M251/M252 PWM Capture Rotary

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 M251/M252 PWM Capture 來偵測 Rotary Encoder Switch 旋轉方向及計數
BSP 版本	M251/M252/M254/M256/M258 Series BSP CMSIS V3.02.007
開發平台	NuMaker-M252SD

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

此範例程式碼使用 M251/M252 系列以抓取 Rotary Encoder Switch 所產生 A 項和 B 項方波訊號，藉由判斷 A 項和 B 項相對位置來確認其旋轉方向和記錄旋轉位置並透過一組 UART 印出擷取結果。

1.1 原理

PWM capture 主要透過設定中斷，中斷方式可選擇上升緣觸發、下降緣觸來測量外部訊號的週期、頻率、佔空比。每個通道中可記錄最近一次觸發時的計數器值，上升緣觸發的計數值及下降緣觸計數值。

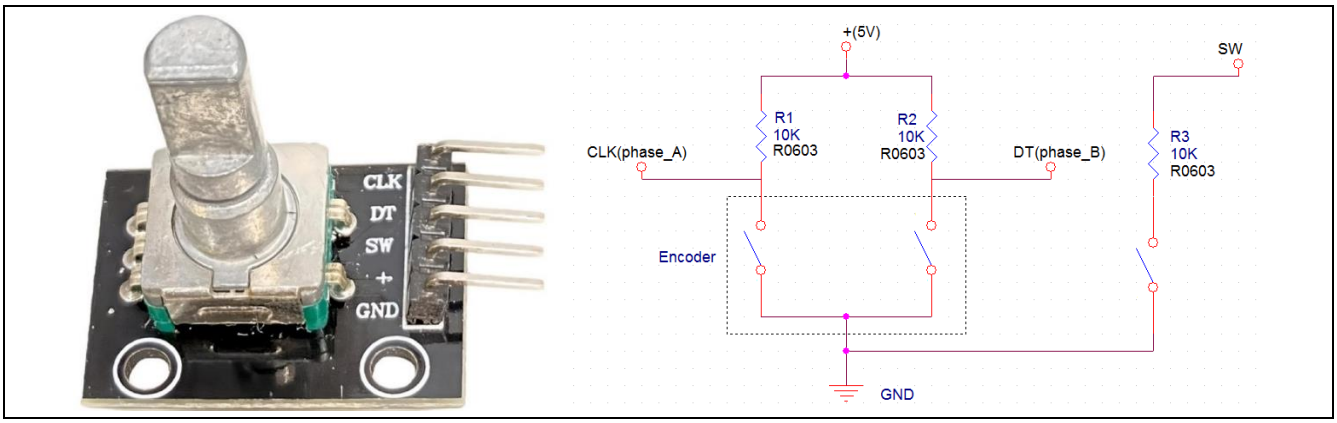

圖 1-1 Rotary Encoder Switch

圖 1-1 為左側為 Rotary Encoder Switch 的實體元件，右側為 Rotary Encoder Switch 的電路，其中有「GND」、「+」、「 SW」、「DT」、「CLK」5 個 PIN，接法為「GND」接電路中的 VSS，「+」接 DC 5V，「 SW」為按鍵開關功能此範例中不使用，「DT」和「CLK」會產生時序不一樣的方波。旋轉方向為順時針時「CLK」產生的方波會超前「DT」產生的方波如圖 1-2 及如圖 1-3 所示，分別為起始上緣觸發和起始下緣觸發；反之旋轉方向為逆時針時「CLK」產生的方波會落後「DT」產生的方波如圖 1-4 及如圖 1-5 所示，分別為起始上緣觸發和起始下緣觸發，本範例的 Rotary Encoder Switch 旋轉一圈「DT」和「CLK」會各自產生 15 個方波，如圖 1-6 及圖 1-7 所示。

