

# UART IDLE 中断使用-接收不定长串口数据

## 前言

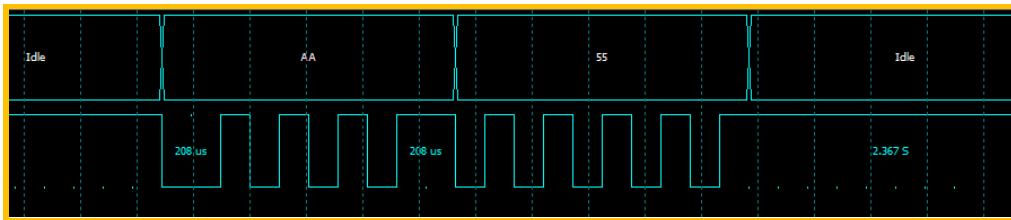
在串口通信应用中，我们常使用接受和发送中断，相信大家都不陌生。这里有个非常用的中断可能被大家所忽略，即总线 IDLE 中断。当一帧数据传输结束之后，总线会维持高电平状态，此时，就可以触发 MCU 的 IDLE 中断。在本文中，将介绍使用该中断来进行不定长串口数据接收的办法。通过该中断，可以省却很多用于检测数据传输是否完成的判断动作。

## 实验环境

- STM32F411RE-NUCLEO
- STM32CubeMX

## 总线状态分析

下图是发送 0xAA 0x55 的所抓取到的波形。从图中我们可以看到在发送该帧之前和之后，数据线处于 IDLE 的状态。在该帧中，字节与字节之间，没有 IDLE 状态出现，即不会出现 IDLE 误触发的情况。



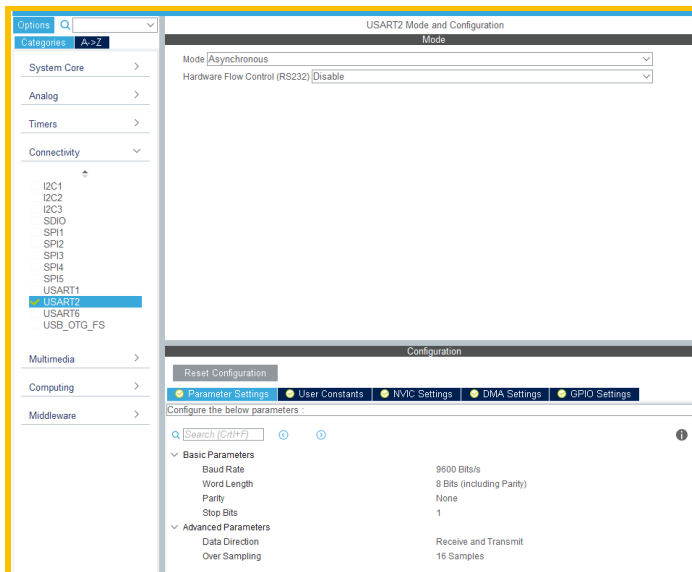
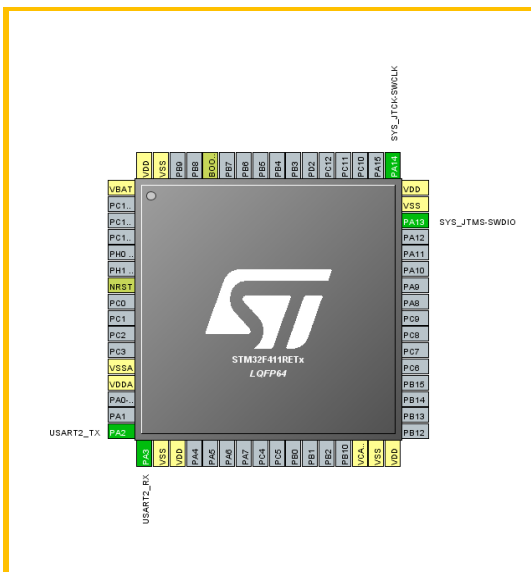
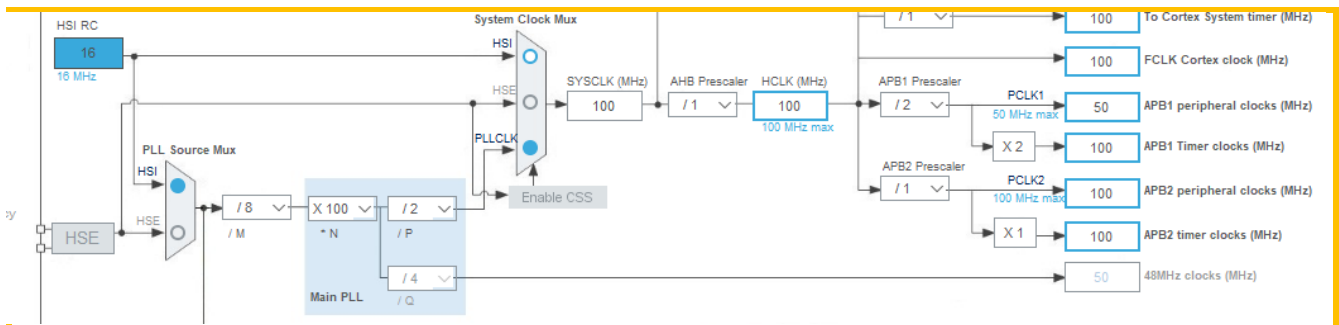
## 不定长数据接收

