在省电模式下 LPUART 接收不定长度数据

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码实现在省电模式下，通过 LPUART 接收不定长度的数据后，再唤醒 MCU 进行数据处理。实现 M2L31 低功耗 LPUART 运行特征。
BSP 版本	M2L31_Series_BSP_CMSIS_V3.00.000
开发平台	NuMaker-M2L31KI V2.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例代码将示范如何使用 M2L31 的 LPUART 和 LPPDMA 在省电模式下接收不定长度的数据后唤醒 MCU，并对数据进行处理，然后再次进入省电模式，以此重复。实现降低系统的功耗。

1.1 原理

本范例代码主要用到三个外设：LPUART0、LPPDMA0 以及 UART0。

- LPUART0 设置为自动操作模式会在省电模式下接收数据；同时启动 LPUART RX 触发 LPPDMA，当 LPUART 收到数据后会触发 LPPDMA 将数据搬运到 LPSRAM；启动了总线空闲超时唤醒功能，当开始接收第一个数据时超时计数器开始计数，收到新数据会重置计数器，计数器超时后会唤醒 MCU，利用这个功能可以实现接收不定长度的数据后唤醒系统。LPUART 自动操作模式接收数据的功能框图如图 1-1 所示。

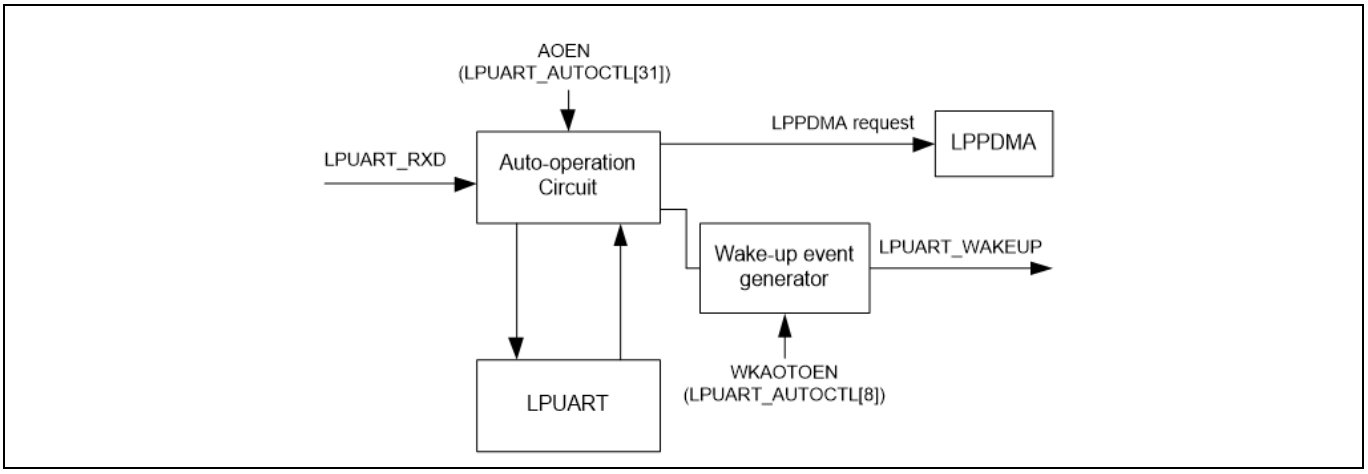

图 1-1 LPUART 自动模式接收数据

- LPPDMA 可以在省电模式下搬运数据。配置基本操作模式，将 LPUART RX 收到的数据搬运到指定的 LPSRAM 里，完成设定长度的数据搬运后会停止 LPPDMA。

1.2 执行结果

本范例代码的执行结果如图 1-2 所示。PC 上打开两个串口工具。左图是连接的 UART0 用于打印调试信息；右图是连接的 LPUART 用于发送数据给 MCU。操作步骤如下：

1. PC 通过串口工具给 MCU 的 UART0 发送任意数据，开始测试。
2. MCU 进入省电模式。
3. PC 通过串口工具给 MCU 的 LPUART 发送 26 字节数据。
4. MCU 收到 26 字节后唤醒，并打印收到的数据。
5. MCU 再次进入省电模式。
6. PC 通过串口工具给 MCU 的 LPUART 发送 1066 字节数据。
7. MCU 收到数据个数超过 buffer 大小，打印警告信息。
8. MCU 唤醒并打印收到的数据。

