CM2003 ModbusRTU 採集熱電偶感測器

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼介紹如何使用 CM2003 的 USCI 介面連接 NADC24 採集熱電偶感測器的溫度訊號。
BSP 版本	CM2003_Series_BSP_CMSIS_V3.00.000
開發平台	CM2003_ModbusRTU_NADC24_V0.1

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

本範例代碼介紹了 CM2003 通過 USCI 介面讀取 NADC24 的差分通道，用以採集外接的熱電偶感測器的溫度訊號。被讀取的溫度信號使用 RS485 介面利用 Modbus 協議傳遞至 USB to RS485 adaptor，USB to RS485 adaptor 再傳至 PC 以便接收分析使用。

Modbus 協議使用了開源 FreeModbus 協議棧。FreeModbus 因其開源、輕量和易移植的特點，在嵌入式 Modbus 設備開發中被廣泛採用。開發者可以根據專案需求進行定制和優化。

更多關於 FreeModbus 的介紹請查看其官網介紹：

https://www.embedded-experts.at/en/freemodbus/about/

1.1 CM2003 介紹

本範例需要 CM2003 的 3 種基本功能，UART、SPI、TIMER：

- UART：介面通過 RS485 負責實現 Modbus 協議的資料的收發。
- SPI：介面負責讀寫 NADC24 的資料，因 CM2003 沒有標準 SPI 介面，使用了 USCI 介面的 SPI 功能來實現。NADC24 有 8-bit 和 24-bit 資料模式，但 USCI 只能支援 8-bit 和 16-bit 兩種模式，因此範例代碼 USCI 統一按照 8-bit 模式來進行資料的讀寫。
- TIMER：TIMER0 用於判斷通訊超時，Modbus 協議要求每一包資料間隔要有 ≥ 3.5 個位元訊號的時間。TIMER1 用於產生秒計數。

1.2 Modbus 協議簡介

Modbus 使用主從架構，且有三種協議類型：

- Modbus RTU（基於串列通信，如 RS-485/RS-232，二進位元編碼，高效緊湊）。
- Modbus ASCII（基於串列通信，ASCII 字元編碼，可讀性強但效率低）。
- Modbus TCP/IP（基於乙太網，使用 TCP，適用於現代工業網路）。

協議的主要內容為：設備地址 + 指令 + 暫存器地址 + 資料 + CRC 校驗

其中 8 種指令，見表 1-1：

功能碼（十進位）	操作	資料類型
1	讀取線圈狀態	1-bit（開關量）
2	讀取離散輸入	1-bit（唯讀）
3	讀取保持暫存器	16-bit（可讀寫）
4	讀取輸入暫存器	16-bit（唯讀）
5	寫入單個線圈	1-bit（開關量）
6	寫入單個保持暫存器	16-bit（可讀寫）
15	寫入多個線圈	1-bit（批量操作）
16	寫入多個保持暫存器	16-bit（批量操作）

表 1-1 Modbus 指令表

本範例中使用了 Modbus 協議中 4 種功能碼：3、4、6、16。

範例中定義的 Modbus 通訊格式為，見表 1-2：

通訊參數	參數值
Address	1
Baud_rate	115200
data_bits	8
Stop_bits	1
Parity	無
Flow_control	無

表 1-2 範例通訊格式

暫存器定義，見表 1-3：

暫存器位址（十進位）	暫存器類型	定義	資料格式

暫存器位址（十進位）	暫存器類型	定義	資料格式
1000	輸入暫存器（唯讀）	硬體版本	16-bit HEX
1001		軟體版本	16-bit HEX
1002		秒計數	1 秒
1003		輸入口狀態	0/1
1004		熱電偶溫度	0.1 ℃
1005		NADC24 溫度	0.1 ℃
2000	保持暫存器（可讀寫）	LED 狀態	0/1
2001		IO 輸出狀態	0/1
2002		NADC24 通道設置	同 NADC 通道暫存器
2003		MCU 工作模式	未啟用

