

LE05 Uart发送不同长度Break应用

版本

Config Tool Version：2.4.0

LE0 SDK version：1.1.0

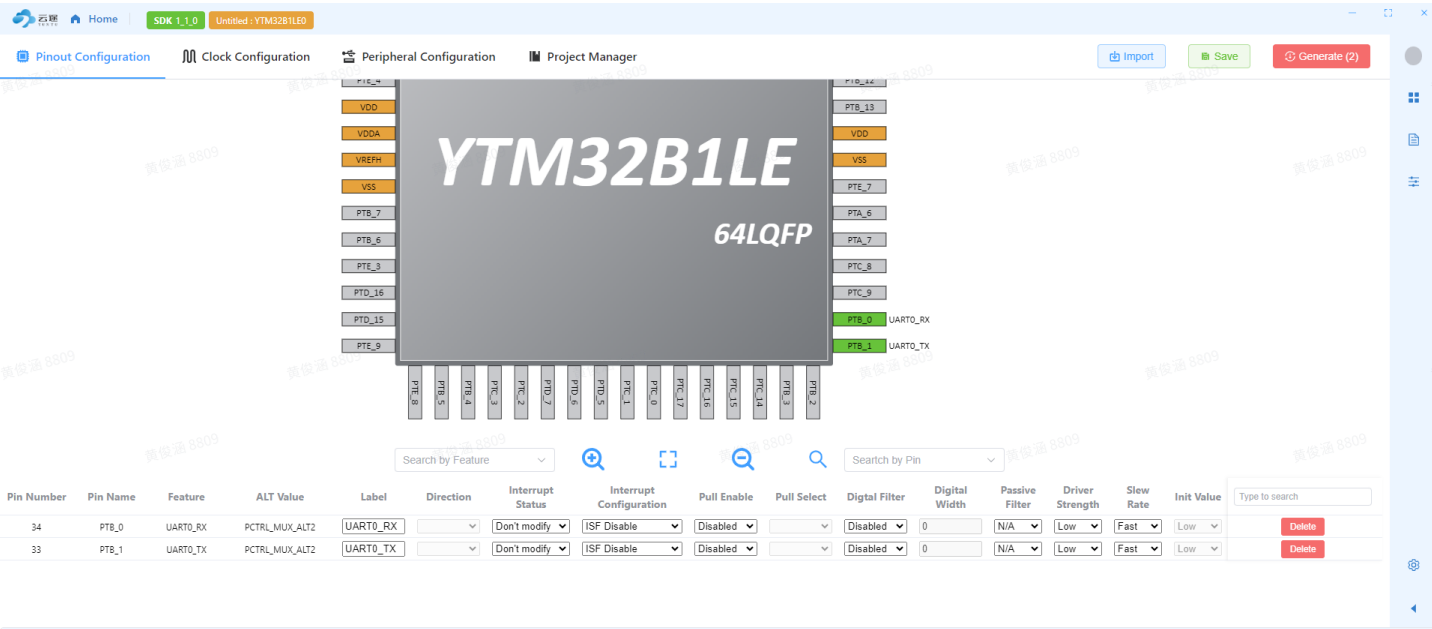
1. 前言

本文基于YTM32B1LE05微控制器，介绍如何使用UART模块单独发送Break场。由于LE系列为UART模拟LIN，在某些应用场景下需要单独发送Break场，本文将分别介绍固定长度Break场以及可变长度Break场的发送方法。

