基于STM32F407HAL库串口DMA+空闲中断

核心板：STM32F407

实验目的：通过DMA接收串口发来的数据，并且利用串口空闲中断在将这些数据发送至串口助手 。

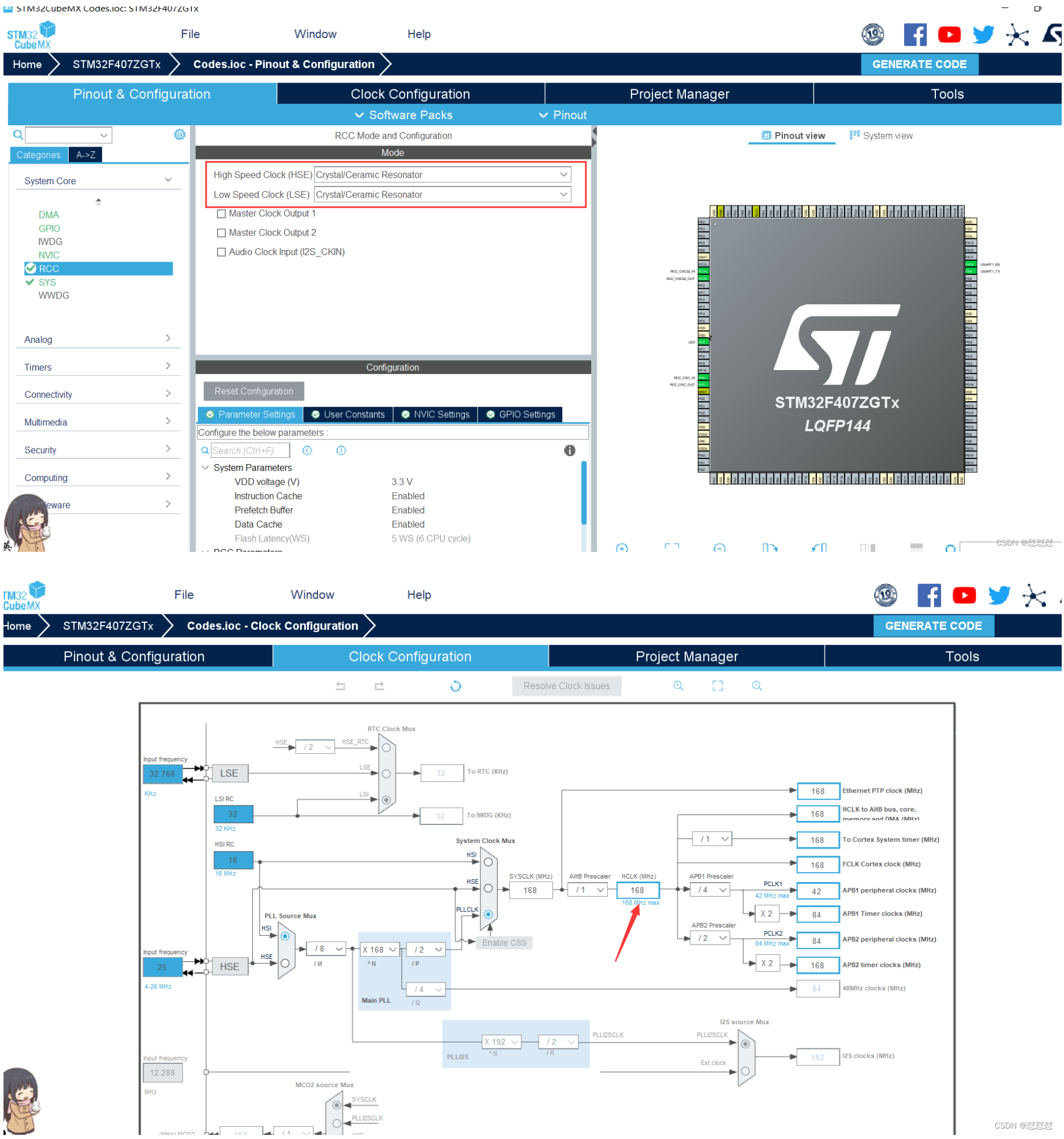
方法一：利用以下函数

```
/* 在DMA模式下接收一定数量的数据，直到接收到预期数量的数据或发生空闲事件 */
HAL_UARTEx_ReceiveToIdle_DMA(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);
/* 以DMA模式发送大量数据 */
HAL_UART_Transmit_DMA(UART_HandleTypeDef *huart, const uint8_t *pData, uint16_t Size);
/* 接待事件回调(使用高级接待服务后调用的Rx事件通知) */
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size);
```