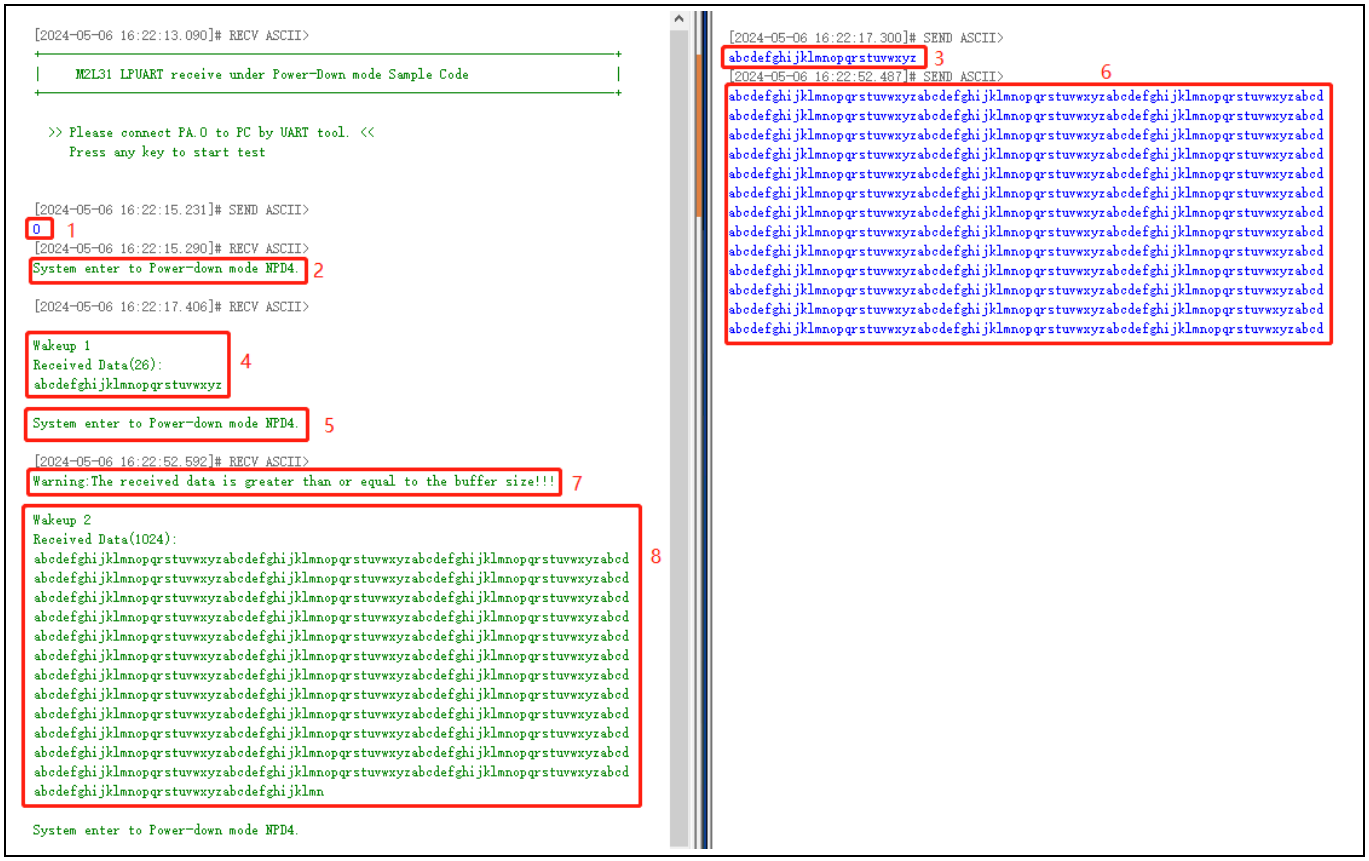

图 1-2 执行结果

NPD4 模式不接收数据时的平均电流 9 μA，NPD4 模式下 LPUART 接收数据时平均电流 244 μA。

2. 代码介绍

定义 LPPDMA 通道、数据接收 buffer 的大小、省电模式的等级。这里 buffer 设置 1024 字节，以免一次收到数据量过大而丢失。

```
#define LPUART_RX_LPPDMA_CH  0     /* LPPDMA channel for LPUART RX */
#define RXBUFSIZE            1024  /* Buffer size */
#define POWER_DOWN_MODE      CLK_PMUCTL_PDMSEL_NPD4   /* Power-down mode */
```

