SDK应用_M系列Uart模块配置及应用（一）

版本

Config Tool Version：2.0.2

ME0 SDK version：1.1.1

1. 前言

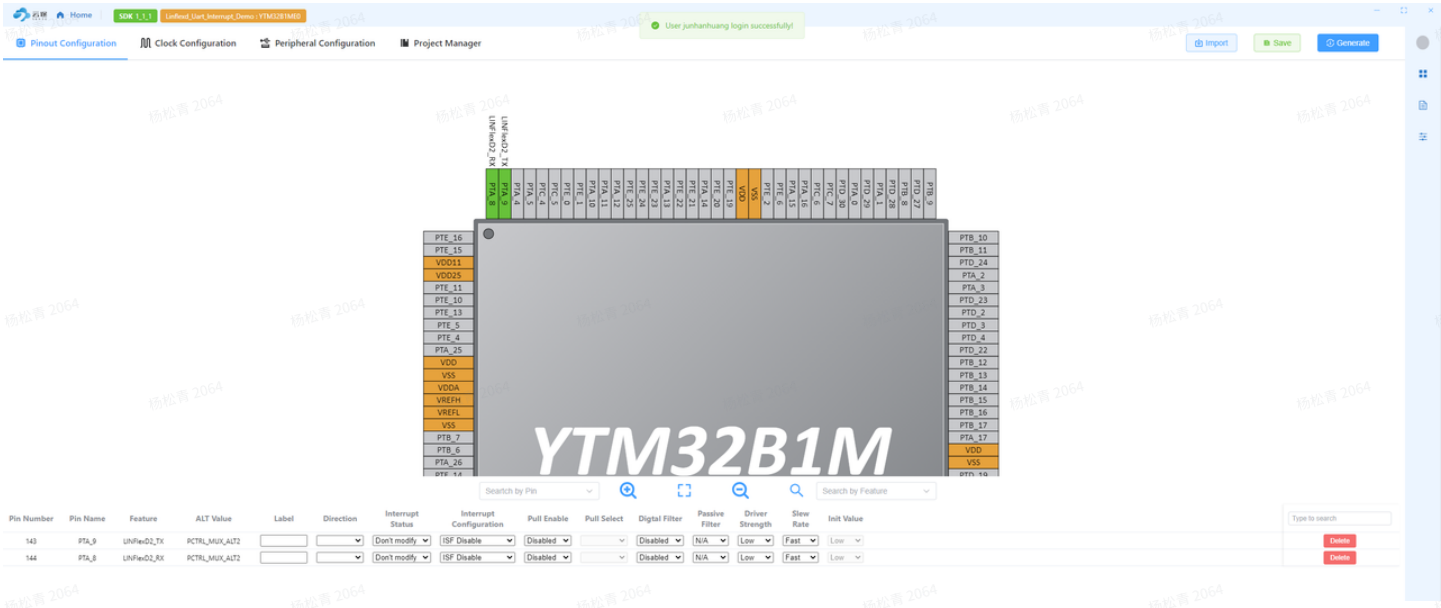
本文基于YTM32B1ME微控制器，**在M系列中，LINFLexD模块既可以作为LIN功能也可以作为UART功能**，简要介绍了LINFLexD外设模块配套SDK驱动的用法。文中使用YCT工具在图形界面中配置LINFLexD驱动，生成LINFLexD驱动的配置参数。介绍了LINFLexD的中断方式、阻塞方式、轮询方式的典型应用。具体介绍了LINFLexD驱动模块中常用的API。