表 1-3 暫存器資料

更多 Modbus 的介紹請查看 Modbus 組織的協議文檔：

https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

1.3 熱電偶溫度採集原理

當兩種不同的金屬導體（或半導體）兩端連接形成閉合回路時，若兩個連接點（稱為熱端和冷端）存在溫度差（ΔT），回路中會產生電動勢（電壓），這種現象稱為塞貝克效應（Seebeck Effect）。產生的電壓稱為熱電勢，其大小與溫度差成正比，關係式為：

$$V = \alpha \cdot \Delta T$$

其中：

- V：熱電勢（單位：mV）。
- α：塞貝克係數（材料特性，單位：μV/°C）。
- ΔT：熱端與冷端的溫度差（單位：°C）。

熱電偶測量的基本結構：

- 熱端（測量端）：暴露在待測溫度環境中，溫度 T_1。
- 冷端（參考端）：保持已知恒定溫度（如 0 °C）或通過補償電路修正，溫度 T_2。

- 導線材料：兩種不同的金屬導體。

熱電偶通過測量熱電勢 V 來推算熱端溫度 T_1，但需已知冷端溫度 T_2。

若熱電偶冷端溫度不為 0 ℃，而是 T_2，則熱電勢 $V(T_1, T_2)$ 與標準分度表 (冷端為 0 ℃) 的關係為：

$$V(T_1, 0) = V(T_1, T_2) + V(T_2, 0)$$

1.3.1 熱電偶分類

熱電偶根據其導體材料的不同，有多種規格可供選擇

- K 型熱電偶：由鎳鉻-鎳矽材料構成

  適用於 -200 ℃至 1300 ℃ 的溫度範圍，廣泛應用於工業領域。

- J 型熱電偶：由鐵-康銅材料構成

  適用於 -200 ℃至 750 ℃ 的溫度範圍，適合在氧化性氣體環境中使用。

- E 型熱電偶：由鎳鉻-銅鎳材料構成

  適用於 -200 ℃至 900 ℃ 的溫度範圍，具有較高的熱電動勢。

- T 型熱電偶：由銅-康銅材料構成

  適用於 -200 ℃至 350 ℃ 的溫度範圍，適合低溫測量。

- N 型熱電偶：由鎳鉻矽-鎳矽鎂材料構成

  適用於 -200 ℃至 1300 ℃ 的溫度範圍，具有良好的抗氧化性。

範例使用了通用性較廣的 K 型熱電偶，並且符合 ITS-90 國際溫標。關於 ITS-90 的更多資訊請查看官方定義：

https://www.nist.gov/system/files/documents/pml/div685/grp01/ITS-90_metrologia.pdf

K 型溫度和電勢的對應關係：當 0 ℃ 對應 0 mV 時

-270 ℃	<===	0 ℃	===>	1372 ℃
-6.458 mV	<===	0 mV	===>	54.886 mV

詳細的溫度電壓對應關係可查看連結檔：

https://www.omega.co.uk/temperature/pdf/Type_K_Thermocouple_Reference_Table.pdf

1.3.2 熱電偶溫度計算方法

熱電偶的溫度計算方法有 2 種，查表法和計算法。兩種計算方法均被 ITS-90 認可。

- 查表法：通常用在處理器主頻不高的情況下，範例代碼選用了查表法進行溫度計算。
- 計算法：通過多項式的方程可計算出溫度。

1.3.3 測熱電偶温度的常规步驟：

1. 讀取 NADC24 內部溫度校準值；
2. 根據校準值計算 NADC24 內部溫度感測器溫度（冷端溫度）；
3. 冷端溫度轉換成冷端電壓；
4. 測量熱電偶熱端電壓；
5. 通過熱端電壓和冷端電壓計算絕對溫度。

1.4 NADC24 介紹

熱電偶的電勢變化很小，常規的 ADC 無法進行精準的採集，需要解析度高，差分通道才能精確的採集熱電偶的電池變化。範例中使用的 NADC24 有 3 組差分通道，並且內部有溫度感測器可作為冷端的溫度採集；CH6 和 CH7 組成一組差分通道用來採集熱端溫度。

NADC24 每取樣一次產生一個中斷訊號，可以設置為連續取樣或者單次取樣。範例中使用增益為 32 倍，取樣速率為 32 kbps 進行連續取樣。

2. 代碼介紹

代碼主要分成 4 部分：

- 程式主框架：負責初始化，資料處理。
- FreeModbus 協議棧：負責 Modbus 協議通訊參數設置，資料收發。
- NADC24 電壓測量：包含 IC 的初始化，校準程式，資料獲取。
- 熱電偶溫度計算：熱電偶溫度的計算。