2. 基础配置

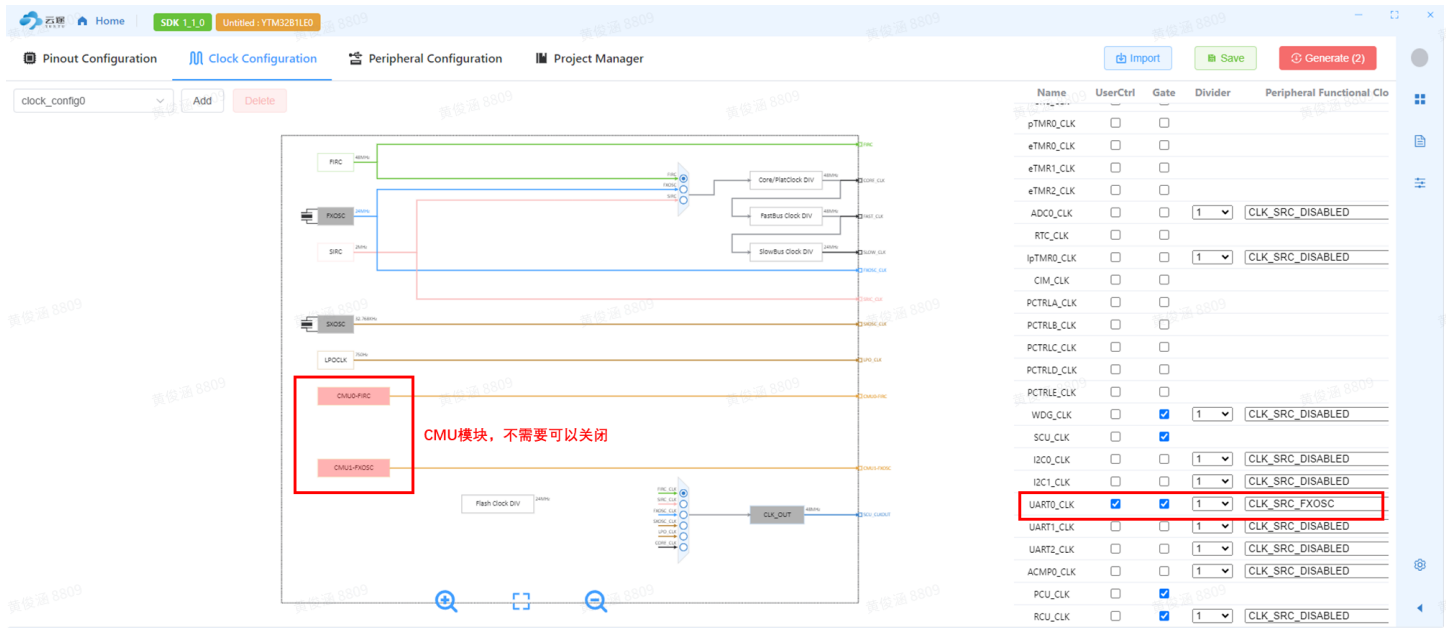
2.1 引脚配置

选择PTB0、PTB1分别配置为UART0_RX和UART0_TX脚

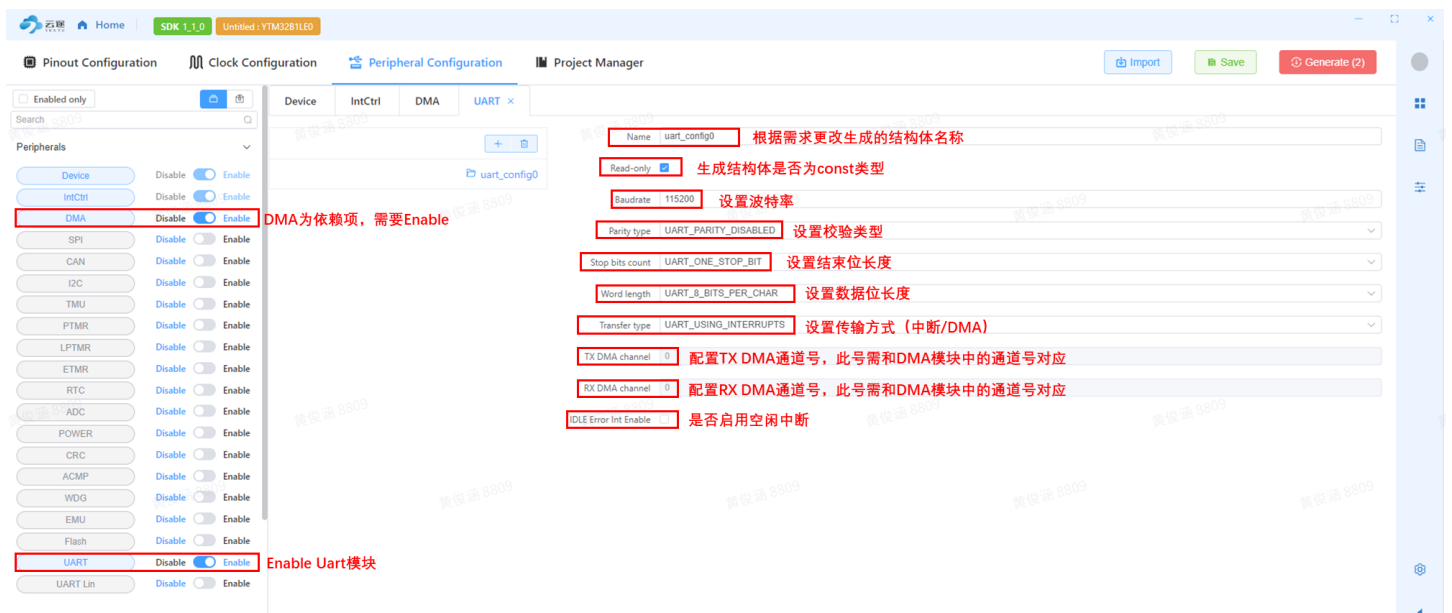