2. 基础配置

2.1 引脚配置

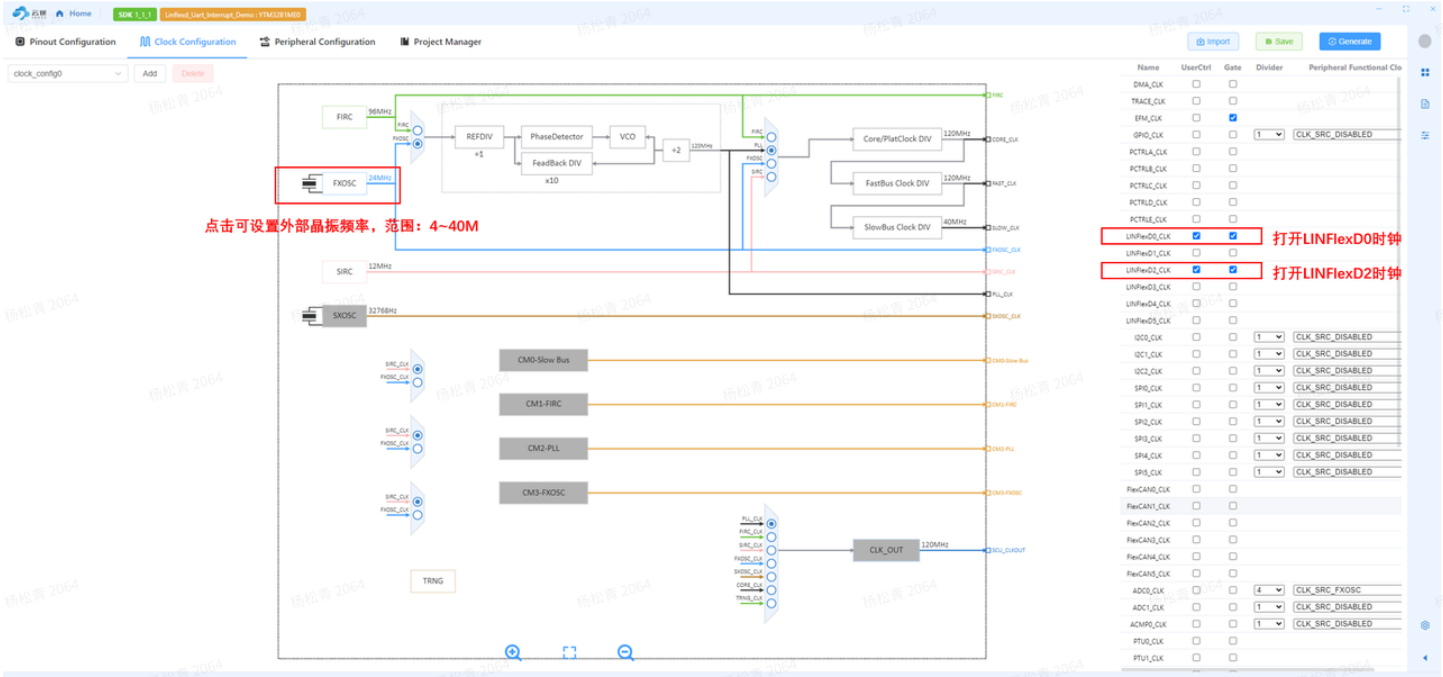
本示例将使用LINFlexD2进行演示

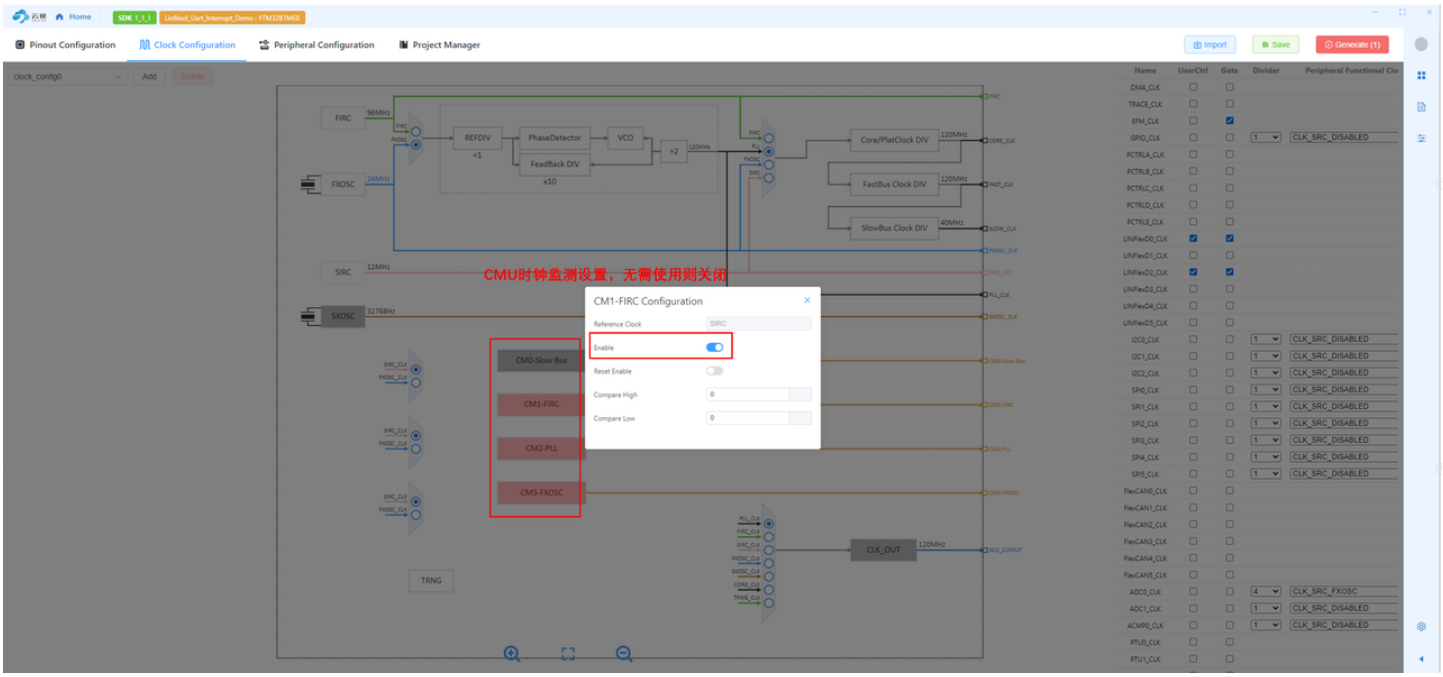
选择LINFlexD2的发送和接收引脚



2.2 时钟配置

通讯模块建议使用外部晶振作为时钟源，云途开发板外部晶振为24M

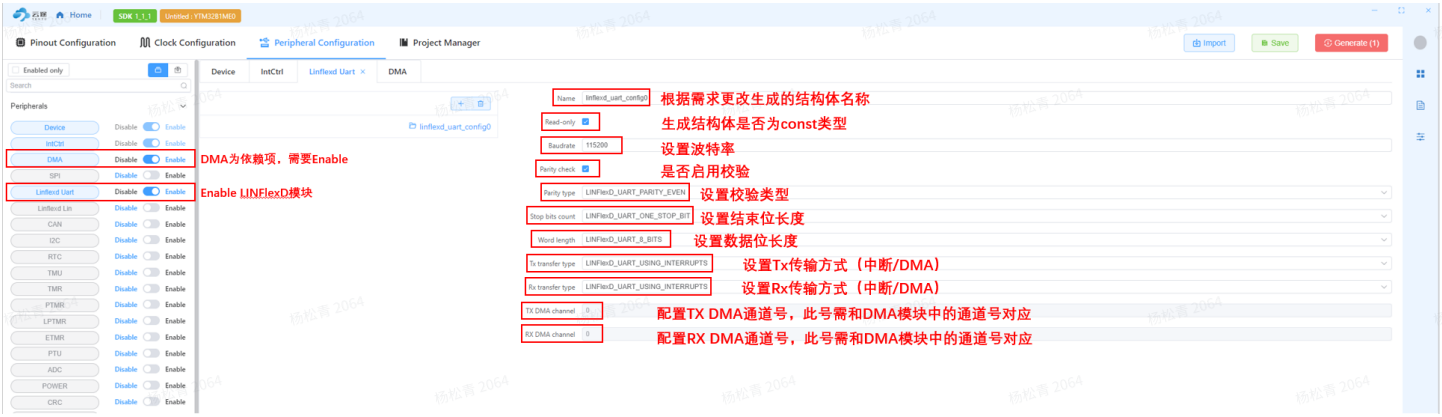




Note：ME0 LINFlexD模块不支持用户设置时钟源，打开时钟将自动设置为FAST_BUS_CLK

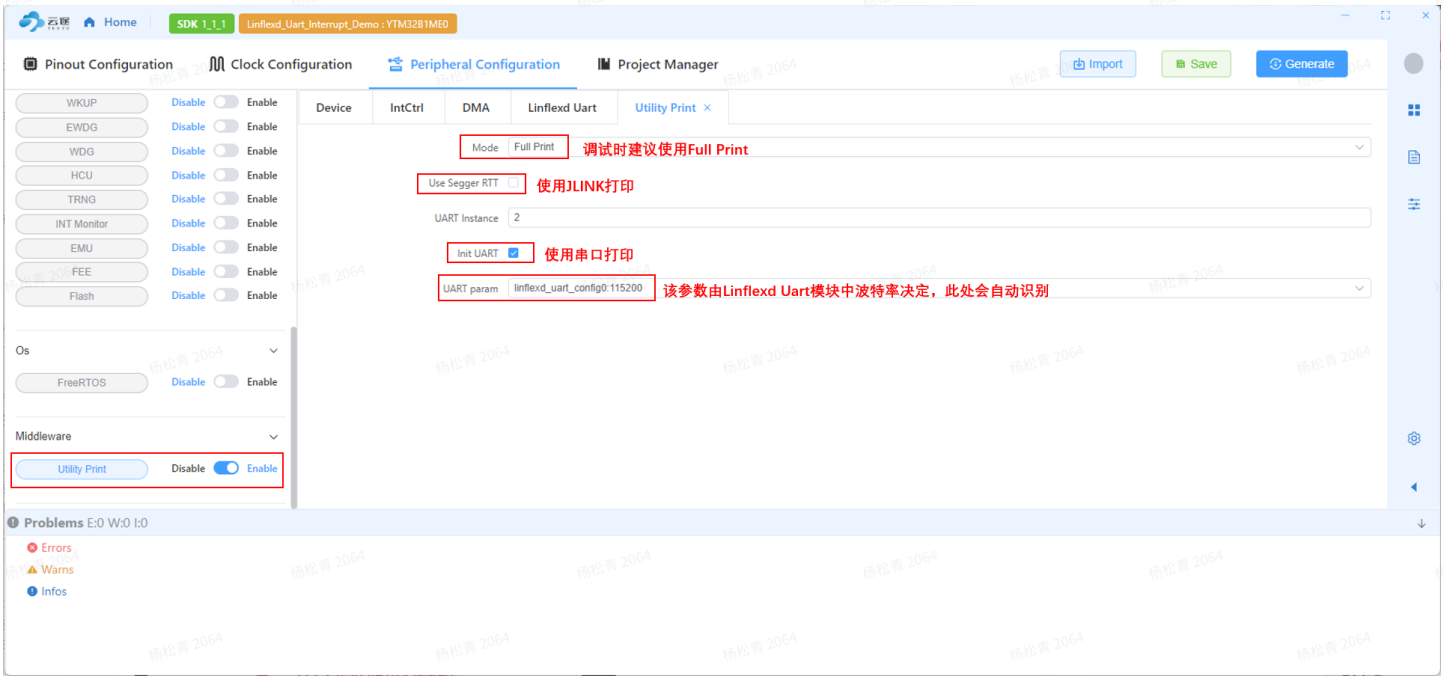
MD1 LINFlexD模块支持用户设置时钟源

2.3 LINFlexD模块配置



2.4 Utility Print模块配置（选用）

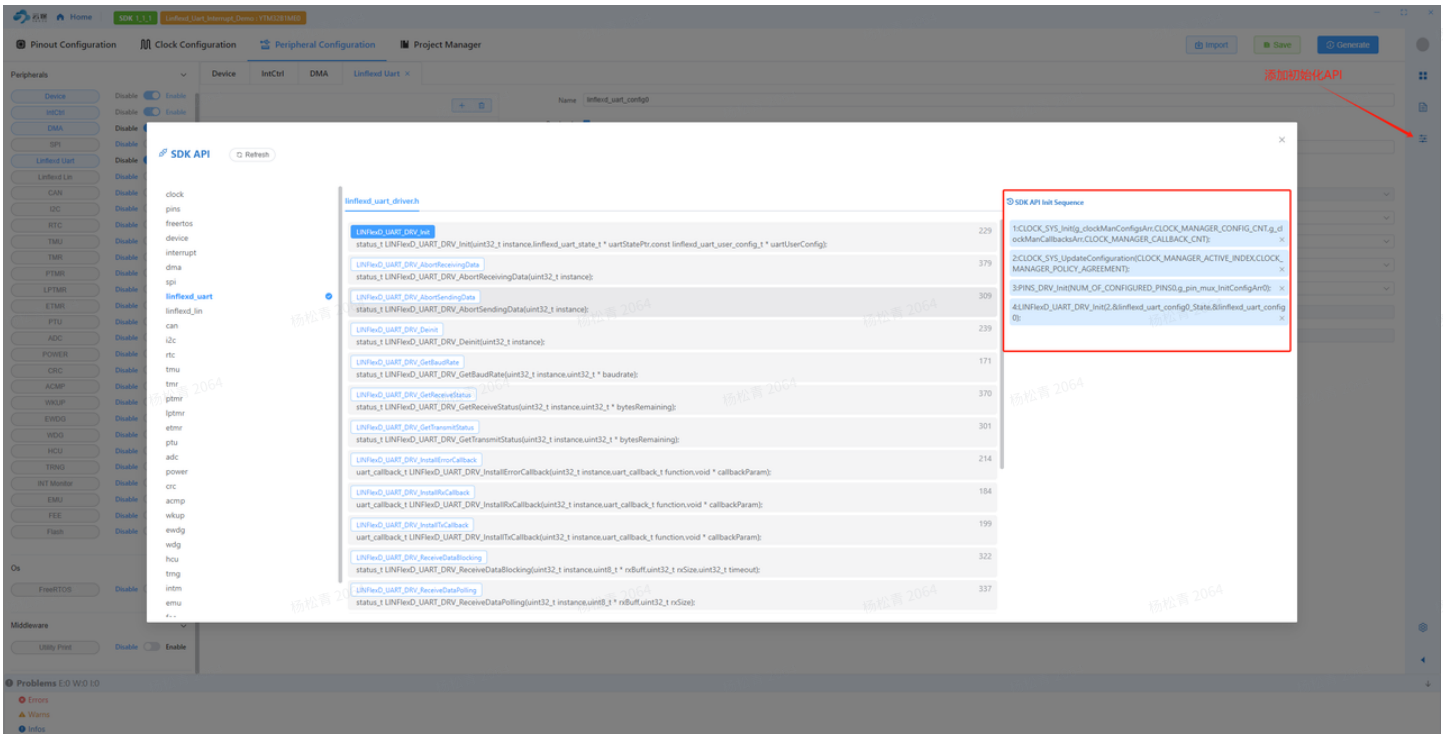
Utility Print模块用于打印输出，若调试时有打印日志需求需enable此模块