2.1 程式主框架

主程序開始對系統進行初始化，包括 CM2003 內部介面的初始化，Modbus 協議的初始化在 *SYS_Init()* 函數內，NADC24 的初始化，Modbus 協議暫存器的初始化。主循环內 *eMBPoll()* 負責 Modbus 協議的解析，*NADC24_Loop()* 在檢測轉換結束時由中斷旗標位元觸發執行。

```
/*---------------------------------------------------------------*/
/* MAIN function                                                 */
/*---------------------------------------------------------------*/
int main(void)
{
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    LED_PIN = 0;
    NADC24_Init();// Input: N_TP, P_TP ,VREF=2.4V

     RegInput[HW_VER_REG] =    HW_VERSION;
     RegInput[SW_VER_REG] =    SW_VERSION;

    /* Set PB multi-function pin for EINT1 (PB.4) */
    SYS->GPB_MFPL = (SYS->GPB_MFPL & ~( SYS_GPB_MFPL_PB4MFP_Msk ) |
    ( SYS_GPB_MFPL_PB4MFP_INT1 ));
    GPIO_EnableInt(PB, 4, GPIO_INT_FALLING);//NADC24 sampling completion interrupt.
    NVIC_EnableIRQ(EINT1_IRQn);

    TIME3_Init();

    while (1)
    {
        /*eMBPoll for modbus event process*/
        (void)eMBPoll();
        NADC24_Loop(); //NADC24 main loop.

        if(PB13 == 1)
            RegInput[MCU_DIN_REG] = 1;
        else
```

```
            RegInput[MCU_DIN_REG] = 0;
    };
}
```

2.2 FreeModbus 協議棧

FreeModbus 協議棧移植完成後只需要關注 *user_modbus.c* 檔中 4 種暫存器類型資料的處理，範例中僅使用了輸入暫存器和保持暫存器 2 種暫存器。輸入暫存器只向外部發送資料，保持暫存器即可以寫入也能發送。如果使用者需要非易失，要在保持暫存器中加入 Flash 的寫入功能。

```
/**********************************************************************************
 *  @brief       Read input Registers
 *  @param[in]   pucRegBuffer: Data pointer.
 *                  usAddress: Register address.
 *                    usNRegs: Number of register operations.
 *  @return     Modbus operational Status.
 *  @details    Only send out data int input Registers.
**********************************************************************************/
eMBErrorCode eMBRegInputCB(UCHAR *pucRegBuffer, USHORT usAddress, USHORT usNRegs)
{
    eMBErrorCode    eStatus = MB_ENOERR;
    int             iRegIndex;

    usAddress--;

    if ((usAddress >= REG_INPUT_START)
            && (usAddress + usNRegs <= (REG_INPUT_START + REG_INPUT_NREGS)))
    {
        iRegIndex = (int)(usAddress - usRegInputStart );

        while (usNRegs > 0)
        {
            *pucRegBuffer++ = (unsigned char)(RegInput[iRegIndex] >> 8);
            *pucRegBuffer++ = (unsigned char)(RegInput[iRegIndex] & 0xFF);
            iRegIndex++;
            usNRegs--;
        }
    }
    else
    {
        eStatus = MB_ENOREG;
    }
    return eStatus;
}

/**********************************************************************************
 *  @brief       Read & write Holding Registers
 *  @param[in]   pucRegBuffer: Data pointer.
 *                  usAddress: Register address.
 *                    usNRegs: Number of register operations.
 *                      eMode: MB_REG_READ or MB_REG_WRITE
 *  @return     Modbus operational Status.
 *  @details    The register operation can be placed in Hold_Reg_Refresh().
```

```
**********************************************************************************/
eMBErrorCode eMBRegHoldingCB(UCHAR *pucRegBuffer, USHORT usAddress, USHORT usNRegs,
eMBRegisterMode eMode)
{

    eMBErrorCode    eStatus = MB_ENOERR;
    int             iRegIndex;

    usAddress--;

    if ((usAddress >= REG_HOLD_START)
            && (usAddress + usNRegs <= (REG_HOLD_START + REG_HOLD_NREGS)))
    {
          iRegIndex = (int)(usAddress - usRegHoldStart);

         if(eMode == MB_REG_READ)
         {
              while (usNRegs > 0)
              {
              *pucRegBuffer++ = (unsigned char)(RegHold[iRegIndex] >> 8);
              *pucRegBuffer++ = (unsigned char)(RegHold[iRegIndex] & 0xFF);
              iRegIndex++;
              usNRegs--;
              }
         }
         else
         {
              while (usNRegs > 0)
              {
              RegHold[iRegIndex]  = (*pucRegBuffer++)<<8;
              RegHold[iRegIndex]  |= *pucRegBuffer++ ;
              iRegIndex++;
              usNRegs--;
              }
         }
    }
    else
    {
        eStatus = MB_ENOREG;
    }
    return eStatus;
}
```

