M2A23 控制儀表指針

NuMicro® 32位元系列微控制器範例程式碼介紹

文件資訊

應用簡述	本範例程式碼使用 M2A23 來實現農機儀表板指針的控制
BSP 版本	M2A23_Series_BSP_CMSIS_V3.00.001
開發平臺	NuMaker-M2A23SG 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

本範例程式碼基於 M2A23 平臺，實現對專用兩相四線步進馬達的精確控制，適用於農機儀表板及各類儀器儀表的指針驅動場景。系統主要包含兩個核心功能：

開機自動歸零校準：系統啟動時先定位機械零點以確立物理基準，隨後校準至儀表板零點，從而確保指針指示的準確性。

週期性往復運動：初始化完成後，控制指針在「零點」與「設定目標位置」之間進行週期性的往復擺動，以驗證驅動效果。

1.1 原理

系統原理主要包含機械傳動與電氣驅動兩個部分。

機械傳動結構如圖 1-1 所示，步進馬達轉子透過齒輪組驅動指針軸轉動。

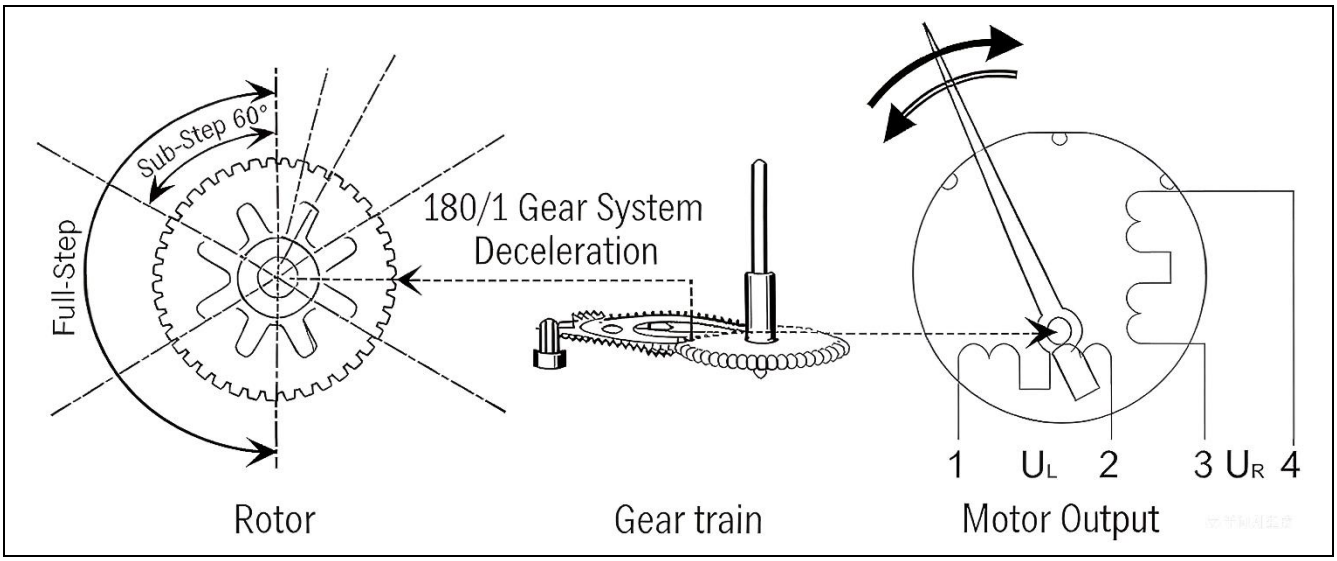

圖 1-1 步進馬達機械傳動系統結構

- **減速比**：180:1。即轉子旋轉 180°，輸出軸（指針）僅旋轉 1°。高減速比設計有助於提高指針的定位精度和保持扭力。
- **步進角度定義**：
  - 分步：轉子轉動 60°。
  - 全步：轉子轉動 180°。

