandleTypeDef*) hUsbDeviceHS.pClassData;
  if (hcdc->TxState != 0) {
    return USBD_BUSY;
  }
  USBD_CDC_SetTxBuffer(&hUsbDeviceHS, Buf, Len);
  result = USBD_CDC_TransmitPacket(&hUsbDeviceHS);
  /* USER CODE END 12 */
  return result;
}

/**
 * @brief  CDC_TransmitCplt_HS
 *       Data transmited callback
 *
 *       @note
 *       This function is IN transfer complete callback used to inform user that
 *       the submitted Data is successfully sent over USB.
 *
 * @param  Buf: Buffer of data to be received
 * @param  Len: Number of data received (in bytes)
 * @retval Result of the operation: USBD_OK if all operations are OK else USBD_FAIL
 */
static int8_t CDC_TransmitCplt_HS(uint8_t *Buf, uint32_t *Len, uint8_t epnum) {
  uint8_t result = USBD_OK;
  /* USER CODE BEGIN 14 */
  UNUSED(Buf);
  UNUSED(Len);
  UNUSED(epnum);
  /* USER CODE END 14 */
  return result;
}

/* USER CODE BEGIN PRIVATE_FUNCTIONS_IMPLEMENTATION */

/* USER CODE END PRIVATE_FUNCTIONS_IMPLEMENTATION */

/**
 * @}
 */

/**
 * @}
 */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```