使用 UART 模拟 ISO7816-3 接口

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	使用 M251 的 UART 模拟基于 ISO7816-3 标准的 Smart Card 接口。
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007
开发平台	NuMaker-M251SD V1.0

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 M251 的 UART 和 CLKO 模拟基于 ISO7816-3 标准的 Smart Card 主机接口。实现对 Smart Card 的激活、冷复位、热复位、交易、释放等操作，其中交易使用 T0 异步半双工字符传输协议。

1.1 原理

本范例代码主要用到四个外设: UART0、UART1、CLKO 以及 TIMER0。

- UART0 作为调试串口，主要用于打印调试信息和 Smart Card 的应答数据，以及输入交互信息。
- UART1 用来模拟 ISO7816-3 的 DAT。ISO7816-3 采用异步半双工通讯，收发共用一根 DAT 线，因此需要将 UART1 的 TX 和 RX 连在一起。
- CLKO 用来输出 ISO7816-3 的时钟 CLK。
- TIMER0 用来实现对 Smart Card 操作时序中的各种延时。

ISO7816-3 传输中字符帧格式如图 1-1 所示。

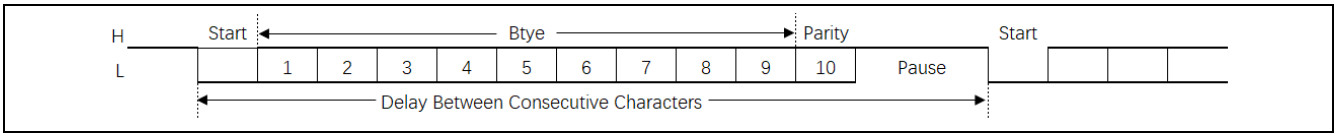


图 1-1 字符帧

基本时间单元是指传输字符帧的一位需要的时间，计算公式如下：

$$1\ etu = \frac{F}{D} \times \frac{1}{f}$$

参数 f 是 CLK 输出频率，范例中由 CLKO 输出，设置为 3 MHz。

参数 F 和 D 分别是时钟转换因子和波特率调整因子，在通讯过程中使用的 etu 依赖于 F 和 D 的实际值，在 ATR 期间默认 F = 372，D=1。

由上述可知 UART1 的波特率是 1/(1 etu)。

Timer0 的 CLK 也设置为 3 MHz。

当接收时，将 PB.6 引脚复用功能设置为 UART1_RX，PB.7 设置为 GPIO 输入，使两个引脚都处于输入模式，不会影响 Smart Card 输出数据。同时还需要使能接收数据可用和接收超时中断，接收完成后需要禁止中断。

当发送时，保持PB.6为UART1_RX功能，因为UART1_RX是输入，不会影响DAT上的数据。将 PB.7 引脚复用功能设置为 UART1_TX，并使能发送保持寄存器空中断，在中断处理函数里完成数据的发送。发送完成后需要禁止发送保持寄存器空中断，并且将 PB.7 再次设置为 GPIO 输入模式，防止 PB.7 和 Smart Card 同时输出。

本范例对Smart Card的操作步骤如下：

1) 激活 Smart Card
2) 冷复位
3) 热复位(冷复位失败)
4) 接收复位应答 ATR
5) PPS 交换
6) 交易：选择文件、获取应答
7) 释放序列