指定数据接收 buffer 的地址在 LPSRAM 里

```
#if (defined(__GNUC__) && !defined(__ARMCC_VERSION))
uint8_t g_u8RecData[RXBUFSIZE] __attribute__((section(".lpSram")));
#else
uint8_t g_u8RecData[RXBUFSIZE] __attribute__((section(".ARM.__at_0x28000000")));
#endif
```

LPPDMA 初始化

配置 LPPDMA 通道 0 搬运 LPUART RX 接收到的数据。采用基本操作模式，完成设定长度的数据搬运后会 LPPDMA 会停止。

```
void LPPDMA_RX_Init(void)
{
/*---------------------------------------------------------------------------------------------------
        LPPDMA transfer configuration:
        Channel = 0
        Operation mode = basic mode
        Request source = LPUART0 RX (peripheral to memory)
        transfer done interrupt = enable
        Transfer count = RXBUFSIZE
        Transfer width = 8 bits (one byte)
        Source address = LPUART0->DAT
        Source address increment size = 0 (fixed)
        Destination address = g_u8RecData
        Destination address increment size = 8 bits (one byte)
        Transfer type = Single transfer
        Total transfer length = RXBUFSIZE * 8 bits
---------------------------------------------------------------------------------------------------*/
    /* Open Channel 0 */
    LPPDMA_Open(LPPDMA0,1 << LPUART_RX_LPPDMA_CH);

    /* Transfer count is RXBUFSIZE, transfer width is 8 bits(one byte) */
    LPPDMA_SetTransferCnt(LPPDMA0,LPUART_RX_LPPDMA_CH, LPPDMA_WIDTH_8, RXBUFSIZE);

    /* Set address */
    LPPDMA_SetTransferAddr(LPPDMA0,
             LPUART_RX_LPPDMA_CH,
             (uint32_t)&(LPUART0->DAT),/* Source address */
             LPPDMA_SAR_FIX,           /* Source address fixed */
             (uint32_t)g_u8RecData,    /* Destination address */
             LPPDMA_DAR_INC);          /* Destination address incremented, size = 1byte */

    /* Request source is LPUART0 RX to memory */
    LPPDMA_SetTransferMode(LPPDMA0,LPUART_RX_LPPDMA_CH, LPPDMA_LPUART0_RX, FALSE, 0);
```

```
    /* Transfer type is Single transfer */
    LPPDMA_SetBurstType(LPPDMA0,LPUART_RX_LPPDMA_CH, LPPDMA_REQ_SINGLE, 1);

    /* Enable transfer done interrupt */
    LPPDMA_EnableInt(LPPDMA0, LPUART_RX_LPPDMA_CH, LPPDMA_INT_TRANS_DONE);

    /* Enable LPPDMA0 IRQ */
    NVIC_EnableIRQ(LPPDMA0_IRQn);
}
```