2.2 配置时钟

选择时钟源，示例以外部晶振为时钟源，云途开发板外部晶振为24M



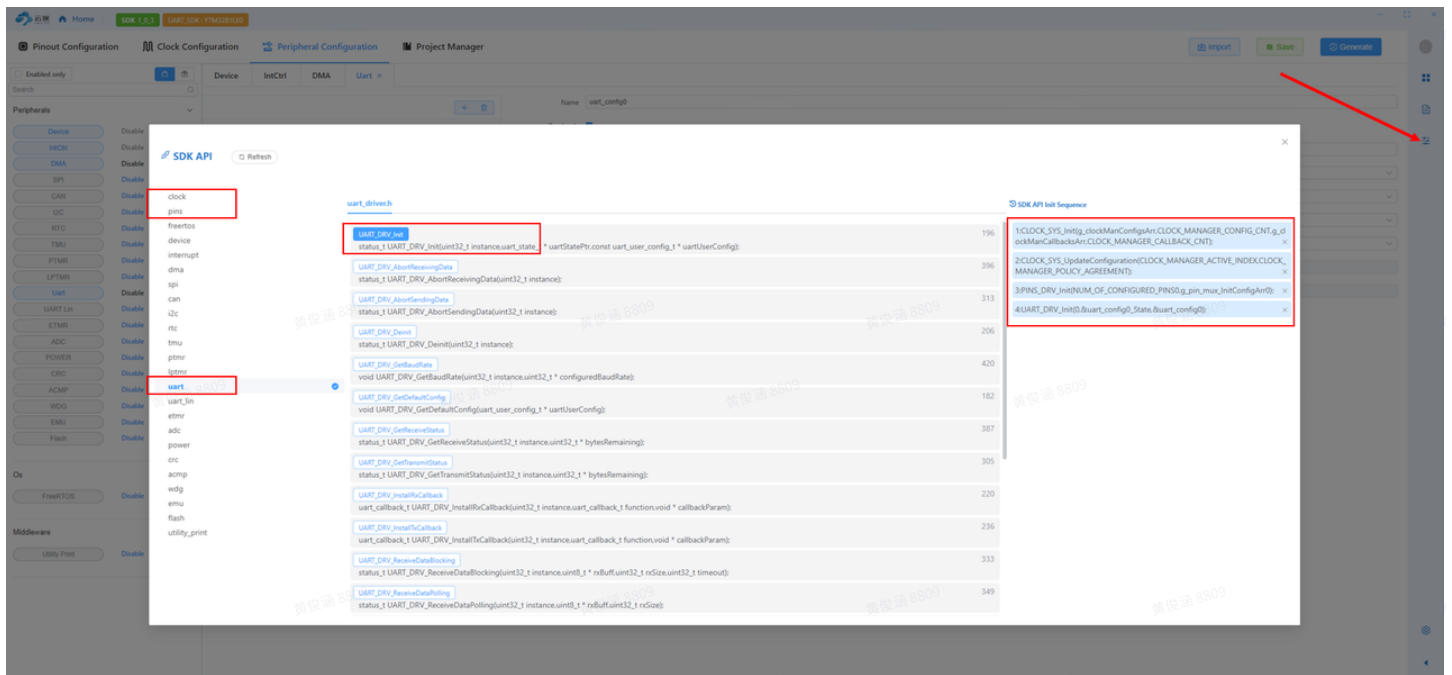
2.3 配置UART模块