電氣驅動時序如圖 1-2 所示，馬達採用六步換相法驅動。藉由改變 GPIO 的輸出順序，可控制馬達的旋轉方向。

正轉序列：1 → 2 → 3 → 4 → 5 → 6

反轉序列：6 → 5 → 4 → 3 → 2 → 1

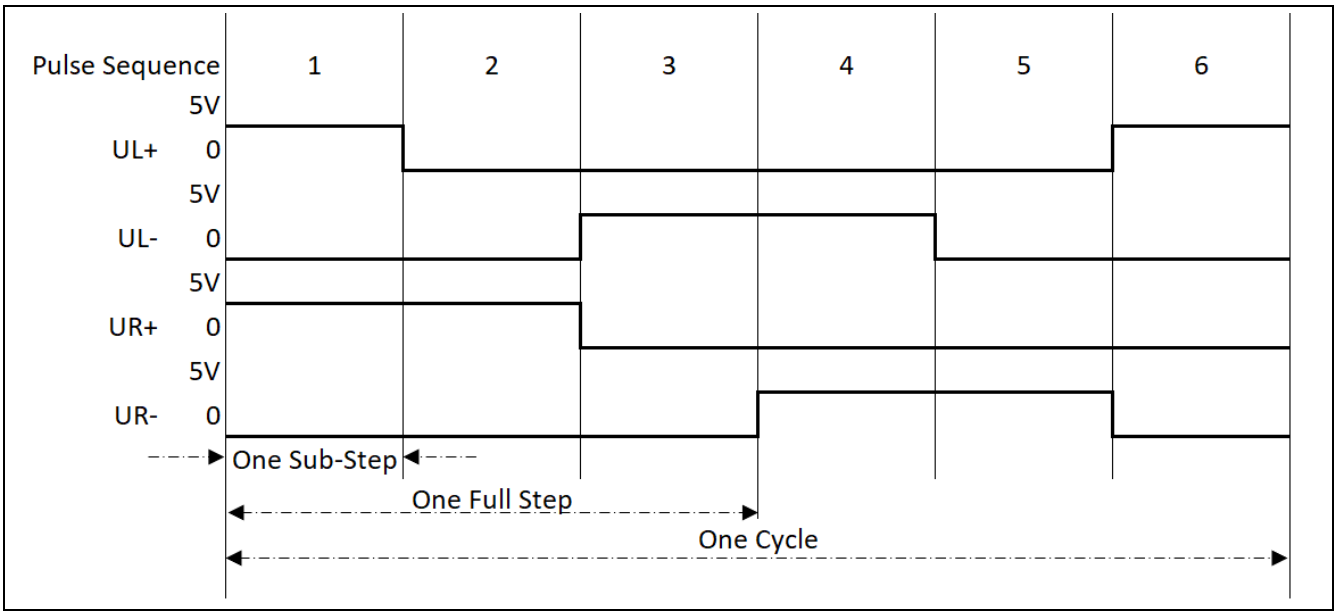

圖 1-2 步進馬達控制時序

本範例使用 GPIO 和 TIMER 協同工作，實現對步進馬達的控制。

GPIO：連接馬達的兩相雙極繞組，透過輸出特定的電位序列驅動轉子步進運動。

TIMER：提供精確的時間基準，用於控制步進脈衝的頻率，從而調節馬達的轉動速度。

1.2 執行結果

PC 上打開終端機，連接 UART0 用於輸出除錯資訊，如圖 1-3 所示。

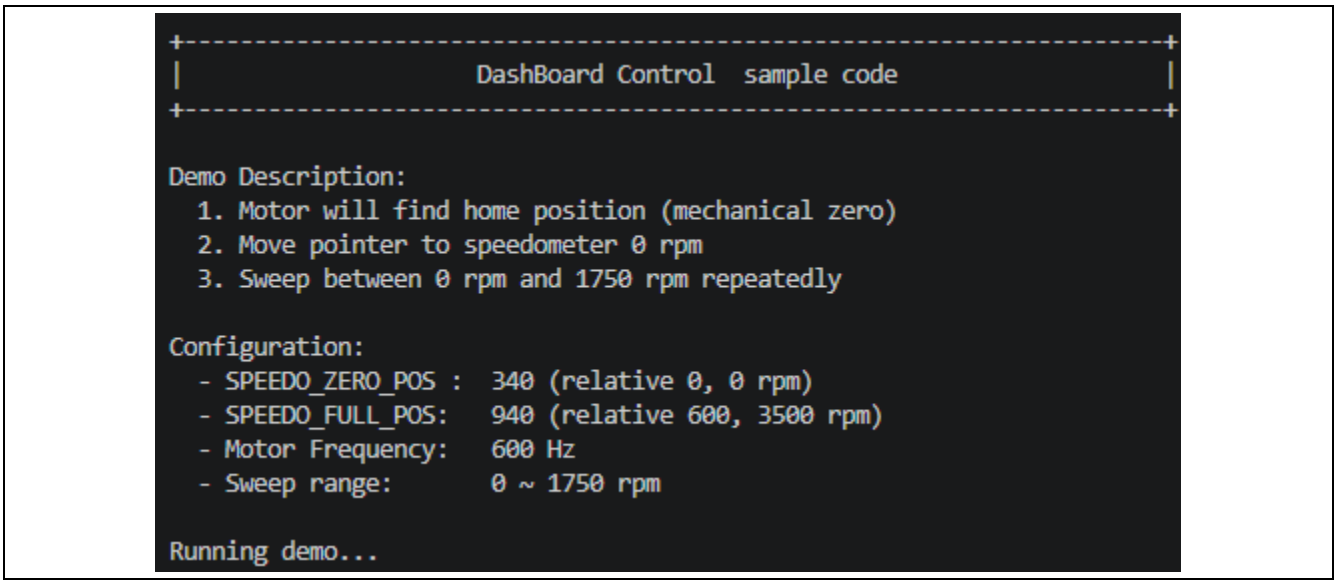
+--+
| DashBoard Control sample code |
+--+
Demo Description:
1. Motor will find home position (mechanical zero)
2. Move pointer to speedometer 0 rpm
3. Sweep between 0 rpm and 1750 rpm repeatedly
Configuration:
- SPEEDO_ZERO_POS : 340 (relative 0, 0 rpm)
- SPEEDO_FULL_POS: 940 (relative 600, 3500 rpm)
- Motor Frequency: 600 Hz
- Sweep range: 0 ~ 1750 rpm
Running demo...

圖 1-3 執行結果

2. 程式碼介紹

程式碼位於 *main.c*。

巨集定義

定義馬達控制腳位、最大步數、速度表零點和終點、馬達的轉動速度等。

```
/*****************************************************************************
 * Motor Pin Definitions
 * GPIO pins used for stepper motor coil control
 *****************************************************************************/
#define UA_POS PB5    // Motor A phase positive (GPIO PB5)
#define UA_NEG PB4    // Motor A phase negative (GPIO PB4)
#define UB_POS PB3    // Motor B phase positive (GPIO PB3)
#define UB_NEG PB2    // Motor B phase negative (GPIO PB2)

/*****************************************************************************
 * Motor Position Definitions
 * All positions are absolute steps from mechanical zero point
 *****************************************************************************/
#define STEP_MAX 1080             // Maximum steps for homing (mechanical zero)