- **Mode：**Mode中可选Full Print和Lite Print。Full Print输出的信息更为详细和全面，比如浮点或者格式化的一些信息，这种模式通常用于深度调试和问题排查，以帮助开发人员深入理解代码执行过程中的各个方面。Lite Print简略输出模式输出的信息更为精简和紧凑，这可以减少输出的量，从而减小对系统性能的影响，但是该模式可打印信息的类型较少，某些信息无法打印。
- 在单片机串口资源全部被占用了，又有串口调试需求的，可以选择Use Segger RTT，RTT由JLINK打印

2.5 添加初始化API

- 添加时钟初始化函数
- 添加引脚初始化函数
- 添加LINFlexD初始化函数



2.6 生成工程

3. 生成代码示例

示例分别介绍UART中断方式、阻塞方式、轮询方式的使用方法，三种方式均以成功收发如下数据为目标进行演示

MCU向上位机周期性发送{0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}

MCU接收来自上位机的数据{0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17}

示例最后将使用逻辑分析仪捕捉总线上的数据进行验证

Note：用户手动添加的函数必须写在BEGIN和END段之间，否则使用YT Config Tool重新生成时用户代码会被覆盖

3.1 中断方式

介绍中断方式的回调函数安装及收发实现方法

3.1.1 定义参数

宏定义UART的instance和接收字节数量，该示例使用UART instance为2

定义SendData数组用于发送，定义RecData数组用于接收数据

定义Tx和Rx完成标志位，用于在回调函数中刷新状态

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define PRINTF_UART (2) /* UART instance for PRINTF */
#define SIZE        (7) /* Array size */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */
uint8_t SendData[SIZE] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
uint8_t RecData[SIZE];
volatile bool rxComplete = false;
volatile bool txComplete = false;
/* USER CODE END PV */
```

3.1.2 定义回调函数

手动安装Tx和Rx的回调函数，此示例仅在回调函数中刷新状态信息操作

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void RxBuff_Callback(void *LINFLexDState, uart_event_t event, void *userData)
{
    (void)LINFLexDState;
    (void)userData;
    if (event == UART_EVENT_END_TRANSFER)
    {
        rxComplete = true;
    }
}

void TxBuff_Callback(void *LINFLexDState, uart_event_t event, void *userData)
{
    (void)LINFLexDState;
    (void)userData;
    if (event == UART_EVENT_TX_EMPTY)
    {
        txComplete = true;
    }
}
```