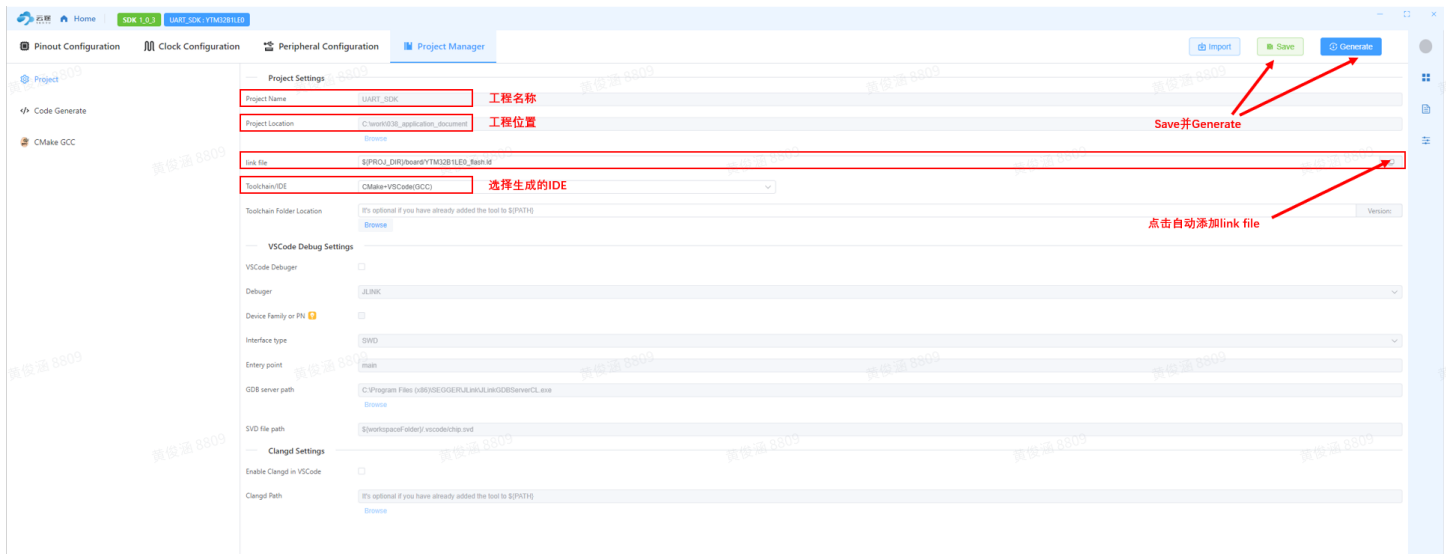
2.4 添加初始化API

应用程序调用的初始化硬件的函数board_init()将要包含对时钟、引脚和UART外设模块的初始化配置过程，在YCT中，也可以选择对应的API函数，自动生成调用代码：