#define SPEEDO_ZERO_POS 340       // Speedometer zero position (absolute, 0 rpm reference)
#define SPEEDO_FULL_POS 940       // Speedometer full scale position (absolute, 3500 rpm)

/*****************************************************************************
 * Motor Control Configuration
 *****************************************************************************/
#define MOTOR_FREQ 600            // Timer interrupt frequency (Hz)
                                  // Controls motor stepping speed
                                  // 600 Hz = 600 steps/second max
#define FULL_STEP_COUNT 6         // Number of steps in one full electrical cycle

#define SPEED_MAX 3500            // Maximum speed on speedometer
                                  // Range: 0 ~ 3500 rpm
```

馬達控制參數

六步換相時序、狀態機、馬達步數、指針位置、方向等。

```
const MotorStep full_step_sequence[] =
{
    {1, 0, 1, 0},  // Step 1: A +5V, B +5V
    {0, 0, 1, 0},  // Step 2: A 0V,  B +5V
    {0, 1, 0, 0},  // Step 3: A -5V, B 0V
    {0, 1, 0, 1},  // Step 4: A -5V, B +5V
    {0, 0, 0, 1},  // Step 5: A 0V,  B -5V
    {1, 0, 0, 0}   // Step 6: A +5V, B 0V
};

......

volatile uint8_t machine_state = MOTOR_STATE_IDLE;    // Current motor state
volatile uint32_t total_step = 0;                     // Remaining steps to move
```

```
volatile uint32_t delay = 0;                          // Software delay counter

......

volatile int16_t speedo_ptr_pos = 0;   // Speedometer pointer position (0 to 600)
volatile uint8_t direction = MOTOR_DIR_FORWARD;  // Current motor direction
```