1.2 执行结果

本范例代码执行结果如图 1-2 所示。PC 上打开串口工具，连接 UART0 用于打印调试信息。

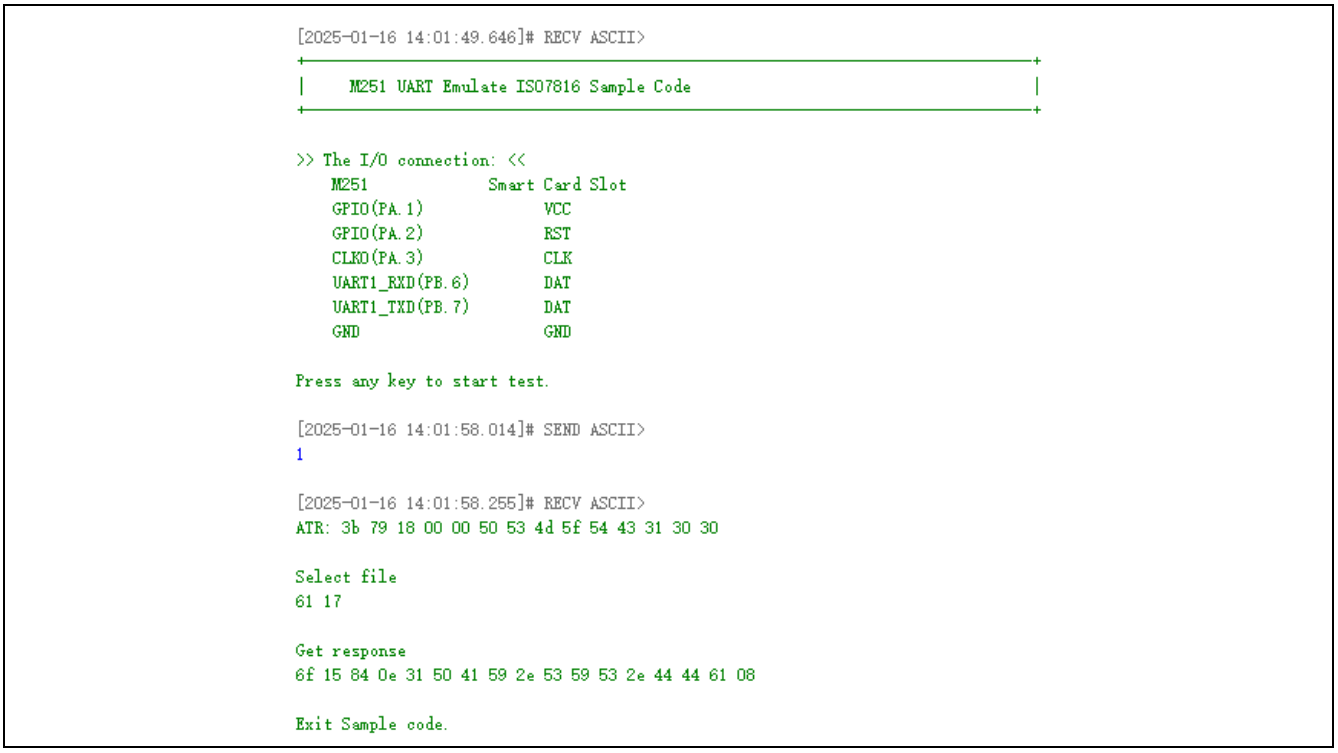


图 1-2 执行结果

2. 代码介绍

本范例程序主要分成三部分：硬件驱动、Smart Card 协议库、主函数。

硬件驱动

外设和引脚的定义，代码位于 *sc_drv_config.h*。如果要使用其他引脚、UART 或 Timer，只需要修改这里的定义，而不需要修改驱动代码。

```
/* Define function of multi-function pin */
#define SET_CLKO_PA3()       SYS->GPA_MFPL = (SYS->GPA_MFPL & ~SYS_GPA_MFPL_PA3MFP_Msk) |
                             SYS_GPA_MFPL_PA3MFP_CLKO
#define SET_UART1_RX_PB6()   SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB6MFP_Msk) |
                             SYS_GPB_MFPL_PB6MFP_UART1_RXD
#define SET_UART1_TX_PB7()   SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB7MFP_Msk) |
                             SYS_GPB_MFPL_PB7MFP_UART1_TXD
#define SET_GPIO_PB7()       SYS->GPB_MFPL = (SYS->GPB_MFPL & ~SYS_GPB_MFPL_PB7MFP_Msk)

/* VCC */
#define VCC       PA1  /* pin */
#define VCC_PORT  PA   /* port */
#define VCC_BIT   BIT1 /* bit */

/* RST */
#define RST       PA2  /* pin */
#define RST_PORT  PA   /* port */
#define RST_BIT   BIT2 /* bit */

/* CLK */
#define SC_CLK        PA3  /* pin */
#define CLK_PORT      PA   /* port */
#define CLK_BIT       BIT3 /* bit */
#define SET_CLKO_PIN  SET_CLKO_PA3 /* multi-function = CLKO */

/* UART_TX */
#define UART_TX             PB7   /* pin */
#define UART_TX_PORT        PB    /* port */
#define UART_TX_BIT         BIT7  /* bit */
#define SET_UART_TX_PIN     SET_UART1_TX_PB7 /* multi-function = UART1_TX */
#define CANCEL_UART_TX_PIN  SET_GPIO_PB7     /* multi-function = GPIO */

/* UART_RX */
#define UART_RX          PB6  /* pin */
#define UART_RX_PORT     PB   /* port */
#define UART_RX_BIT      BIT6 /* bit */
#define SET_UART_RX_PIN  SET_UART1_RX_PB6 /* multi-function = UART1_RX */

/* UART */
#define UART      UART1      /* Uart select */
#define UART_IRQ  UART1_IRQn /* IRQ */
#define UART_IRQHandler UART1_IRQHandler /* Interrupt handling function */
#define UART_CLK_SELECT()   CLK_SetModuleClock(UART1_MODULE, CLK_CLKSEL1_UART1SEL_HIRC,
```

```
 CLK_CLKDIV0_UART1(1)) /* Clock source */
#define UART_CLK_ENABLE()   CLK_EnableModuleClock(UART1_MODULE) /* Enable clock */

/* Timer */
#define TIMER   TIMER0
#define TIMER_CLK_SELECT()   CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0SEL_HIRC, 0)
#define TIMER_CLK_ENABLE()   CLK_EnableModuleClock(TMR0_MODULE) /* Enable clock */
```