- 添加时钟初始化函数
- 添加引脚初始化函数
- 添加UART初始化函数



2.5 生成工程



3. 生成代码示例

以下将分别介绍：

- **方式一：** 将发送固定长度Break，第一帧正常发送两个字节的的数据{0x30, 0x31}，第二帧将包含一个10bit长度的Break和两个字节的的数据{0x30, 0x31}；
- **方式二：** 发送可变长度Break，第一帧正常发送两个字节的的数据{0x30, 0x31}，第二帧起始将包含Break和两个字节的的数据{0x30, 0x31}；

3.1 方式一：发送固定长度Break

该方式需要手动操作寄存器；

1个UART帧长度包含 **start位+数据位+stop位**，将LINBRK寄存器位置1将发送一个和UART帧等长度的Break帧；



Note: 在发送完Break后LINBRK寄存器会自动清0，若要多次发送Break，则需多次将LINBRK位置1。

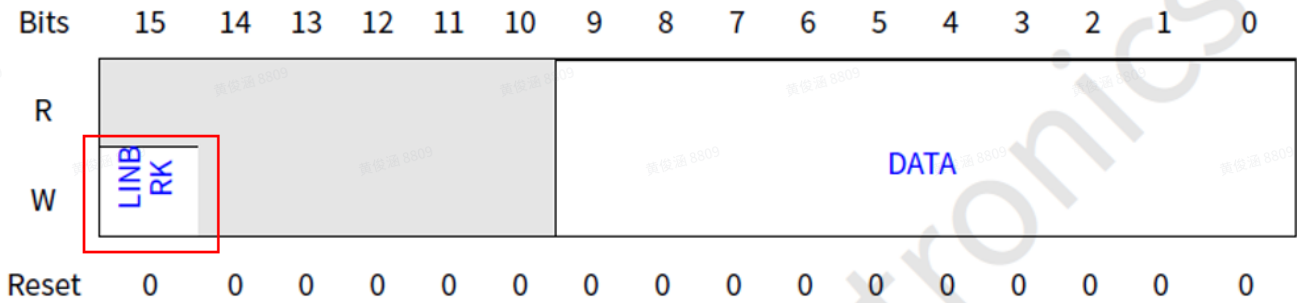


Table 17.11: UART DATA Register Description

Field	Function
16 RXEMPT	RX FIFO Empty 0b - UART RX FIFO is not empty, reading data is valid 1b - UART RX FIFO is empty, reading data is invalid
15 LINBRK	Send LIN Break Writing LINBRK bit will let UART send out LIN break instead of DATA
9 - 0 DATA	DATA Field Reading DATA field will return data in RX FIFO, writing DATA field will push data to TX FIFO


3.1.1 代码实现

1. 定义变量

Note: g_bytesRemaining为调用获取发送是否完成函数时需要传入的参数，该参数将存放剩余字节数

```
39  /* USER CODE BEGIN PM */
40  #define UART_INSTANCE          (0) 定义UART的instance
41  #define UART_TX_LENGTH         (2) 发送Data字节数
42  #define UART_BREAK_LENGTH      (2) 发送Break时选择发送多少个Data字节
43  /* USER CODE END PM */
44
45  /* Private variables -----*/
46  /* USER CODE BEGIN PV */
47  uint8_t g_uartTxBuffer[] = {0x30, 0x31 }; 定义收发buffer
48  uint8_t g_uartRxBuffer[UART_TX_LENGTH] = {0};
49  uint32_t g_bytesRemaining; 定义剩余字节数变量，用于判断发送是否完成
50  /* USER CODE END PV */
```