方法二：自己手动打开IDLE空闲中断，开启DMA接收模式，编写空闲中断中的逻辑代码。

首先利用STM32CubeMX 进行配置

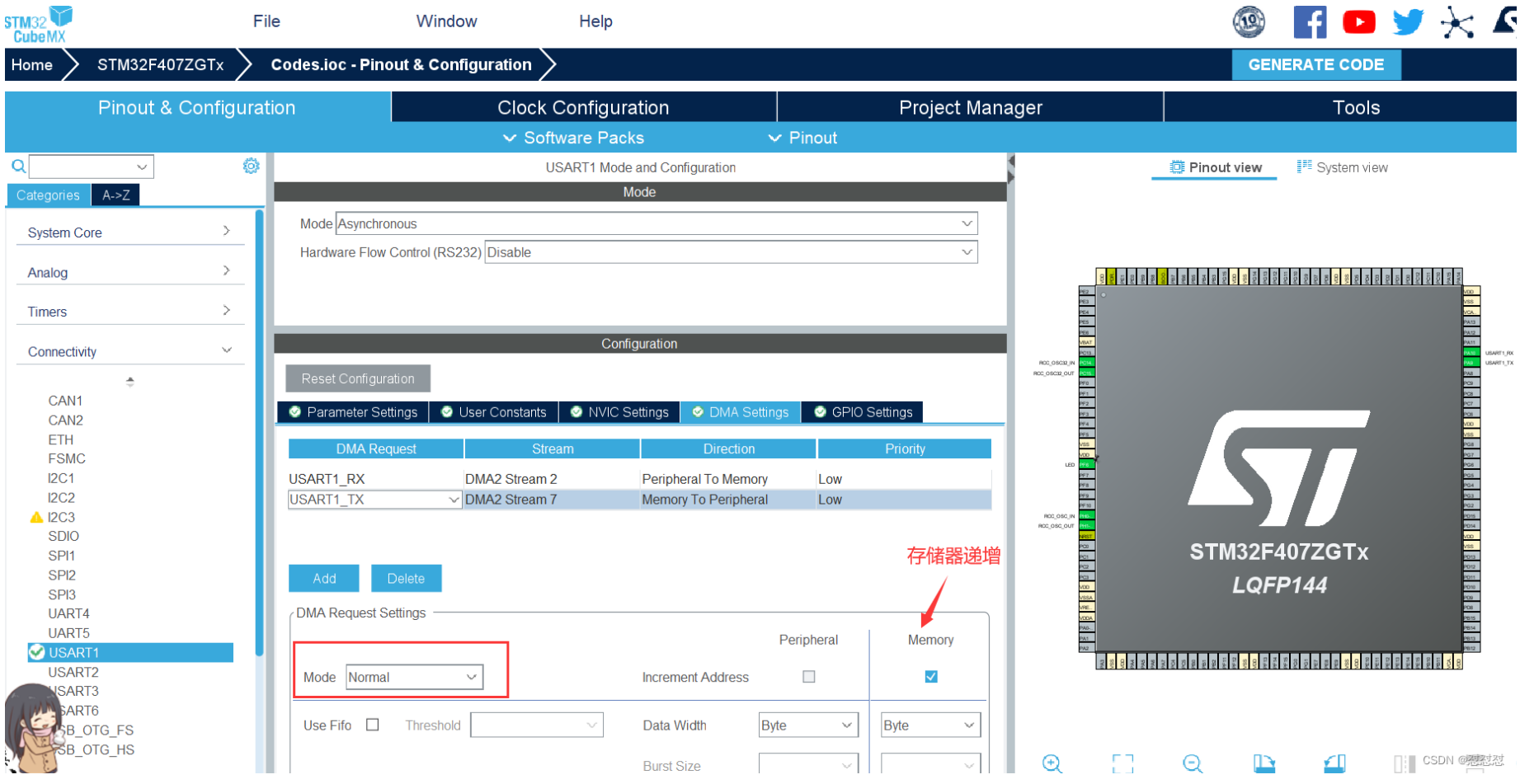
配置为最高时钟



配置串口DMA模式接收与发送

开启串口全局中断

配置DMA，点击ADD进行添加，接收为循环模式，发送为正常模式



生成代码。

方法一具体代码及步骤

打开main.c，在主函数外定义相关变量并且编写回调函数

```c
char pData[255];

void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)
{//Size为接收到的数据大小
    if(huart->Instance == USART1)
    {
        HAL_UART_DMAStop(&huart1);//关闭是为了重新设置发送多少数据，不关闭会造成数据错误

        HAL_UART_Transmit_DMA(&huart1, (uint8_t *)pData, Size);//设置DMA发送多少数据

        HAL_UARTEx_ReceiveToIdle_DMA(&huart1,(uint8_t *)pData, 255);//继续开启空闲中断DMA发送
    }
}
```

主函数

```c
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

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

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */

  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

   HAL_UARTEx_ReceiveToIdle_DMA(&huart1, (uint8_t *)pData, 255);//开启串口空闲中断DMA接收数据

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
```

```
    /* USER CODE END WHILE */
39
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

方法二具体代码及步骤

在main.c中声明变量并在main.h声明该变量为全局变量

```
/* main.c中 */
char pData[255];

/* main.h中 */
extern char pData[255];
```

在usart.c中编写空闲中断中的逻辑代码并在usart.h中声明

```
void USER_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART1)
    {
        if(__HAL_UART_GET_FLAG(&huart1, UART_FLAG_IDLE) != RESET)