硬件相关驱动函数，代码位于 *sc_drv.c*。

- Smart Card 接口初始化

```
void SC_Init(void)
{
    /* 1. Set pin multi-function */
    SC_Pin_Init();

    /* 2. UART init */
    UART_Open(UART, 8065);/* By default, F=372 D = 1, Baud Rate = f/372 = 3M/372 = 8065 */

    /* Set Rx Time-out counter */
    UART_SetTimeoutCnt(UART, 0x10);

    /* Set Rx FIFO Interrupt Trigger Level */
    UART->FIFO &= ~ UART_FIFO_RFITL_Msk;

    UART->FIFO |= UART_FIFO_RFITL_4BYTES;

    NVIC_EnableIRQ(UART_IRQn);

    /* 3. Set the clock output of the card reader using clko. f = 3M Hz */
    CLK_EnableModuleClock(CLKO_MODULE); /* Enable clk */

    /* Select CKO clock source. clock source = HCLK, No frequency division */
    CLK_SetModuleClock(CLKO_MODULE, CLK_CLKSEL1_CLKOSEL_HCLK, NULL);

    /* CKO = clock source / 2^(u32ClkDiv + 1) */
    CLK->CLKOCTL = (CLK->CLKOCTL & ~(CLK_CLKOCTL_FREQSEL_Msk | CLK_CLKOCTL_DIV1EN_Msk))
                   | (3 << CLK_CLKOCTL_FREQSEL_Pos) /* div = 2^(3+1) */
                   | (0 << CLK_CLKOCTL_DIV1EN_Pos); /* div enable   f = 48/2^3 = 3M */

    /* 4. Set timer for delay. */
    /* Set timer frequency to SC_CLK (3M Hz) */
    TIMER->CTL = TIMER_CONTINUOUS_MODE | 15;/* 3M = HCLK/(15+1) */
}
```

- DAT 输出使能函数

```
void SC_DAT_OutPut_Enable(void)
{
    /* Set multi-function pin for UART TXD */
    SET_UART_TX_PIN();

    /* Enable transmit holding register empty interrupt */
    UART_EnableInt(UART, UART_INTEN_THREIEN_Msk);
```

```
}
```

- DAT 输出禁止函数

```
void SC_DAT_OutPut_Disable(void)
{
    /* Disable TX interrupt */
    UART_DisableInt(UART, UART_INTEN_THREIEN_Msk);

    /* Wait for transmission to end */
    while (!UART_IS_TX_EMPTY(UART));

    /* Disable TX pin. Set multi-function pin for GPIO */
    CANCEL_UART_TX_PIN();
}
```