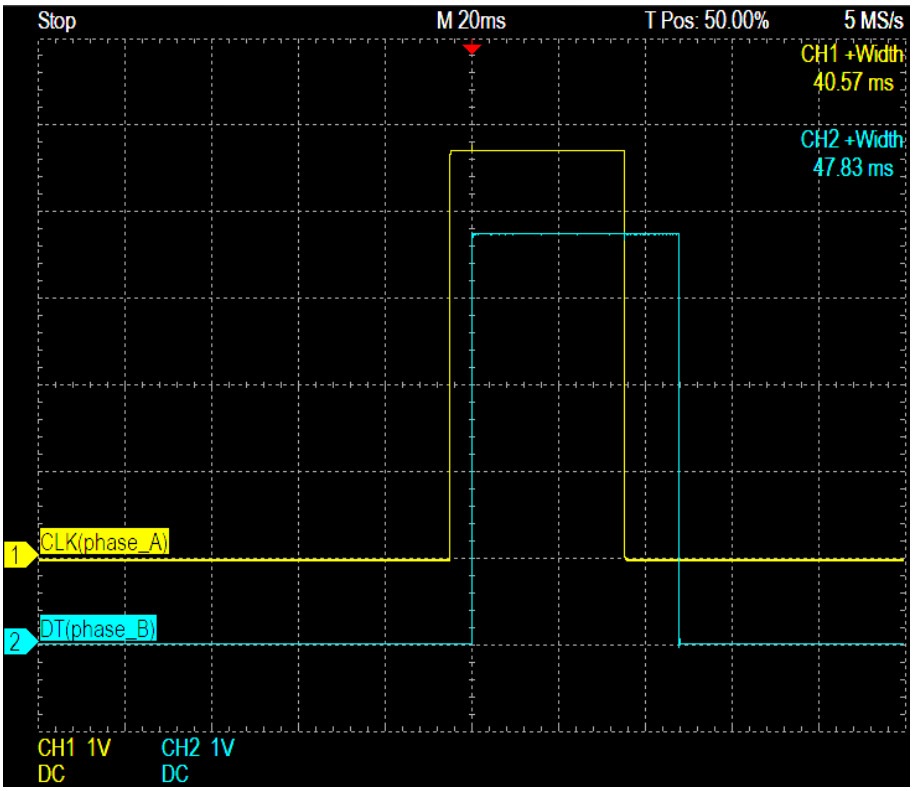

圖 1-2 Rotary Encoder Switch 順時針旋轉（起始上緣觸發）

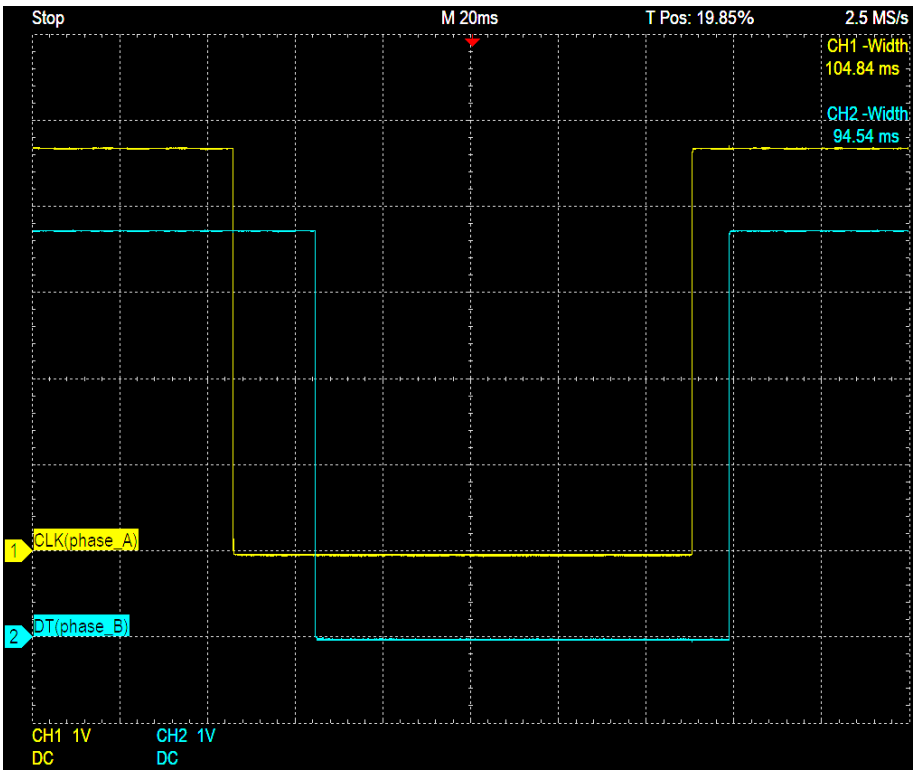

圖 1-3 Rotary Encoder Switch 順時針旋轉（起始下緣觸發）