本次制作的工程是基于 HAL 库。在原生的 HAL 库中，并没有集成 IDLE 中断的处理。所以，在本文我们介绍的方法中，需要修改一些库文件来实现。

### 使用 STM32CubeMX 生成实验工程

工程的配置如下图：

1. 系统始终配置为 100MHz
2. 配置 USART2 为 Asynchronous，管脚配置为 PA2，PA3。
3. USART2 参数：9600Bits/s, 8bits, None, 1Stop



为了方便打印接收到的相关信息，需要对生成的工程做如下修改来映射 **print** 函数。

main.c-声明

```
#ifndef __GNUC__
/* With GCC, small printf (option LD Linker->Libraries->Small printf
   set to 'Yes') calls __io_putchar() */
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
```

main.c-Code

```
/**
 * @brief Retargets the C library printf function to the USART.
 * @param None
 * @retval None
 */
PUTCHAR_PROTOTYPE
{
    /* Place your implementation of fputc here */
    /* e.g. write a character to the EVAL_COM1 and Loop until the end of transmission */
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);

    return ch;
}
```

```
}

```

## 修改工程代码

增加接收 Buffer

main.c

```
//Modification 0
//Store the received bytes number
uint32_t Rev_Size = 0;
//Receive buffer
uint8_t UART_RX_Buf[15];

```

stm32f4xx\_hal\_uart.c

```
//Modification 0
extern uint32_t Rev_Size;

```

在接收函数中使能 IDLE 中断

stm32f4xx\_hal\_uart.c -> HAL\_UART\_Receive\_DMA()函数

```
//Modification 1
/* Enable the UART IDLE Interrupt*/
SET_BIT(huart->Instance->CR1, USART_CR1_IDLEIE);

```

处理 IDLE 中断

stm32f4xx\_hal\_uart.c -> HAL\_UART\_IRQHandler ()函数

```
//Modification 2
if(((isrflags & USART_SR_IDLE) != RESET) && ((cr1its & USART_CR1_IDLEIE) != RESET))
{
    //Record the received bytes number
    Rev_Size = huart->RxFferSize - huart->hdmarx->Instance->NDTR;
    //clear the IDLE flag
    __HAL_UART_CLEAR_IDLEFLAG(huart);
    //Abord the received process
    HAL_UART_AbortReceive_IT(huart);
    return;
}

```

接收完成处理 (IDLE 产生, 一帧数据传输完成)

stm32f4xx\_hal\_uart.c -> HAL\_UART\_AbortReceive\_IT ()函数

```
//Modification 3
CLEAR_BIT(huart->Instance->CR1, (USART_CR1_RXNEIE | USART_CR1_PEIE | USART_CR1_IDLEIE));
// CLEAR_BIT(huart->Instance->CR1, (USART_CR1_RXNEIE | USART_CR1_PEIE));