2.3 NADC24 電壓測量

NADC24 在初始化的時候已經做好了測量電路的校準。內部溫度感測器在系統啟動的時候需進行第一次取樣（冷端取樣），取樣完成後將通道再切換到熱電偶的取樣通道持續取樣（熱端取樣）。2 種取樣均通過 16 次平均取樣，32 kbps 的取樣速率 16 次取樣結束後將溫度計算後放入 Modbus 相應的輸入暫存器內。需要注意的是進行 16 次平均後，實際溫度的更新速率大約為 0.5 秒。

```
/**********************************************************************************
```

```
 *  @brief      NADC24 main loop.
 *  @param[in]  NULL
 *  @return     NULL
 *  @details    Put it in the main loop of the program.
It needs to run to completion before the next sampling completion flag is set.
*****************************************************************************/
void NADC24_Loop(void)
{
    static float32_t temp;
    if(nadc24IntFlag)
    {
    /* Read Current value */
    u32AdcData = SPI_ReadADCData();

    if (u32AdcData & 0x800000)
         u32AdcData += 0xFF000000;//Convert to a signed number.

    s32AdcData = (int32_t)u32AdcData;

    switch(nadc24_mode)
    {
    case NADC24_GET_CHTP://Internal temperature sensor.

    NADC24_RawData_Database[Average_Count] = s32AdcData;

    Average_Count++;
    /* Average value */
    if (Average_Count >= 16)
    {
         if(RegInput[FREE_RUN_REG] > 1)
         SPI_Send_ADC_Commnad(ADC_STOP_CONVERSION_CMD);

         arm_mean_q31(&NADC24_RawData_Database[0], 16, &NADC24_RawData);
         Calc_NADC24TP(NADC24_RawData);

         /* Finish */
         Average_Count = 0;
         if(RegInput[FREE_RUN_REG] > 1)
         {
         NADC24_Mode_Set(NADC24_GET_CH67);//Channel switching.
         }
    }
    break;

    case NADC24_GET_CH67://Thermocouple temperature sensor.

    NADC24_RawData_Database[Average_Count] = s32AdcData;

    Average_Count++;
    /* Average value */
    if (Average_Count >= 16)
    {
         Average_Count = 0;
         arm_mean_q31(&NADC24_RawData_Database[0], 16, &NADC24_RawData);

         temp = (float32_t)NADC24_RawData;
```

```
        //PGA = 32; Cold junction compensation.
        temp = ((1200*temp)/0x7fffff)/32 + TempColdVol;

        /* Finish */
        RegInput[TCK_REG]=(int16_t)(TC_Table_Convert(TS_Table, TCK_Temp_Min, temp)*10);


        printf("TC = %d,%d\n\n", NADC24_RawData,RegInput[TCK_REG]);
    }
    default :break;
    }
    nadc24IntFlag = 0;
    RegHold[NADC24_CH_REG] = nadc24_mode ;
    }
}
```

2.4 熱電偶溫度計算

熱電偶的溫度計算在前面已有介紹，這裡展示一下根據熱電偶的電勢計算溫度。先根據查表法得出當前溫度整數位元的電勢區段，然後再通過比例計算溫度的小數位，將整數位元和小數位相加則可以得出浮點數的溫度資料。下面程式僅僅是根據電勢計算溫度的功能，計算絕對溫度的時候需要進行冷端溫度的補償，補償的計算過程在 *NADC24_Loop()* 函數中。

```
/*****************************************************************************
 *  @brief        TC Table Convert
 *  @param[in]      tc_table: Table pointer.
 *                  temp_min: The lowest measurable temperature.
 *                voltage_mv: Measure the voltage value of the thermocouple.
 *  @return     Temperature value.
 *  @details    Calculate the temperature value based on the voltage lookup table.
*****************************************************************************/
float TC_Table_Convert( const float * tc_table, int16_t temp_min, float voltage_mv )
{
  uint16_t index_temp;
  float   tc_temp   ;

    //Searches for data locations in a table through indexes.
  index_temp = Get_TableIndex(tc_table, TCK_Table_Total,voltage_mv );

  //temp_min + index_temp + ( voltage_mv - table_vlotage ) / (The voltage difference
  between two indexes)
  tc_temp    = temp_min + index_temp + ( voltage_mv - * ( tc_table + index_temp ) ) / ( *
  ( tc_table + index_temp + 1 ) - * ( tc_table + index_temp ) );

  return (float)tc_temp;
}
```

