在省電模式下 LPUART 接收不定長度資料

NuMicro® 32 位元系列微控制器範例代碼介紹

文件訊息

應用簡述	本範例代碼實現在省電模式下，通過 LPUART 接收不定長度的資料後，再喚醒 MCU 進行資料處理。實現 M2L31 低功耗 LPUART 運行特徵。
BSP 版本	M2L31_Series_BSP_CMSIS_V3.00.000
開發平臺	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

本範例代碼將示範如何使用 M2L31 的 LPUART 和 LPPDMA 在省電模式下接收不定長度的資料後喚醒 MCU，並對資料進行處理，然後再次進入省電模式，以此重複。實現降低系統的功耗。

1.1 原理

本範例代碼主要用到三個周邊：LPUART0、LPPDMA0 以及 UART0。

- LPUART0 設置為自動操作模式會在省電模式下接收資料；同時啟動 LPUART RX 觸發 LPPDMA，當 LPUART 收到資料後會觸發 LPPDMA 將資料搬運到 LPSRAM；啟動了匯流排空閒超時喚醒功能，當開始接收第一個資料時超時計數器開始計數，收到新資料會重置計數器，計數器超時後會喚醒 MCU，利用這個功能可以實現接收不定長度的資料後喚醒系統。LPUART 自動操作模式接收資料的功能框圖如圖 1-1 所示。

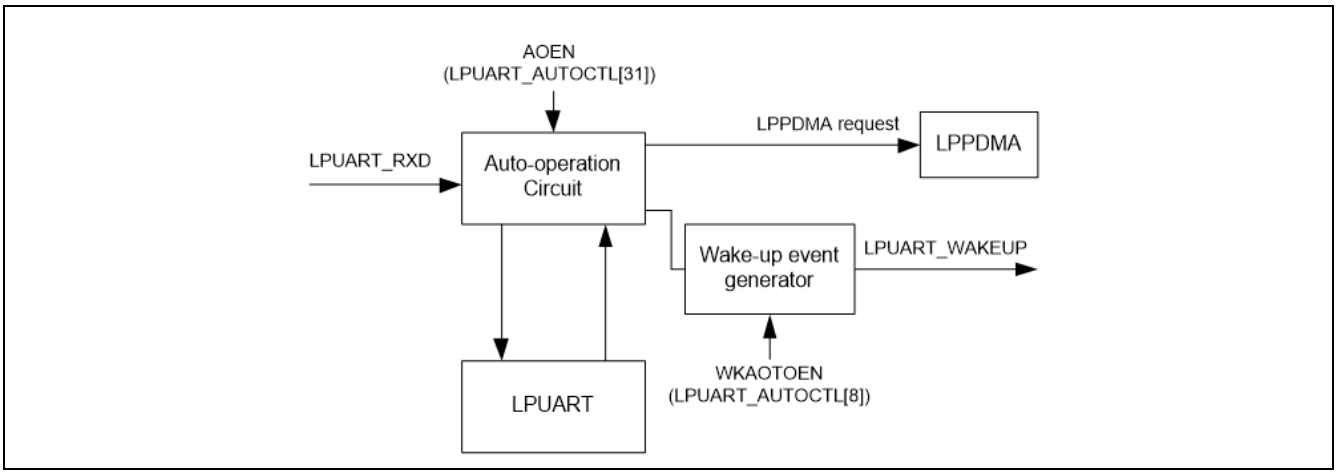

圖 1-1 LPUART 自動模式接收資料

- LPPDMA 可以在省電模式下搬運資料。配置基本操作模式，將 LPUART RX 收到的資料搬運到指定的 LPSRAM 裡，完成設定長度的資料搬運後會停止 LPPDMA。

1.2 執行結果

本範例代碼的執行結果如圖 1-2 所示。PC 上打開兩個串口工具。左圖是連接的 UART0 用於列印調試資訊；右圖是連接的 LPUART 用於發送資料給 MCU。操作步驟如下：

1. PC 通過串口工具給 MCU 的 UART0 發送任意資料，開始測試。
2. MCU 進入省電模式。
3. PC 通過串口工具給 MCU 的 LPUART 發送 26 位元組資料。
4. MCU 收到 26 位元組後喚醒，並列印收到的資料。
5. MCU 再次進入省電模式。
6. PC 通過串口工具給 MCU 的 LPUART 發送 1066 位元組資料。
7. MCU 收到資料個數超過 buffer 大小，列印警告資訊。
8. MCU 喚醒並列印收到的資料。