LPUART 初始化

设置 LPUART 的波特率为 9600 bps，使能自动操作模式、RX 触发 LPPDMA、总线空闲超时唤醒和接收超时中断。在省电模式下 LPUART 收到数据会触发 LPPDMA 搬运到 LPSRAM 里。当收到第一个数据，超时计数器开始计数；收到新数据，计数器复位；一旦计数器超时会唤醒 MCU、并产生中断。超时时间在寄存器 TOUT[7:0](TOIC)里设置，TOIC 的值对应的时间和波特率有关。例如 TOIC = 40、波特率= 9600 bps。40 代表传输 40 位需要的时间，以波特率 9600 bps 来计算就是 4.17 ms。如果数据格式 = 1 位起始位 + 8 位数据+1 位停止位(无奇偶校验位)，TOIC = 40 相当于 4 个字节。为了避免在接收一个字节期间立即产生超时中断，建议 TOIC 设置在 40~255 之间。

```
void LPUART0_Init(void)
{
    /* Reset Low Power UART0 */
    SYS_ResetModule(LPUART0_RST);

    /* Configure Low Power UART0 and set Low Power UART0 Baudrate */
    LPUART_Open(LPUART0, 9600);

    /* Keep LPUART clock at power-down mode NPD3/NDP4 */
    LPSCC->CLKKEEP0 |= LPSCC_CLKKEEP0_LPUART0KEEP_Msk;

    /* Set Automatic Operation Enable */
    LPUART0->AUTOCTL |= LPUART_AUTOCTL_AOEN_Msk;

    /* Enable LPUART PDMA RX */
    LPUART0->INTEN |= LPUART_INTEN_RXPDMAEN_Msk;

    /* Enable bus idle time-out(Mode 1) and set time-out counter */
    LPUART0->TOUT =  255 | LPUART_TOUT_BITOMEN_Msk;

    /* Enable time-out counter */
    LPUART0->INTEN |= LPUART_INTEN_TOCNTEN_Msk;

    /* Enable Bus Idle Time-out Wake-up */
    LPUART0->AUTOCTL |= LPUART_AUTOCTL_WKAOTOEN_Msk;

    /* Enable RX Time-out interrupt */
    LPUART_EnableInt(LPUART0, LPUART_INTEN_RXTOIEN_Msk);
```

```
/* Enable LPUART0 IRQ */
    NVIC_EnableIRQ(LPUART0_IRQn);
}
```

记录传输数据个数

MCU 唤醒后，在中断函数里计算传输数据的个数。这里存在两种情况:

- LPPDMA 没有传完，传输个数 = LPPDMA 设定个数 - LPPDMA 剩余个数。

```
void LPUART0_IRQHandler(void)
{
    if (LPUART_GET_INT_FLAG(LPUART0, LPUART_INTSTS_HWTOINT_Msk))
    {
        /* Record the number of data */
        g_u32cnt = RXBUFSIZE - LPPDMA_GET_TRANS_CNT(LPPDMA0, LPUART_RX_LPPDMA_CH) - 1;
    }

    /* Clear interrupt flag */
    LPUART0->INTSTS = LPUART0->INTSTS;
}
```

- LPPDMA 传输完成，传输个数 = LPPDMA 设定个数。

注意：因为设置的 LPPDMA 搬运数据长度是 1024 字节，如果传输的数据超过 1024 字节会导致数据丢失。可以修改宏定义 RXBUFSIZE 的值改变最大能接收的数据大小。

```
void LPPDMA0_IRQHandler(void)
{
  volatile uint32_t u32regISR;

  u32regISR = LPPDMA0->INTSTS;

  if ((u32regISR & LPPDMA_INTSTS_TDIF_Msk) == LPPDMA_INTSTS_TDIF_Msk)/* transfer done */
  {
    /* Record the number of data */
    g_u32cnt = RXBUFSIZE;

    printf("Warning:The received data is greater than or equal to the buffer size!!!\n");
  }

  /* Clear interrupt flag */
  LPPDMA0->TDSTS = LPPDMA0->TDSTS;
}
```

唤醒后停止传输

主要是停止 LPPDMA 和禁止 LPUART 触发 LPPDMA 以及停止 LPUART 总线空闲超时计数器。

保证下次进入省电后 LPPDMA 和 LPUART 的总线空闲超时计数器会重新计数。

```
void Transfer_Stop(void)
{
    /* Disable LPUART PDMA RX and time-out counter */
    LPUART0->INTEN &= ~(LPUART_INTEN_RXPDMAEN_Msk | LPUART_INTEN_TOCNTEN_Msk);

    /* Stop LPPDMA */
    LPPDMA0->LPDSCT[1].CTL &= ~LPPDMA_DSCT_CTL_OPMODE_Msk;
}
```

重新配置传输

是为下一次进入省电后正常接收数据和唤醒做准备。LPPDMA 不需要重新配置所有寄存器，主要需要配置传输个数和操作模式。LPUART 复位后重新配置。

```
void Transfer_Recfg(void)
{
    ClearBuf((uint32_t)g_u8RecData, RXBUFSIZE, 0xFF);

    g_u32cnt = 0;

    /* Restart LPPDMA channel */
    /* Transfer count is RXBUFSIZE, transfer width is 8 bits(one byte) */
    LPPDMA_SetTransferCnt(LPPDMA0, LPUART_RX_LPPDMA_CH, LPPDMA_WIDTH_8, RXBUFSIZE);

    /* Request source is LPUART RX to memory */
    LPPDMA_SetTransferMode(LPPDMA0, LPUART_RX_LPPDMA_CH, LPPDMA_LPUART0_RX, FALSE, 0);

    /* Restart LPUART */
    LPUART0_Init();
}
```