2.5 連接測試

範例測試使用 Mthing 讀取 Modbus 協議的資料。Mthing 是一款功能強大的 Modbus 協議分析軟體，個人版可免費使用。更多資訊可訪問其官網：

http://www.gulink.cn/

1. 按照箭頭順序添加通訊埠，參考圖 2-1：

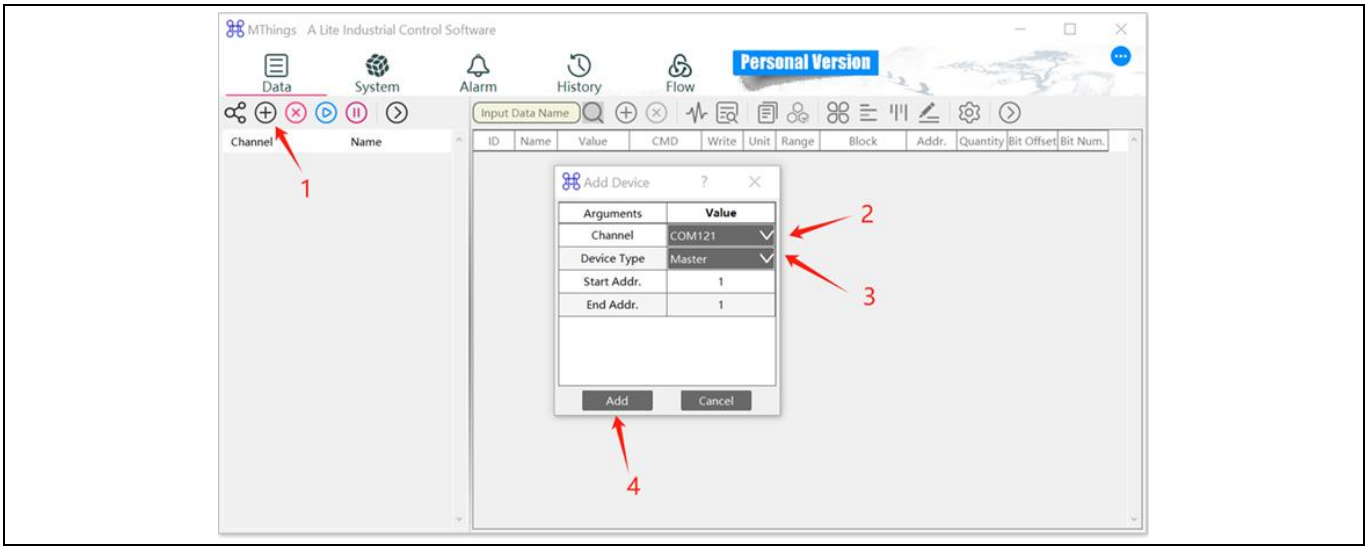

圖 2-1 添加通訊埠

2. 按照箭頭順序設置通訊速率，參考圖 2-2：

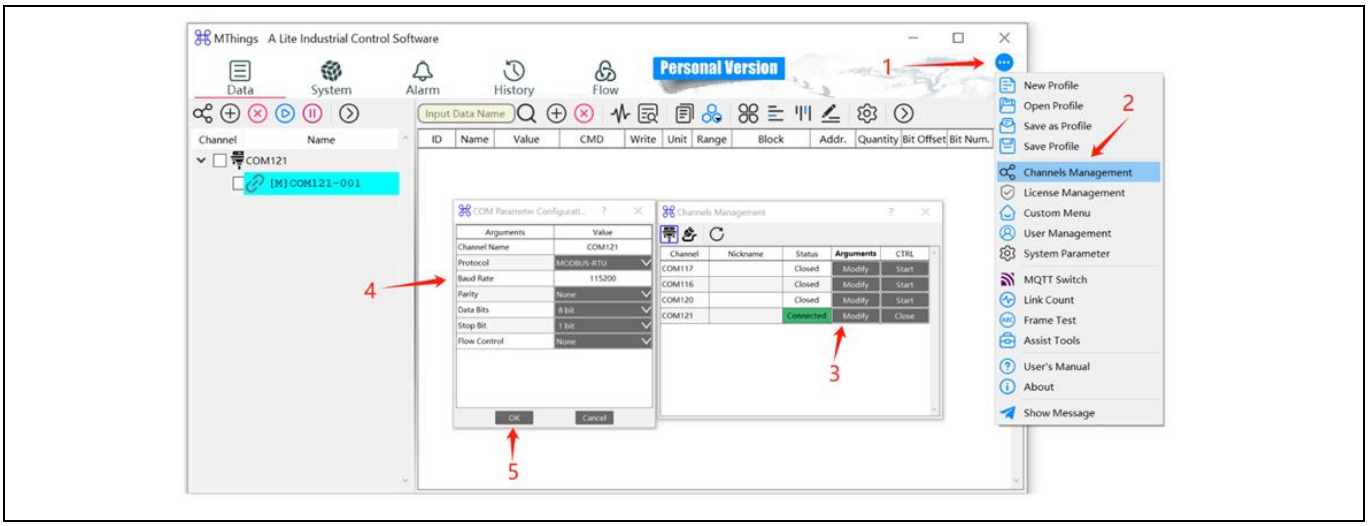

圖 2-2 設置通訊速率

3. 添加讀暫存器的參數，參考圖 2-3：

   1) 添加讀數據；

   2) 一次讀 6 個暫存器；

   3) 暫存器類型為輸入暫存器；

   4) 讀暫存器起始位址為 1000；

   5) 添加結束。

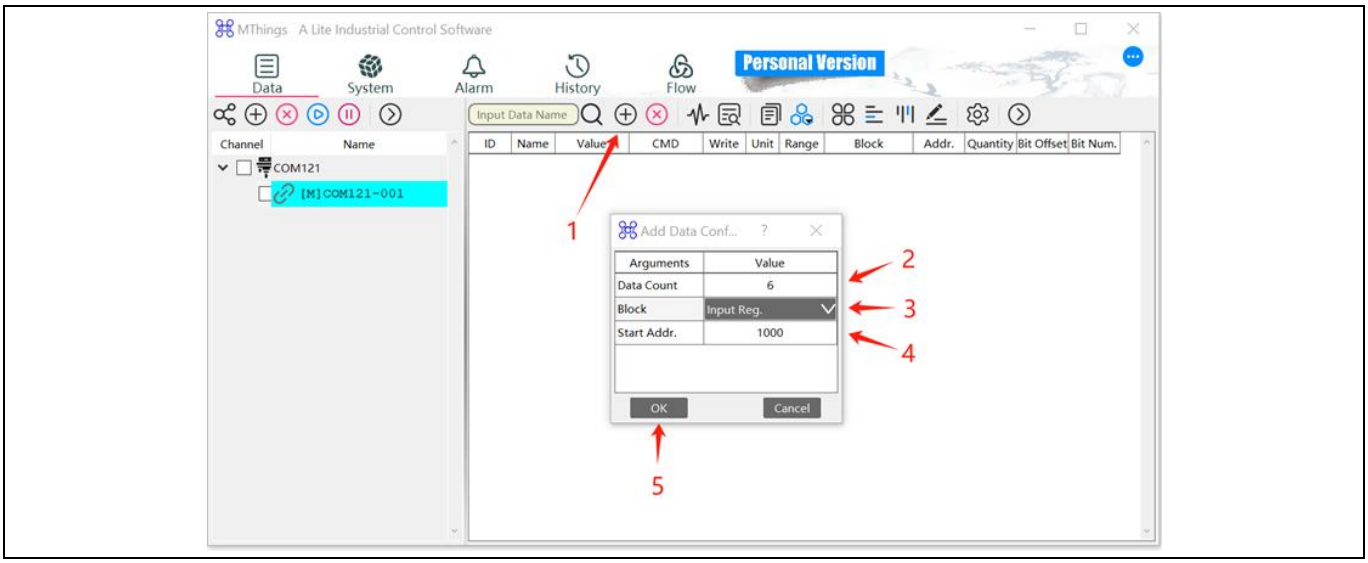

圖 2-3 添加讀暫存器

4. 開始資料輪詢，參考圖 2-4：

   - 當前熱電偶測試溫度十進制數據為 259，實際溫度為 25.9 ℃。