系統初始化

啟用 TIMER0 的時脈、設定馬達控制腳位。

```
void SYS_Init(void)
{
    ......
    /* Enable TIMER module clock */
    CLK_EnableModuleClock(TMR0_MODULE);

    /* Select TIMER clock source */
    CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0SEL_PCLK0, 0);

    ......

    /* Set multi-function pins for UART0 RXD and TXD */
    GPIO_SetMode(PB, BIT5, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT4, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT3, GPIO_MODE_OUTPUT);
    GPIO_SetMode(PB, BIT2, GPIO_MODE_OUTPUT);
}
```

馬達初始化

初始化馬達輸出以及相關全域變數。

```
void Motor_Init(void)
{
    // Disable all motor coils (set all pins to LOW)
    UA_POS = 0;
    UA_NEG = 0;
    UB_POS = 0;
    UB_NEG = 0;

    // Reset step counter (no pending movement)
    total_step = 0;

    // Set default motor direction
    direction = MOTOR_DIR_FORWARD;

    // Set motor state to idle
    machine_state = MOTOR_STATE_IDLE;

    // Reset speedometer pointer position to zero (0 rpm)
    speedo_ptr_pos = 0;
}
```

馬達單步驅動

執行步進馬達單步控制，採用六步查表法驅動馬達。

```
void Motor_Step(uint8_t direction)
{
    static uint32_t current_phase = 0;      // Internal phase state tracking
    static uint8_t last_direction = MOTOR_DIR_FORWARD;    // Track last direction
                                                          for smooth reversal

    // Handle direction change - adjust phase to avoid repetition
    if (direction != last_direction)
    {
        // Adjust phase index for smooth direction transition
        if (direction == MOTOR_DIR_FORWARD) // Switch to Forward
        {
            current_phase = (current_phase + 2) % FULL_STEP_COUNT;
        }
        else  // Switch to Reverse
        {
            current_phase = (current_phase + 4) % FULL_STEP_COUNT;
        }

        last_direction = direction;
    }

    // Directly assign values from full_step_sequence to GPIO pins
    UA_POS = full_step_sequence[current_phase].A1;
    UA_NEG = full_step_sequence[current_phase].A2;
    UB_POS = full_step_sequence[current_phase].B1;
    UB_NEG = full_step_sequence[current_phase].B2;

    // Update phase for next call (AFTER)
    if (direction == MOTOR_DIR_FORWARD) // Forward
    {
        current_phase = (current_phase + 1) % FULL_STEP_COUNT;
    }
    else  // Reverse
    {
        current_phase = (current_phase - 1 + FULL_STEP_COUNT) % FULL_STEP_COUNT;
    }
}
```

馬達啟動與停止控制系統

計時器中斷會自動負責馬達的相位切換，這兩個函式透過調整共用變數來控制馬達的開啟與停止，實現主程式和中斷之間的分離。

```
void Motor_Start(uint8_t dir, uint32_t steps)
{
    if (steps == 0)
        return;  // No movement needed

    direction = dir;
```

```
    machine_state = (dir == MOTOR_DIR_FORWARD) ? MOTOR_STATE_FORWARD :
                                                 MOTOR_STATE_REVERSE;
    total_step = steps;
}
/**
 * @brief Stop motor immediately
 */
void Motor_Stop(void)
{
    total_step = 0;
    machine_state = MOTOR_STATE_IDLE;

    UA_POS = 0;
    UA_NEG = 0;
    UB_POS = 0;
    UB_NEG = 0;
}
```

速度表指針歸零校準

為確保指針指示的準確性，系統在啟動時必須執行自動歸零校準流程。

- 尋找機械零點：馬達反向旋轉到機械零點。
- 校準速度表零點：從機械零點正向旋轉至速度表零點。

```
void Speedo_ZeroCalibration(void)
{
    // Step 1: Reverse to mechanical zero point (Home Position)
    direction = MOTOR_DIR_REVERSE;
    total_step = STEP_MAX;

    while (total_step > 0); // Wait until homing complete (step counts down to 0)

    // Step 2: Forward to speedometer working zero position (0 RPM reference)
    direction = MOTOR_DIR_FORWARD;
    total_step = SPEEDO_ZERO_POS;

    while (total_step);

    // Step 3: Reset relative position counter to 0
    speedo_ptr_pos = 0;
}
```