- DAT 输入使能函数

```
void SC_DAT_Input_Enable(void)
{
    /* Enable Receive Data Available Interrupt and RX Time-out Interrupt. */
    UART_EnableInt(UART, (UART_INTEN_RDAIEN_Msk | UART_INTEN_RXTOIEN_Msk));
}
```

- DAT 输入禁止函数

```
void SC_DAT_Input_Disable(void)
{
    UART_DisableInt(UART, (UART_INTEN_RDAIEN_Msk | UART_INTEN_RXTOIEN_Msk));
}
```

- 延时函数

```
void SC_Timer_Delay(uint32_t ticks)
{
    uint32_t t1;

    t1 = TIMER->CNT;

    while(TIMER->CNT - t1 < ticks);
}
```

Smart Card 协议库

Smart Card 协议库的代码全部位于 SmartcardT0Lib 文件夹下。包含用于应用程序调用的 API 接口函数、协议的实现代码和协议内部调用的函数。

Smart Card 协议库的 API 接口函数，代码位于 *SmartcardT0Lib\Source\sclib_api.c* 文件下。

- 卡激活

```
int32_t SCLIB_Activate(uint32_t num, uint32_t u32EMVCheck)
{
    SCLIB_DEV_T *dev = &_scDev[num];
```

```
    int32_t retval, clk;

    /* 1. Read clock of card slot. */
    clk = _SCLIB_GetInterfaceClock(num);

    /* Upper and lower clock limits */
    if(clk < 1000 || clk > 5000)
        return SCLIB_ERR_CLOCK;

    /* 2. Set card status=card present */
    dev->bCardRemoved = SCLIB_CARD_PRESENT;

    /* 3. Set whether to check EMV */
    dev->EMV = u32EMVCheck;

    /* 4. Cold Reset */
    retval = SCLIB_ColdReset(num);

    /* 5. Warm Reset */
    if(retval == SCLIB_ERR_ATR_INVALID_PARAM) /* Cold Reset failed. */
        retval = SCLIB_WarmReset(num); /* Warm Reset */

    /* 6. Set Tx and Rx buffers */
    dev->snd_buf = &g_TxBUF[0]; /* Tx buffer */

    dev->rcv_buf = &g_RxBUF[0]; /* Rx buffer */

    return(retval);
}
```

- 冷复位

```
int32_t SCLIB_ColdReset(uint32_t num)
{
    SCLIB_DEV_T *dev = &_scDev[num]; /* Device pointer */

    /* 1. Cold Reset */
    dev->errno = _SCLIB_ResetCard(num, SCLIB_COLDRESET);/* Smartcard interface number,
    Cold Reset*/

    /* Cold Reset failed. */
    if(dev->errno != SCLIB_SUCCESS)
        return (dev->errno);

    /* 2. Update card Capabilities, read from ATR. */
    dev->pps_complete = 0;

    dev->errno = _SCLIB_UpdateCardCapabilities(num);

    /* Update failed */
    if(dev->errno != SCLIB_SUCCESS)
        return (dev->errno);

    /* 3. Parsing ATR, and PPS exchange */
    dev->errno = _SCLIB_ParseATR(num, SCLIB_COLDRESET);/* Smartcard interface number,
    Cold Reset */
```

```
    if(dev->errno != SCLIB_SUCCESS) /* Failed */
        SCLIB_ERR("Cold Reset Parse ATR..Error :%d \n", dev->errno);

    if(dev->errno == SCLIB_SUCCESS) /* Success */
        /* Smartcard has been activated. */
        dev->openflag = 1;

    return (dev->errno);
}
```