2. 发送函数

 **Note:** 在下图中，当寄存器LINBRK位被置1后再次调用UART_DRV_SendData函数将发送Break+Data，变量UART_BREAK_LENGTH控制发送Data的数量，若UART_BREAK_LENGTH为2则发送1个Break+2个Data，若UART_BREAK_LENGTH为0则只发送Break

```
66 int main(void)
67 {
68     /* USER CODE BEGIN 1 */
69     /* USER CODE END 1 */
70     Board_Init();
71     /* USER CODE BEGIN 2 */
72     UART_DRV_SendData(UART_INSTANCE, g_uartTxBuffer, UART_TX_LENGTH);
73     /* Waiting for transmission to complete */
74     while (UART_DRV_GetTransmitStatus(UART_INSTANCE, &g_bytesRemaining)){
75
76         UART0->DATA = UART_DATA_LINBRK(1);
77         UART_DRV_SendData(UART_INSTANCE, g_uartTxBuffer, UART_BREAK_LENGTH);
78         while (UART_DRV_GetTransmitStatus(UART_INSTANCE, &g_bytesRemaining)){
79             /* USER CODE END 2 */
80
81             /* Infinite loop */
82             /* USER CODE BEGIN WHILE */
83             while (1)
84             {
85                 /* USER CODE END WHILE */
86                 /* USER CODE BEGIN 3 */
87             }
88             /* USER CODE END 3 */
89         }
90     }
```