```

main.c

```
//Modification 4
void HAL_UART_AbortReceiveCpltCallback (UART_HandleTypeDef *huart)
{
    //Print received Bytes
    printf("\n\r[IDLE]Received %d Bytes:",Rev_Size);
}

```

```
for(uint16_t i = 0; i < Rev_Size; i++)
{
    printf(" 0x%02X", UART_RX_Buf[i]);
}
//Re-start receiving
HAL_UART_Receive_DMA(&huart2, UART_RX_Buf, 15);
/* NOTE : This function should not be modified, when the callback is needed,
the HAL_UART_AbortTransmitCpltCallback can be implemented in the user file.
*/
}
```

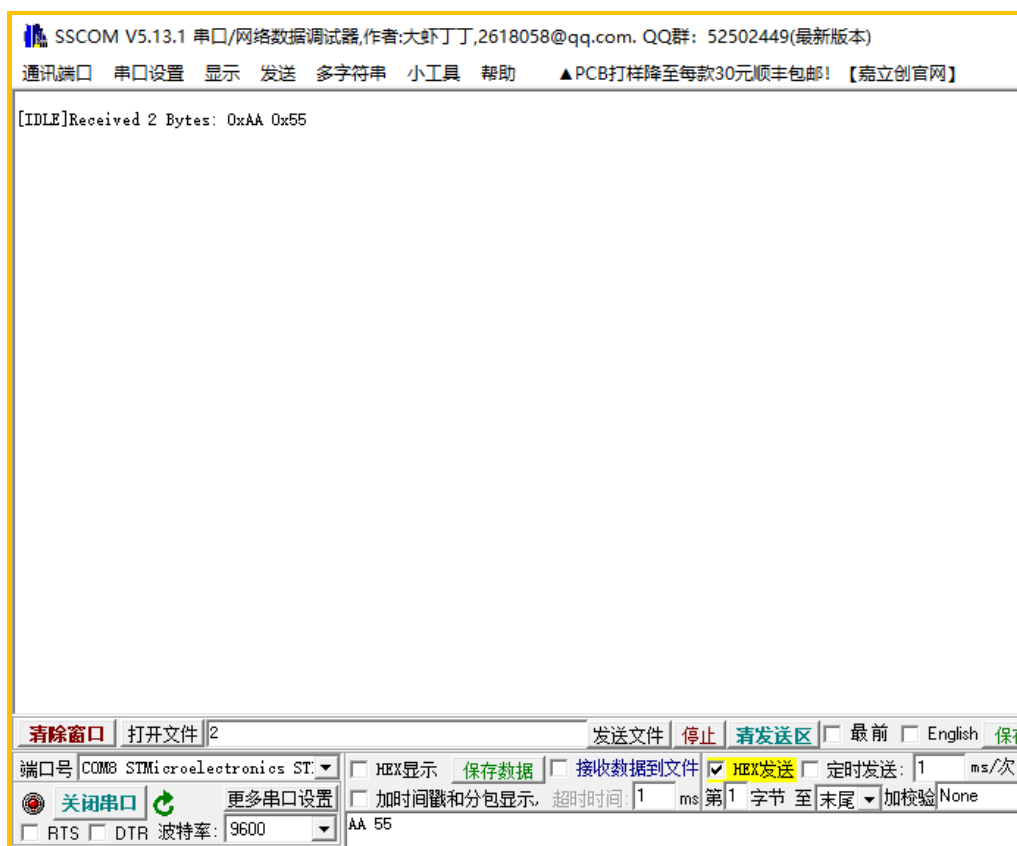
使能接收  
main.c

```
//Modification 5. '15' is the total number to be received. Make sure it can cover the
longest frame.
HAL_UART_Receive_DMA(&huart2, UART_RX_Buf, 15);
```

## 实验结果

使用串口调试串口，通过 **STLINK** 的虚拟串口发送数据，MCU 会返回接收多少个字节的数据，并将接收到的数据打印出来。

下图是发送 0xAA 0x55 的实验结果。



## 小结

IDLE 作为指示总线空闲状态的中断，合理的使用能够很大程度的节省代码程序设计工作量。这里只是做个简单示范，抛砖引玉，实际应用中希望大家可以灵活使用满足所需。

### **重要通知 - 请仔细阅读**

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。