- 启动传输

```
int32_t SCLIB_StartTransmission(uint32_t num, uint8_t *cmdBuf, uint32_t cmdLen, uint8_t
*rspBuf, uint32_t *rspLen)
{
    int32_t retval = SCLIB_SUCCESS;
    SCLIB_DEV_T *dev = &_scDev[num];

   /* 1. Not activate yet.*/
    if(dev->openflag == 0)      /* Try to access smartcard before activation. */
        return SCLIB_ERR_DEACTIVE;

    /* 2. Smartcard busy. */
    if(dev->op_state)           /* Smartcard busy, previous transmission not complete. */
        return SCLIB_ERR_CARDBUSY;

    /* Give up data remain in dev->buf, if anything remains... */
    dev->snd_pos = 0; /* Send buffer offset = 0 */

    dev->rcv_pos = 0; /* Receive buffer offset = 0 */

    /* 3. Send a command and get response from smartcard according to protocol */
    if(dev->CardCapabilities.Protocol.Selected == SCLIB_PROTOCOL_T0) /* T0 */
    {
        /* Using T=0 protocol */
        retval = _SCLIB_T0Transmit(num,     /* Smartcard interface number */
                                   cmdBuf,  /* cmd buffer */
                                   cmdLen,  /* cmd len */
                                   rspBuf,  /* response buffer */
                                   rspLen); /* response len */
    }
    else if(dev->CardCapabilities.Protocol.Selected == SCLIB_PROTOCOL_T1) /* T1 */
    {
        retval = _SCLIB_T1Transmit(num, cmdBuf, cmdLen, rspBuf, rspLen);
    }

    /* 4. Stop all timers here, the transmission is complete no matter success or not. */
    SC_Timer_Stop();

    return retval;
}
```

- 卡传输事件处理

此函数需要在 UART 的中断处理函数里被调用。

```
uint32_t SCLIB_CheckTxRxEvent(uint32_t num)
{
   ……
    /* 1. Transmit buffer empty interrupt */
    if (SC_GetTxIntFlag())
    {
        retval = 1;

        if( dev->op_state == SCLIB_OP_WRITE )/* Operation = Write */
        {
            ……
        }
    }
    /* 2. Receive Data Available Interrupt or Rx TimeOut interrupt. */
    if(SC_GetRxIntFlag())
    {
        ……
        if(dev->op_state == SCLIB_OP_READ) /* Operation = Read */
        {
            ……
        }
        else if(dev->op_state == SCLIB_OP_ATR_READ)  /* Operation = Read ATR */
        {
            ……
        }
         else {
            ……
        }
    }
out:
    return retval;
}
```

另外 T0 异步半双工字符传输协议的实现代码位于 *SmartcardT0Lib\Source\sclib_t0protocol.c*。Smart Card 协议库的内部相关函数代码位于 *SmartcardT0Lib\Source\sclib_util.c*。

主函数

```
int32_t main(void)
{
    SCLIB_CARD_INFO_T s_info;
    int retval, i;
    uint32_t u32Ret;

    /* Unlock protected registers */
    SYS_UnlockReg();

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

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

```
#if defined (__GNUC__) && !defined(__ARMCC_VERSION) && defined(OS_USE_SEMIHOSTING)
    initialise_monitor_handles();
#endif

    /* Init UART0 for printf and test */
    UART0_Init();

    printf("+-------------------------------------------------------------------------+\n");
    printf("|     M251 UART Emulate ISO7816 Sample Code                               |\n");
    printf("+-------------------------------------------------------------------------+\n\n");
    printf(">> The I/O connection: <<\n");
    printf("    M251             Smart Card Slot\n");
    printf("    GPIO(PA.1)             VCC\n");
    printf("    GPIO(PA.2)             RST\n");
    printf("    CLKO(PA.3)             CLK\n");
    printf("    UART1_RXD(PB.6)        DAT\n");
    printf("    UART1_TXD(PB.7)        DAT\n");
    printf("    GND                    GND\n");
    printf("\nPress any key to start test.\n");

    getchar();

    /* Init ISO7816 */
    SC_Init();

    /* Activate Smartcard interface 0 */
    retval = SCLIB_ActivateDelay(0,      /* Smartcard interface number */
                                 FALSE,  /* Disable EMV error check */
                                 10000); /* Delay clk */

    if (retval == SCLIB_SUCCESS) /* Activated */
    {
        /* Copy ATR information to variable. */
        SCLIB_GetCardInfo(0, &s_info);  /* Smartcard interface number, variable */

        /* Print ATR information */
        printf("ATR: ");

        for (i = 0; i < s_info.ATR_Len; i++)
            printf("%02x ", s_info.ATR_Buf[i]);

        printf("\n\n");