速度表指針控制

將速度值對應為馬達控制參數（方向和步數），實現速度表指針的精確位置控制。

```
uint8_t Speedo_SetSpeed(uint32_t speed)
{
    uint32_t target_position;
    int32_t steps_to_move;
    uint8_t dir;
```

```
    // Validate speed range
    if (speed > SPEED_MAX)
    {
        return 2;  // Invalid speed value
    }

    // Convert speed to speedometer pointer position
    // Linear mapping: speed 0~3500 → position 0~600
    if (speed > SPEED_MAX)
        speed = SPEED_MAX;

    target_position = (speed * (SPEEDO_FULL_POS - SPEEDO_ZERO_POS)) / SPEED_MAX;

    // Validate position range
    if (target_position < 0 || target_position > (SPEEDO_FULL_POS - SPEEDO_ZERO_POS))
    {
        return 2;  // Invalid position
    }

    // Calculate steps needed to reach target speed
    if (target_position > speedo_ptr_pos)
    {
        steps_to_move = target_position - speedo_ptr_pos;
        dir = MOTOR_DIR_FORWARD;   // Move pointer toward higher speed
    }
    else if (target_position < speedo_ptr_pos)
    {
        steps_to_move = speedo_ptr_pos - target_position;
        dir = MOTOR_DIR_REVERSE;   // Move pointer toward lower speed
    }
    else
    {
        return 1;  // Already at target speed
    }

    // Start motor movement to update speedometer
    Motor_Start(dir, steps_to_move);

    return 0;  // Success
}
```

計時器初始化

初始化並啟動計時器 TIMER0，為馬達控制和軟體延時提供精確的時間基準。頻率由巨集定義 MOTOR_FREQ 決定。

```
void TIMER0_Init(void)
{
    // Configure Timer0 in periodic mode
    // Interrupt frequency: MOTOR_FREQ Hz (600 Hz = 600 steps/second)
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, MOTOR_FREQ);

    // Enable Timer0 interrupt
    TIMER_EnableInt(TIMER0);
```

```
    // Enable NVIC interrupt for TMR0
    NVIC_EnableIRQ(TMR0_IRQn);

    // Start Timer0 counting
    TIMER_Start(TIMER0);
}
```

計時器中斷服務程式

處理馬達步進、位置更新以及軟體延時計數。

```
void TMR0_IRQHandler(void)
{
    // Check Timer0 interrupt flag
    if (TIMER_GetIntFlag(TIMER0) == 1)
    {
        /* Clear Timer0 time-out interrupt flag */
        TIMER_ClearIntFlag(TIMER0);

        // Execute motor step if remaining steps exist
        if (total_step > 0)
        {
            // Execute one motor step
            Motor_Step(direction);

            // Decrement remaining step count
            total_step--;

            // Update speedometer pointer position
            if (machine_state == MOTOR_STATE_FORWARD)
                speedo_ptr_pos++;      // Forward: pointer moves toward higher speed
            else if (machine_state == MOTOR_STATE_REVERSE)
                speedo_ptr_pos--;      // Reverse: pointer moves toward lower speed
        }
        else
        {
            // Steps complete, set motor state to idle
            machine_state = MOTOR_STATE_IDLE;
        }

        // Decrement delay counter (if delay in progress)
        if (delay > 0)
            delay--;
    }
}
```