发送只有两个字节的数据帧

等待发送完成

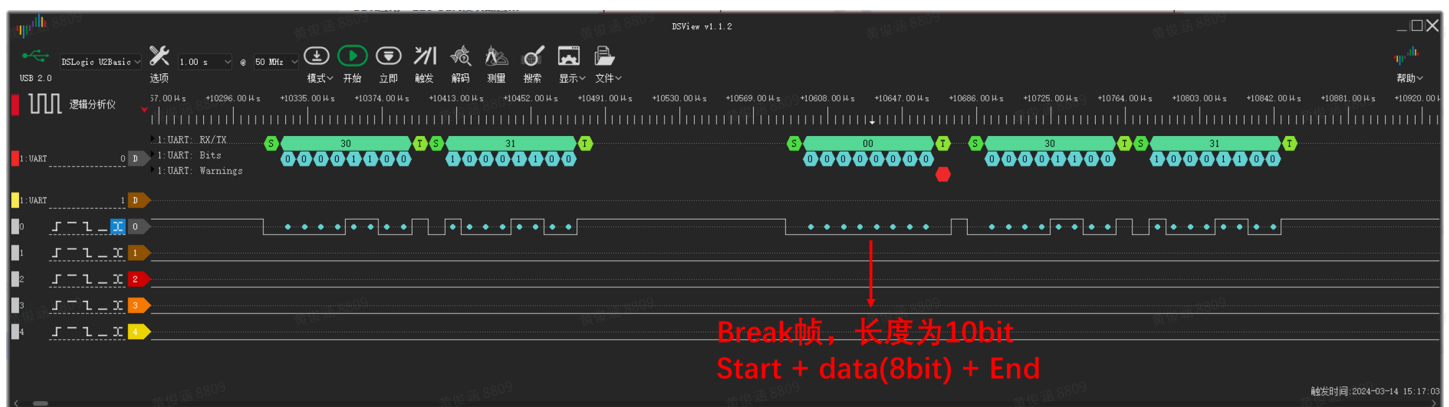
将LINBRK寄存器位置1

再次发送将是Break + Data

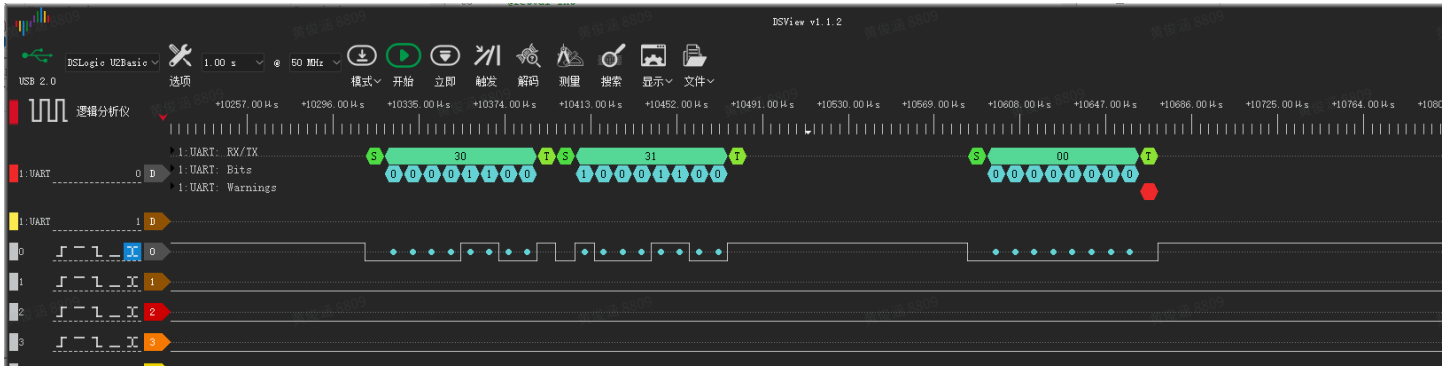
此时该参数为发送的Data数量，不包含Break在内

3.1.2 测试结果

下图为UART_BREAK_LENGTH为2时候的发送情况，第一帧为正常发送两个字节数据{0x30, 0x31}，第二帧则包含1个Break+2个Data{0x30, 0x31}



下图为UART_BREAK_LENGTH为0时候的发送情况，第一帧为正常发送两个字节数据{0x30, 0x31}，第二帧则只有1个Break

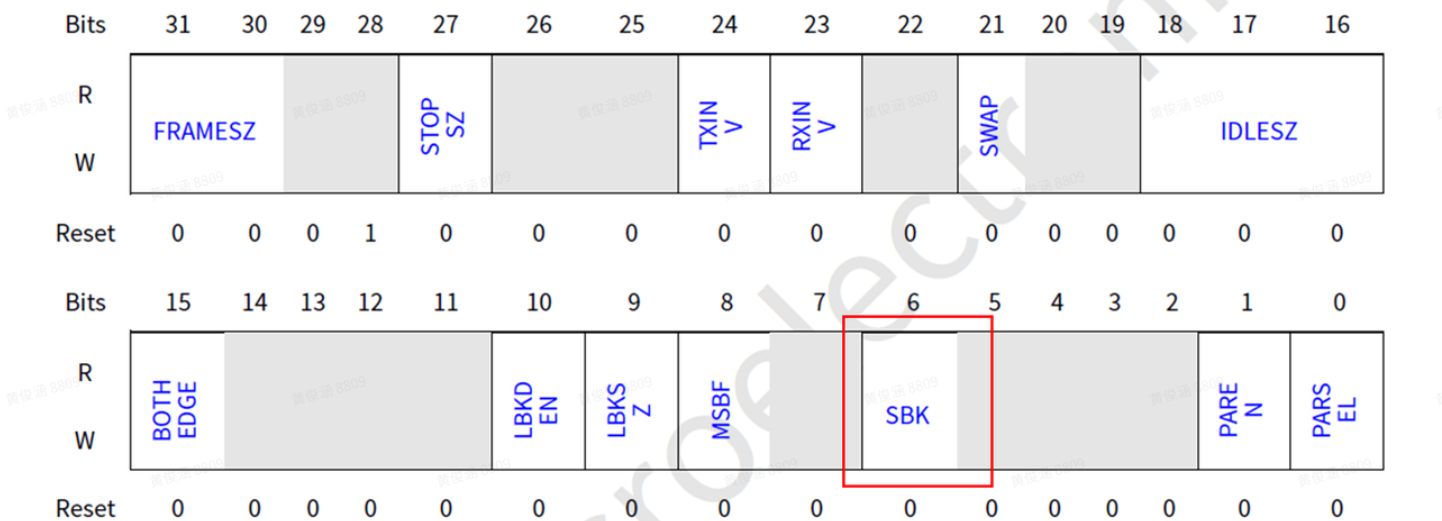


3.2 方式二：发送可变长度Break

将SBK寄存器位置1将持续发送低电平，直到用户主动清0，用户可以用延时或者定时器去控制低电平时间，该方式一般用于Break位长较长时使用

17.3.1.3 UART CTRL1 Register

Offset: 8h



Field	Function
SBK	Send LIN break out, when SBK bit set, UART will send out LIN break continuously until SBK bit cleared. 0b - Normal mode 1b - Send out continuous LIN break