完成Tx或者Rx后将状态标志位刷新为true
以便主循环中轮询标志位

```c
int main(void)
{
    /* USER CODE BEGIN 1 */
    status_t status = STATUS_SUCCESS;
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    LINFlexD_UART_DRV_InstallRxCallback(PRINTF_UART, RxBuff_Callback, NULL);
    LINFlexD_UART_DRV_InstallTxCallback(PRINTF_UART, TxBuff_Callback, NULL);
    /* USER CODE END 2 */
```

注册Rx和Tx回调函数

3.1.3 循环收发

在while(1)中循环进行UART中断收发测试

Note：LINFlexD_UART_DRV_ReceiveData函数调用一次即接收一次，若要多次接收，则需多次调用

```c
int main(void)
{
    /* USER CODE BEGIN 1 */
    status_t status = STATUS_SUCCESS;
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    LINFlexD_UART_DRV_InstallRxCallback(PRINTF_UART, RxBuff_Callback, NULL);
    LINFlexD_UART_DRV_InstallTxCallback(PRINTF_UART, TxBuff_Callback, NULL);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        status |= LINFlexD_UART_DRV_ReceiveData(PRINTF_UART, RecData, SIZE);
        while (!rxComplete);    // wait for receive complete
        status |= LINFlexD_UART_DRV_SendData(PRINTF_UART, SendData, SIZE);
        while(!txComplete);     // wait for send complete
        txComplete = false;
        rxComplete = false;
        OSIF_TimeDelay(10);

        if (status != STATUS_SUCCESS)
        {
            break;
        }
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
    }
    return status;
    /* USER CODE END 3 */
}
```

在收发完成后重新将Tx和Rx状态标志刷成false

3.2 阻塞方式（Blocking）

阻塞方式相较于中断方式多了一个超时时间参数，时间单位：毫秒，超过设定时间即跳出，示例中为超时1000ms

Note：阻塞方式同样会起中断，同样可以进回调函数，与中断方式最大的区别在于中断为异步，阻塞会死等直到收发完成或者超时

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    status_t status = STATUS_SUCCESS;
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    LINFlexD_UART_DRV_InstallRxCallback(PRINTF_UART, RxBuff_Callback, NULL);
    LINFlexD_UART_DRV_InstallTxCallback(PRINTF_UART, TxBuff_Callback, NULL);
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        LINFlexD_UART_DRV_ReceiveDataBlocking(PRINTF_UART, RecData, SIZE, 1000);
        LINFlexD_UART_DRV_SendDataBlocking(PRINTF_UART, SendData, SIZE, 1000);
        OSIF_TimeDelay(10);

        if (status != STATUS_SUCCESS)
        {
            break;
        }
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
    }
    return status;
    /* USER CODE END 3 */
}
```

3.3 轮询方式（Polling）

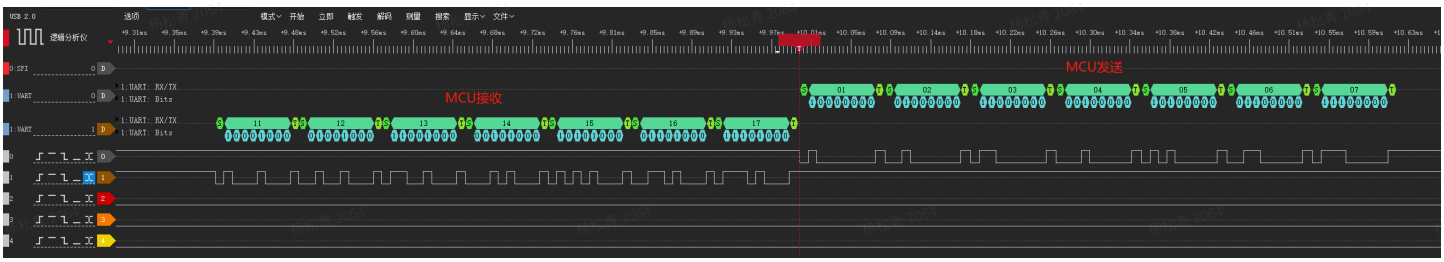
轮询方式不会触发中断，也无超时机制，会一直等待直到收发完成才会继续运行