        /* Smartcard transmissiontransmission test: Select file MF */
        u32Ret = SCLIB_StartTransmission(0,              /* Smartcard interface number */
                                         g_cmd11buf,     /* Cmd buffer */
                                         g_cmd1len,      /* Cmd len */
                                         g_cmdrsp,       /* Response buffer */
                                         &g_cmd1rsplen); /* Response len */

        if (u32Ret != 0) /* Error */
        {
            dwErrorNum++;
        }
        else
```

```
        {
            /* Print cmd response data */
            printf("Select file\n");

            for (i = 0; i < g_cmd1rsplen; i++)
                printf("%02x ", g_cmdrsp[i]);

            printf("\n\n");

            /* Smartcard transmission test: get response */
            u32Ret = SCLIB_StartTransmission(0,           /* Smartcard interface number */
                                             g_cmd3buf,      /* Cmd buffer */
                                             g_cmd3len,      /* Cmd len */
                                             g_cmdrsp,       /* Response buffer */
                                             &g_cmd1rsplen); /* Response len */

            if (u32Ret != 0)
            {
                dwErrorNum++;
            }
            else
            {
                /* Print cmd response data */
                printf("Get response\n");

                for (i = 0; i < g_cmd1rsplen; i++)
                    printf("%02x ", g_cmdrsp[i]);

                printf("\n");
            }
        }
    }
    else
    {
        printf("Smartcard activate failed\n");
    }

    printf("\nExit Sample code.\n");

    while (1);

}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007
- IDE 版本
  - Keil uVision 5.40.0.0

3.2 硬件需求

- 电路组件
  - NuMaker-M251SD V1.0
- 线路示意图
  - 将 PA.1、PA.2、PA.3 分别连接到 Smart Card 槽的 VCC、RST、CLK。
  - 将 UART1 TX (PB.7) 脚和 UART1_RX(PB.6) 都连接到 Smart Card 槽的 DAT。
  - 将 UART0 TX (PB.13) 脚和 UART0 RX (PB.12) 脚通过 Nu-Link-me 转接到 PC。