系统初始化

注意 NPD4 模式下 HCLK1 不能超过 4 MHz，所以选择 MIRC 作为时钟源。

```
void SYS_Init(void)
{
  /* Unlock protected registers */
  SYS_UnlockReg();

  /* Enable External XTAL (4~24 MHz) */
  CLK_EnableXtalRC(CLK_PWRCTL_HIRCEN_Msk);

  /* Switch HCLK clock source to HIRC */
  CLK->CLKSEL0 = (CLK->CLKSEL0 & ~CLK_CLKSEL0_HCLK0SEL_Msk) |
                 CLK_CLKSEL0_HCLKSEL_HIRC48M ;

  /* Switch HCLK1 clock source to MIRC */
  CLK->CLKSEL0 = (CLK->CLKSEL0 & ~CLK_CLKSEL0_HCLK1SEL_Msk) |
                 CLK_CLKSEL0_HCLK1SEL_MIRC ;

  /* Select UART0 clock source is HIRC and UART module clock divider as 1 */
  CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL4_UART0SEL_HIRC, CLK_CLKDIV0_UART0(1));

  /* Select LPUART0 clock source is MIRC and LPUART module clock divider as 1*/
```

```
  CLK_SetModuleClock(LPUART0_MODULE, LPSCC_CLKSEL0_LPUART0SEL_MIRC,
                     LPSCC_CLKDIV0_LPUART0(1));

  /* Enable UART0 peripheral clock */
  CLK_EnableModuleClock(UART0_MODULE);

  /* Enable LPUART0 peripheral clock */
  CLK_EnableModuleClock(LPUART0_MODULE);

  /* Enable LPSRAM clock */
  CLK_EnableModuleClock(LPSRAM_MODULE);

  /* LPPDMA Clock Enable */
  CLK_EnableModuleClock(LPPDMA0_MODULE);

  /* Set GPIO mode for unused pins */
  GPIO_Init();

  /* Set GPB multi-function pins for UART0 RXD and TXD */
  Uart0DefaultMPF();

  /* Set PA multi-function pins for LPUART0 TXD and RXD */
  SYS->GPA_MFP0 = (SYS->GPA_MFP0 & ~(SYS_GPA_MFP0_PA0MFP_Msk | SYS_GPA_MFP0_PA1MFP_Msk)) |
                  (SYS_GPA_MFP0_PA0MFP_LPUART0_RXD | SYS_GPA_MFP0_PA1MFP_LPUART0_TXD);

    /* Lock protected registers */
    SYS_LockReg();
}
```