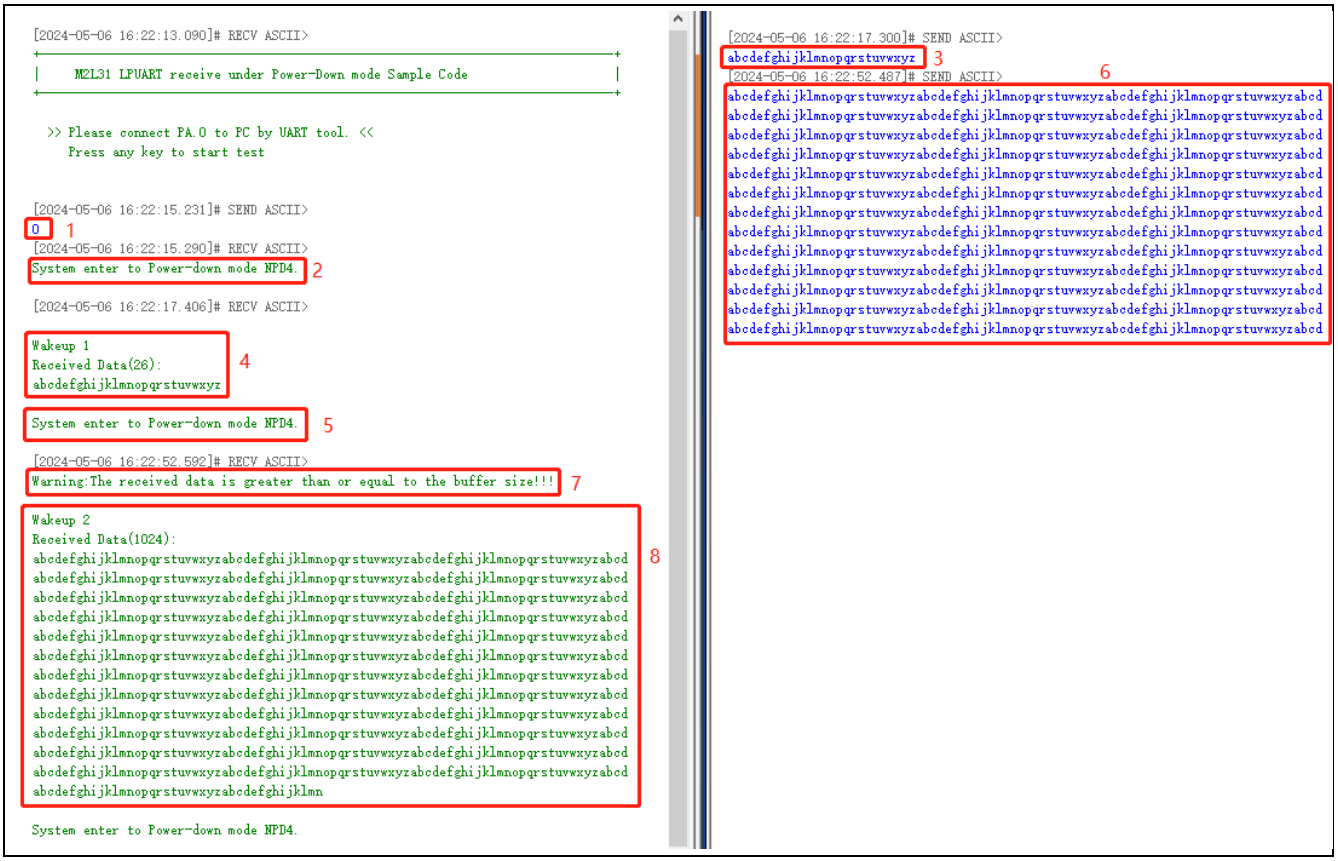

圖 1-2 執行結果

NPD4 模式不接收資料時的平均電流 9 μA，NPD4 模式下 LPUART 接收資料時平均電流 244 μA。

2. 代碼介紹

定義LPPDMA通道、資料接收buffer的大小、省電模式的等級。這裡buffer設置1024位元組，以免一次收到資料量過大而丟失。

```
#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 */
```

指定資料接收 buffer 的位址在 LPSRAM 裡

```
#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 搬運 LPUART RX 接收到的資料。採用基本操作模式，完成設定長度的資料搬運後會 LPPDMA 會停止。

```
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

設置 LPUART 的串列傳輸速率為 9600 bps，使能自動操作模式、RX 觸發 LPPDMA、匯流排空閒超時喚醒和接收超時中斷。在省電模式下 LPUART 收到資料會觸發 LPPDMA 搬運到 LPSRAM 裡。當收到第一個資料，超時計數器開始計數；收到新資料，計數器重定；一旦計數器超時會喚醒 MCU、並產生中斷。超時時間在寄存器 TOUT[7:0](TOIC)裡設置，TOIC 的值對應的時間和串列傳輸速率有關。例如 TOIC = 40、串列傳輸速率= 9600 bps。40 代表傳輸 40 位元需要的時間，以串列傳輸速率 9600 bps 來計算就是 4.17 ms。如果資料格式 = 1 位元起始位元 + 8 位元資料+1 位元停止位元(無同位檢查位元)，TOIC = 40 相當於 4 個位元組。為了避免在接收一個位元組期間立即產生超時中斷，建議 TOIC 設置在 40~255 之間。

```
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);
}
```