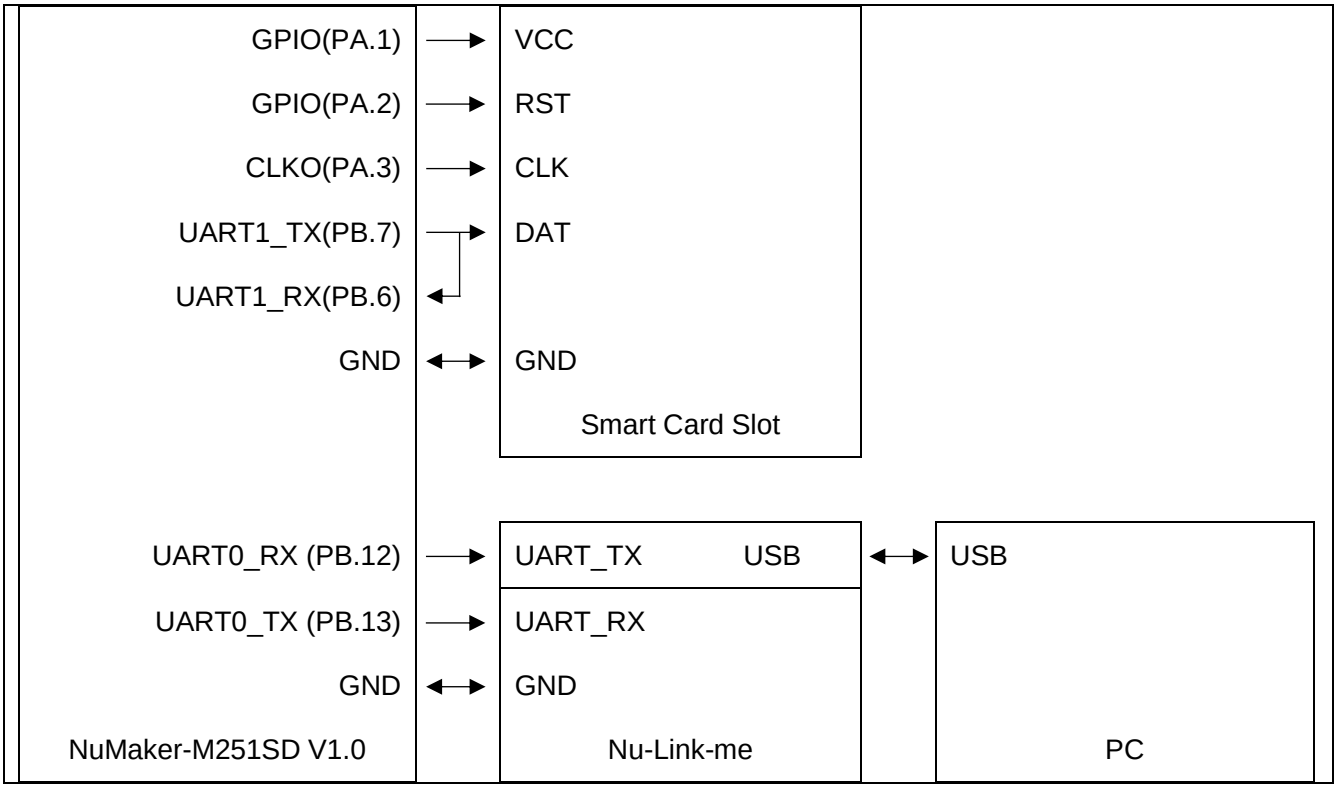

GPIO(PA.1)
GPIO(PA.2)
CLKO(PA.3)
UART1_TX(PB.7)
UART1_RX(PB.6)
GND
VCC
RST
CLK
DAT
GND
Smart Card Slot
UART0_RX (PB.12)
UART0_TX (PB.13)
GND
UART_TX
USB
UART_RX
GND
USB
NuMaker-M251SD V1.0
Nu-Link-me
PC

图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

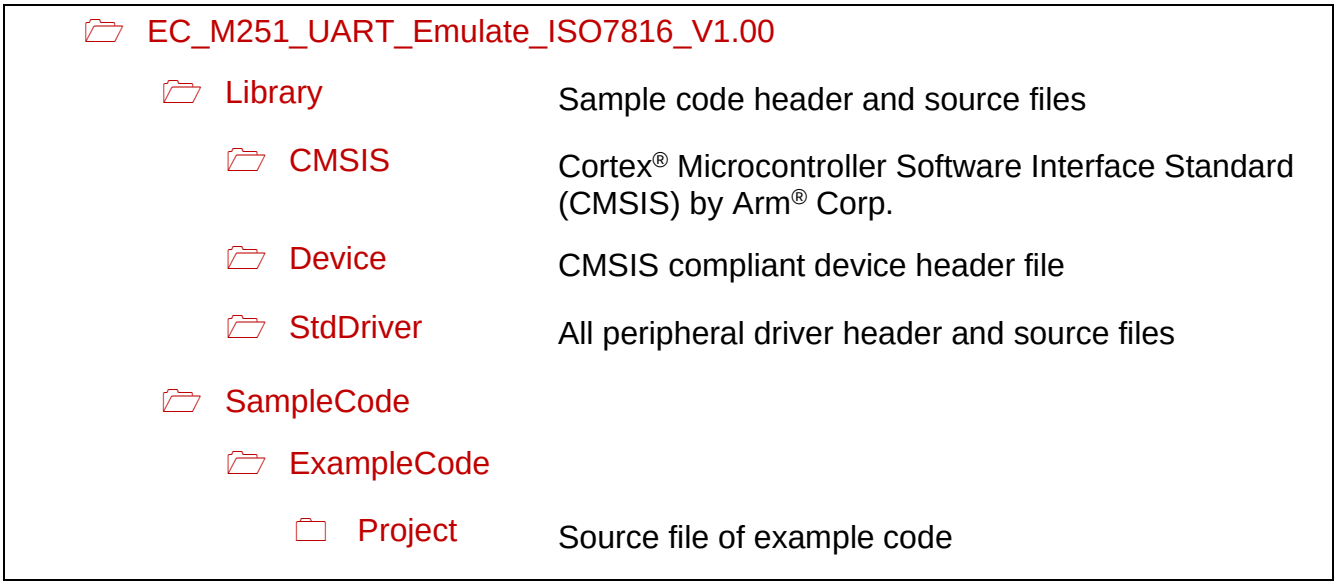


图 4-1 目录信息

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 *M251_UART_Emulate_ISO7816.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
2025.02.11	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.