主函数

```
int main(void)
{
    uint32_t i, count = 0;

    /* Init System, peripheral clock, and multi-function I/O */
    SYS_Init();

    /* Init UART0 for printf */
    UART_Init();

    /*---------------------------------------------------------------------------------------*/
    /* LPUART Receives Variable Length Data in Power-down Mode Test                          */
    /* 1. LPUART uses LPPDMA Channel-0 to transfer RX data to g_u8RecData                    */
    /* 2. LPUART RX pin (PA.0) is connected to UART Tool                                     */
    /* 3. System enters power-down mode and UART RX receives data in power-down mode         */
    /* 4. When RX transfer done or timed out and wake-up system                              */
    /* 5. Print the received data through uart0                                              */
    /* 6. Repeat steps 3~5                                                                   */
    /*---------------------------------------------------------------------------------------*/
    printf("+-----------------------------------------------------------------------+\n");
    printf("|     M2L31 LPUART receive under Power-Down mode Sample Code          |\n");
    printf("+-----------------------------------------------------------------------+\n\n");
    printf("  >> Please connect PA.0 to PC by UART tool. << \n");
    printf("     Press any key to start test\n\n");

    getchar();

    /* Clear buffer */
```

```
    ClearBuf((uint32_t)g_u8RecData, RXBUFSIZE , 0xFF);

    /* LPPDMA CH-0 basic Mode RX to receive data */
    LPPDMA_RX_Init();

    /* Init LPUART0 as Auto-operation Mode for test */
    LPUART0_Init();

    while (1)
    {
        /* Reconfigure transmission */
       Transfer_Recfg();

        count++;

        /* Enter power down mode */
        PowerDown_Enter();

        printf("\nWakeup %d\n", count);

        /* Wait for the number of data to be recorded */
        while (g_u32cnt == 0);

       Transfer_Stop();

        /* Add data processing code */
        printf("Received Data(%d):\n", g_u32cnt);

        for (i = 0; i < g_u32cnt; i++)
            printf("%c", g_u8RecData[i]);

        printf("\n\n");
    }
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M2L31_Series_BSP_CMSIS_V3.00.000
- IDE 版本
  - Keil uVersion 5.38.0.0

3.2 硬件需求

- 电路组件
  - NuMaker- M2L31KI V2.1
- 线路示意图
  - 将 UART0 TX (PB.13) pin 通过 UART 转 USB 工具连接到 PC。
  - 将 UART0 RX (PB.12) pin 通过 UART 转 USB 工具连接到 PC 。
  - 将 LPUART RX (PA.0) pin 通过 UART 转 USB 工具连接到 PC。

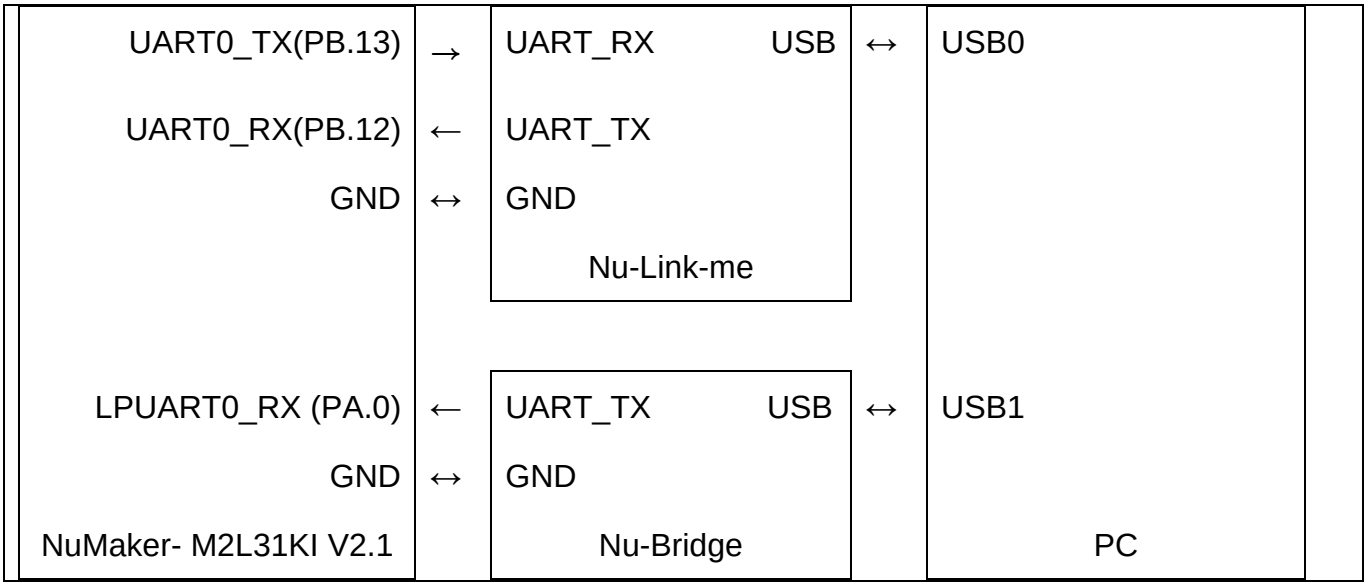

图 3-1 线路示意图

4. 目录信息

EC_ M2L31_LPUART_RX_in_PD_V1.01	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
Proect	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M2L31_LPUART_RX_in_PD.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2025.03.07	1.01	去掉关闭LVR的代码，防止造成上下电问题。
2024.04.19	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.