記錄傳輸資料個數

MCU 喚醒後，在中斷函數裡計算傳輸資料的個數。這裡存在兩種情況:

- LPPDMA 沒有傳完，傳輸個數 = LPPDMA 設定個數 - LPPDMA 剩餘個數。

```
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 傳輸完成，傳輸個數 = LPPDMA 設定個數。

注意：因為設置的 LPPDMA 搬運資料長度是 1024 位元組，如果傳輸的資料超過 1024 位元組會導致資料丟失。可以修改巨集定義 RXBUFSIZE 的值改變最大能接收的資料大小。

```
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;
}
```

喚醒後停止傳輸

主要是停止 LPPDMA 和禁止 LPUART 觸發 LPPDMA 以及停止 LPUART 匯流排空閒超時計數器。

保證下次進入省電後 LPPDMA 和 LPUART 的匯流排空閒超時計數器會重新計數。

```
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;
}
```

重新配置傳輸

是為下一次進入省電後正常接收資料和喚醒做準備。LPPDMA 不需要重新配置所有寄存器，主要需要配置傳輸個數和操作模式。LPUART 重定後重新配置。

```
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();
}
```

系統初始化

注意 NPD4 模式下 HCLK1 不能超過 4 MHz，所以選擇 MIRC 作為時鐘源。

```
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();
}
```

主函數

```
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. 軟體與硬體需求

3.1 軟體需求

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

3.2 硬體需求

- 電路元件
  - NuMaker- M2L31KI V2.1
- 線路示意圖
  - 將 UART0 TX (PB.13) pin 通過 UART 轉 USB 工具連接到 PC。
  - 將 UART0 RX (PB.12) pin 通過 UART 轉 USB 工具連接到 PC。
  - 將 LPUART RX (PA.0) pin 通過 UART 轉 USB 工具連接到 PC。

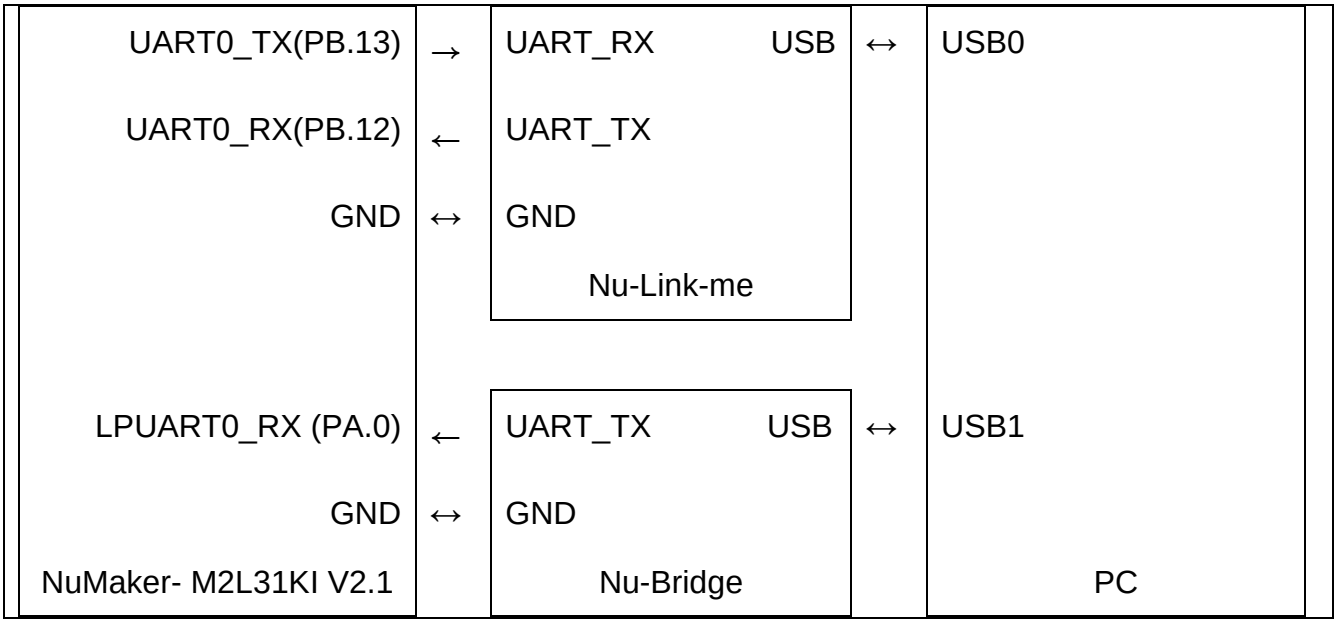

圖 3-1 線路示意圖

4. 目錄資訊

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	
Project	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *M2L31_LPUART_RX_in_PD.uvprojx*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開模擬模式
3. 進入模擬模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2025.03.07	1.01	去掉關閉LVR的代碼，防止造成上下電問題。
2024.04.19	1.00	初始發佈。

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.