   - NADC24 內部的溫度十進制數據為 263，實際溫度為 26.3 ℃。

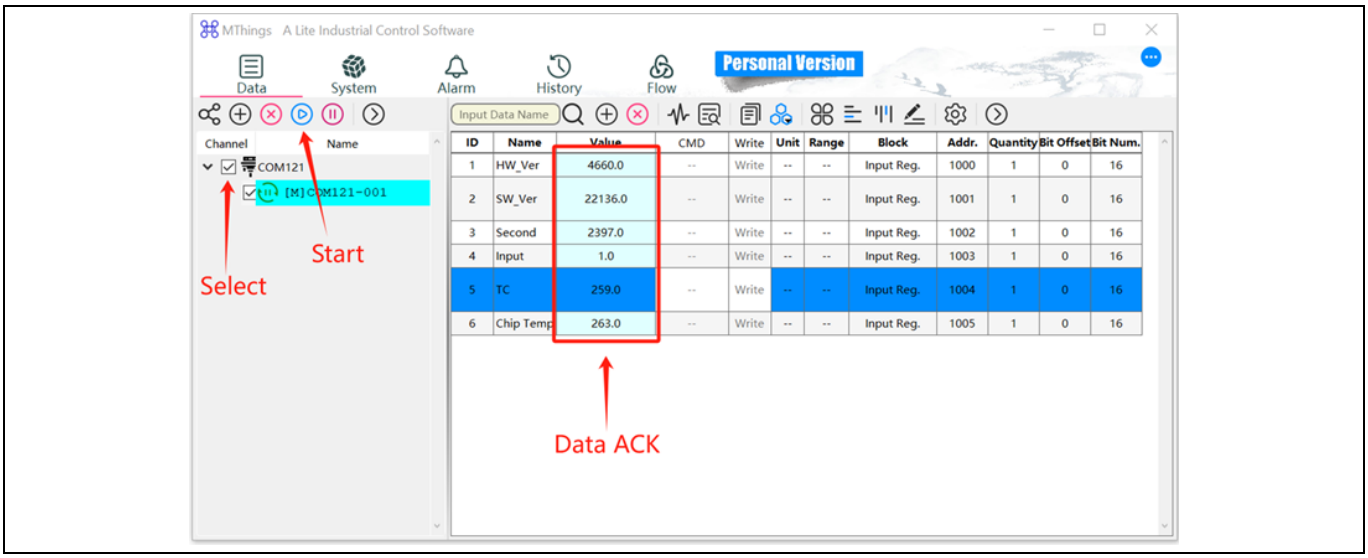
MThings A Lite Industrial Control Software
Data
System
Alarm
History
Flow
Personal Version
Input Data Name
Channel
Name
COM121
[M]COM121-001
Start
Select
Data ACK
ID Name Value CMD Write Unit Range Block Addr. Quantity Bit Offset Bit Num.
1 HW_Ver 4660.0 -- Write -- -- Input Reg. 1000 1 0 16
2 SW_Ver 22136.0 -- Write -- -- Input Reg. 1001 1 0 16
3 Second 2397.0 -- Write -- -- Input Reg. 1002 1 0 16
4 Input 1.0 -- Write -- -- Input Reg. 1003 1 0 16
5 TC 259.0 -- Write -- -- Input Reg. 1004 1 0 16
6 Chip Temp 263.0 -- Write -- -- Input Reg. 1005 1 0 16

圖 2-4 開始讀數據

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - CM2003_Series_BSP_CMSIS_V3.00.000
- IDE 版本
  - Keil uVision 5.37@ Arm Compiler 6
- PC APP 版本
  - MThings　V0.8.0.6

3.2 硬體需求

- 電路元件
  - CM2003_ModbusRTU_NADC24_V0.1
  - USB 轉 RS485 適配器
- 線路示意圖
  - USB 轉 RS485 設備接入電腦，以實現上位機的軟體的資料讀寫。
  - 將 ModbusRTU 從機 RS485 連接到 USB 轉出的 RS485，注意 RS485 訊號線的 A/B 極性。