```c
int main(void)
{
    /* USER CODE BEGIN 1 */
    status_t status = STATUS_SUCCESS;
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        LINFlexD_UART_DRV_ReceiveDataPolling(PRINTF_UART, RecData, SIZE);
        LINFlexD_UART_DRV_SendDataPolling(PRINTF_UART, SendData, SIZE);

        if (status != STATUS_SUCCESS)
        {
            break;
        }
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
    }
    return status;
    /* USER CODE END 3 */
}
```

3.4 测试结果

中断方式、阻塞方式、轮询方式测试结果均为如下所示：

图中为捕获的TX和RX线上数据，测量TX和RX线上数据与预期数据一致



4. UART主要API介绍

4.1 UART初始化函数

代码块

```
status_t LINFlexD_UART_DRV_Init(uint32_t instance, linflexd_uart_state_t *
uartStatePtr, const linflexd_uart_user_config_t * uartUserConfig)
```

- instance：UART通道索引号；

- uartStatePtr：UART状态机变量；
- uartUserConfig：UART配置结构体。

4.2 UART反初始化函数

代码块

```
status_t LINFlexD_UART_DRV_Deinit(uint32_t instance)
```

- instance：UART通道索引号。

4.3 注册RX/TX/ERROR回调函数

代码块

```
uart_callback_t LINFlexD_UART_DRV_InstallRxCallback(uint32_t instance,
uart_callback_t function, void * callbackParam)
uart_callback_t LINFlexD_UART_DRV_InstallTxCallback(uint32_t instance,
uart_callback_t function, void * callbackParam)
uart_callback_t LINFlexD_UART_DRV_InstallErrorCallback(uint32_t instance,
uart_callback_t function, void * callbackParam)
```

- instance：UART通道索引号；
- function：用户自定义注册的回调函数名称；
- callbackparam：需要传到状态机的用户参数。

4.4 UART阻塞模式 发送/接收

代码块

```
status_t LINFlexD_UART_DRV_SendDataBlocking(uint32_t instance, const uint8_t *
txBuff, uint32_t txSize, uint32_t timeout)
status_t LINFlexD_UART_DRV_ReceiveDataBlocking(uint32_t instance, uint8_t *
rxBuff, uint32_t rxSize, uint32_t timeout)
```

- instance：UART通道索引号；
- txBuff / rxBuff：发送/接收buffer；
- txSize / rxSize：发送/接收字节数；
- timeout：设置超时时间，单位：毫秒。

4.5 UART轮询模式 发送/接收

代码块

```
status_t LINFlexD_UART_DRV_SendDataPolling(uint32_t instance, const uint8_t *
txBuff, uint32_t txSize)
status_t LINFlexD_UART_DRV_ReceiveDataPolling(uint32_t instance, uint8_t *
rxBuff, uint32_t rxSize)
```

- instance：UART通道索引号；
- txBuff / rxBuff：发送/接收buffer；
- txSize / rxSize：发送/接收字节数。

说明：该模式不会触发UART中断

4.6 UART中断模式 发送/接收

代码块

```
status_t LINFlexD_UART_DRV_SendData(uint32_t instance, const uint8_t * txBuff,
uint32_t txSize)
status_t LINFlexD_UART_DRV_ReceiveData(uint32_t instance, uint8_t * rxBuff,
uint32_t rxSize)
```

- instance：UART通道索引号；
- txBuff / rxBuff：发送/接收buffer；
- txSize / rxSize：发送/接收字节数。

4.7 UART获取发送/接收状态

代码块

```
status_t LINFlexD_UART_DRV_GetTransmitStatus(uint32_t instance, uint32_t *
bytesRemaining)
status_t LINFlexD_UART_DRV_GetReceiveStatus(uint32_t instance, uint32_t *
bytesRemaining)
```

- instance：UART通道索引号；
- bytesRemaining：将剩余传输字节数返回至该变量。

说明：此功能返回上一次UART发送/接收是否完成。当执行**非阻塞**发送/接收时，用户可以调用此函数来确定当前发送/接收进度的状态：正在进行或已完成。此外，如果发送/接收仍在进行中，则用户可以获得当前已发送/接收的字节数。

4.8 UART停止发送/接收

代码块

```
status_t LINFlexD_UART_DRV_AbortSendingData(uint32_t instance)
status_t LINFlexD_UART_DRV_AbortReceivingData(uint32_t instance)
```

- instance：UART通道索引号；

说明：该函数提前终止**非阻塞**发送/接收

文档历史

版本号	日期	修订记录
V1.0	2024.01.12	初始版本