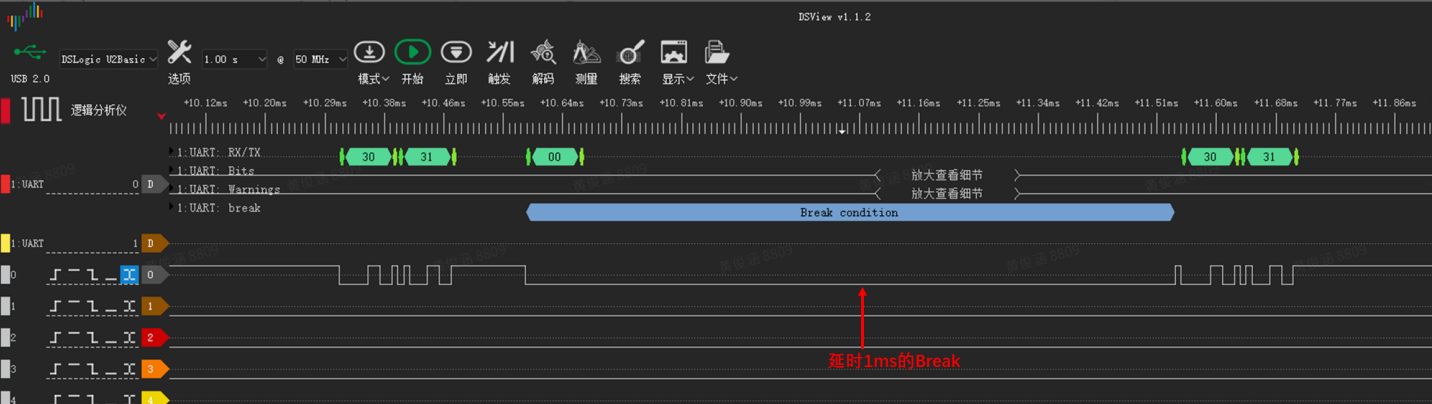
3.2.1 代码实现

Note: SBK寄存器位不会自动清除，需要用户控制什么时候清除，当SBK位置1时电平拉低，SBK位清0时电平被拉高，变量UART_BREAK_LENGTH控制Break发送完后紧接着发多少个字节的数据

```
81
82 UART_DRV_SendData(UART_INSTANCE, g_uartTxBuffer, UART_TX_LENGTH); // 发送只有两个字节的数据帧
83 /* Waiting for transmission to complete */ // 等待发送完成
84 while (UART_DRV_GetTransmitStatus(UART_INSTANCE, &g_bytesRemaining){}
85
86 UART0->CTRL1 |= UART_CTRL1_SBK(1); // 将SBK寄存器位置1
87 UART_DRV_SendData(UART_INSTANCE, g_uartTxBuffer, UART_BREAK_LENGTH);
88
89 OSIF_TimeDelay(1); // 延时1ms
90 // 此时该参数为发送的Data数量，不包含Break在内
91 UART0->CTRL1 &= (~UART_CTRL1_SBK(1)); // 将SBK寄存器位1清0
92 while (UART_DRV_GetTransmitStatus(UART_INSTANCE, &g_bytesRemaining){} // 等待发送完成
93 /* USER CODE END 2 */
```

3.2.2 测试结果

下图第一帧为正常发送两个字节数据{0x30, 0x31}，第二帧为发送一个1ms的Break+两个字节数据{0x30, 0x31}



文档历史

版本号	日期	修订记录
V1.0	2024.03.15	初始版本