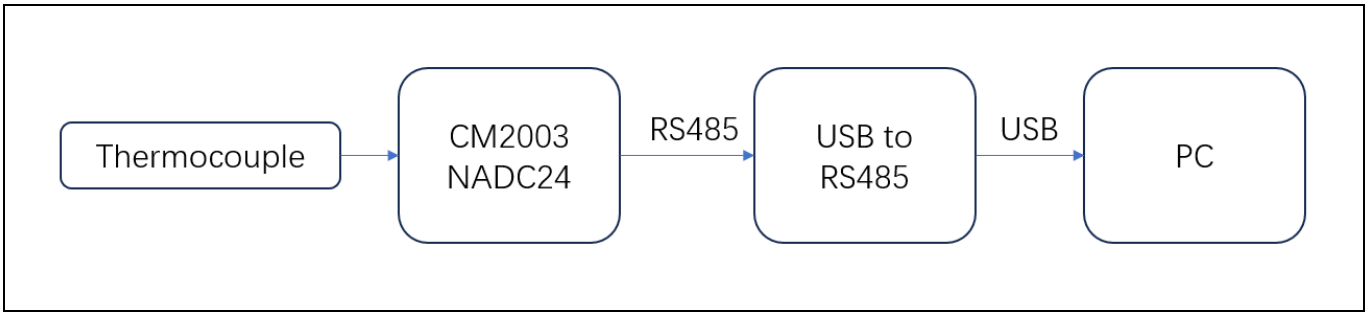

圖 3-1 線路示意圖

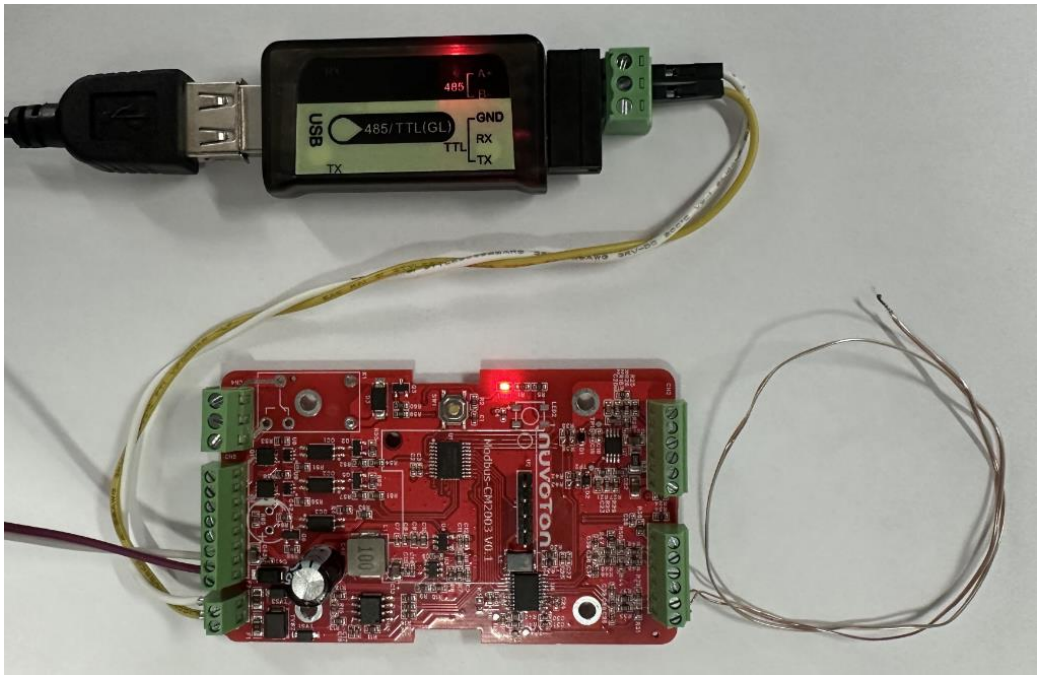

圖 3-2 實物接線

4. 目錄資訊

目錄結構如下圖所示：

EC_CM2003_ModbusRTU_Thermocouple_V1.0	
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 *Modbus_Thermocouple.uvprojx*
2. 進入編譯模式介面
   - 編譯
   - 下載程式碼至記憶體
   - 進入 / 離開模擬模式
3. 進入模擬模式介面
   - 執行程式碼

6. 修訂紀錄

Date	Revision	Description
2025.05.28	1.00	初始發布。

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.