        {
             __HAL_UART_CLEAR_IDLEFLAG(&huart1); //清除IDLE标志

            uint8_t Len = 255 - __HAL_DMA_GET_COUNTER(&hdma_usart1_rx);

            HAL_UART_DMAStop(&huart1);//停止DMA，为了重新设置DMA发送多少数据

            HAL_UART_Transmit_DMA(&huart1, (uint8_t *)pData, Len);

            HAL_UART_Receive_DMA(&huart1, (uint8_t *)pData, 255);
        }
    }
}
```

在stm32f4xx_it.c中包含usart.h，并在void USART1_IRQHandler()中调用USER_UART_RxCpltCallback(&huart1);

```
void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */

  USER_UART_RxCpltCallback(&huart1);

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */

  /* USER CODE END USART1_IRQn 1 */
}
```

在主函数中开启空闲中断，开启DMA接收

```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

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

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

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

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART1_UART_Init();
```

```
  /* USER CODE BEGIN 2 */
    __HAL_UART_ENABLE_IT(&huart1, UART_IT_IDLE);//使能串UART1 IDLE中断

    HAL_UART_Receive_DMA(&huart1, (uint8_t *)pData, 255);//开启DMA接收模式

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

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

实验结果：（两个都一样）

对于不太清除为什么DMA要不断使能失能可以看一下下面这个文章

浅谈使用DMA时需要的不断使能失能的原因

以上仅供自己与大家学习积累，欢迎各位大佬批评与指正！