主程式

依序執行硬體初始化、馬達歸零校準，並進入無窮迴圈以展示速度表指針的往復運動。

```
int32_t main(void)
{
    ……
```

```
    // Initialize motor variables and pins
    Motor_Init();

    // Initialize Timer0 for motor stepping control
    TIMER0_Init();

    // Perform speedometer zero calibration (find mechanical zero, then move to 0 rpm)
    Speedo_ZeroCalibration();

    // Main loop - continuously sweep speedometer pointer
    while (1)
    {
        // Short delay at current position (1 second)
        Delay(MOTOR_FREQ * 1);

        // Wait for any pending movement to complete
        while (Motor_IsBusy());

        // Set speedometer to mid speed 1750 rpm
        Speedo_SetSpeed(SPEED_MAX/2);

        // Wait for movement to complete
        while (Motor_IsBusy());

        // Short delay at mid speed position (1 second)
        Delay(MOTOR_FREQ * 1);

        // Wait for any pending movement to complete
        while (Motor_IsBusy());

        // Set speedometer to zero (0 rpm)
        Speedo_SetSpeed(0);

        // Wait for movement to complete
        while (Motor_IsBusy());
    }
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M2A23_Series_BSP_CMSIS_V3.00.001
- IDE 版本
  - Keil uVision 5.43
  - VS Code 1.115.3

3.2 硬體需求

- 電路元件
  - NuMaker-M2A23SG V0.1
- 線路示意圖
  - 將 MCU 的 PB.2 / PB.3 / PB.5 / PB.4 分別連接到步進馬達的 1 / 2 / 3 / 4 腳位。
  - 將 UART0 TX （PB.13）腳位透過 UART 轉 USB 工具連接到 PC，以顯示範例程式碼的執行結果。

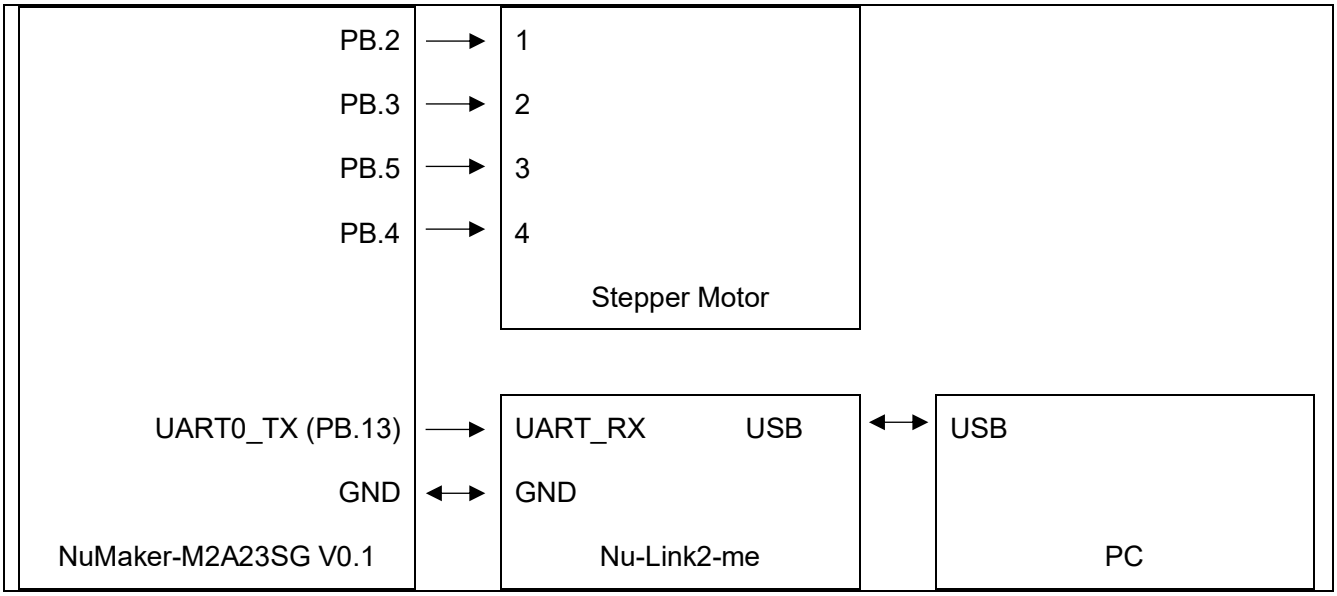

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

EC_M2A23_DashBoard_Control_V1.00	
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

5.1 Keil 平臺

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，按兩下 M2A23_DashBoard_Control.uvprojx。
2. 進入編譯模式介面
   - 編譯
   - 下載程式碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行程式碼

5.2 VS Code 平臺

1. 選擇 File
2. 選擇 Open Folder
3. 選擇 ExampleCode 路徑中的 VSCode 資料夾
4. 選擇 CMSIS
   - 按一下「Build solution」編譯
   - 按一下「Load & Run application」下載程式碼至記憶體
5. 進入除錯模式介面
   - 按一下「Load & Debug application」執行程式碼

6. 修訂紀錄

Date	Revision	Description
2026.04.15	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.