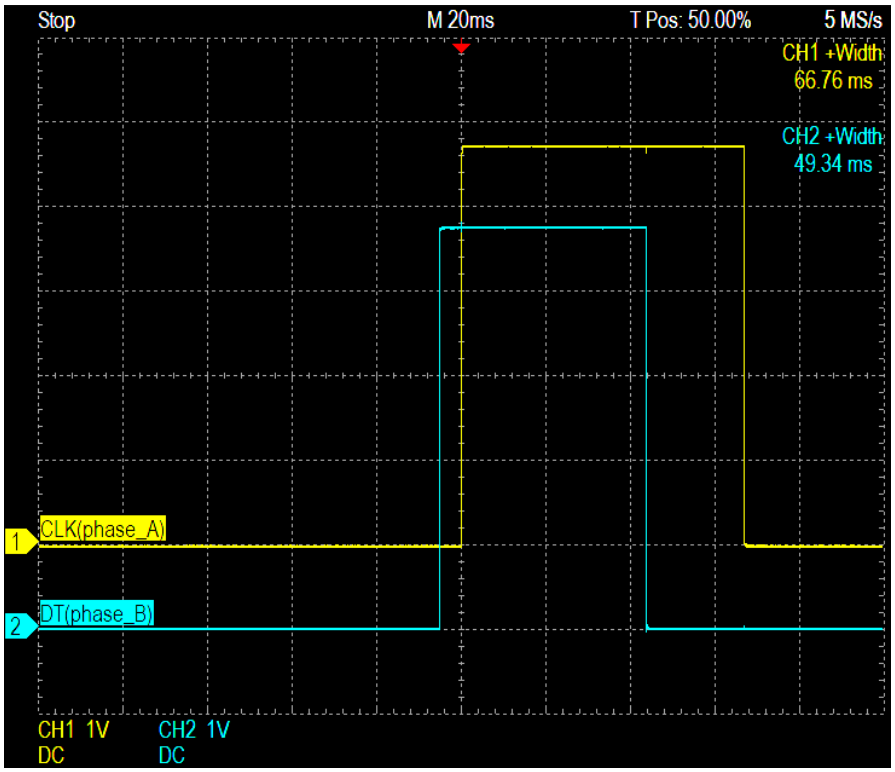

圖 1-4 Rotary Encoder Switch 逆時針旋轉（起始上緣觸發）

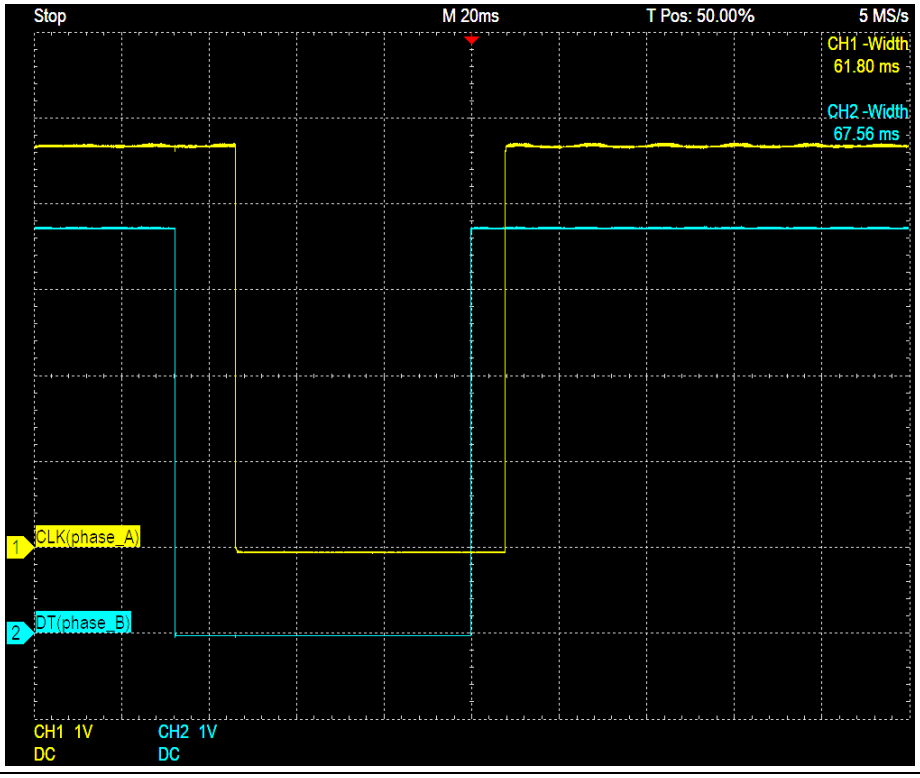

圖 1-5 Rotary Encoder Switch 逆時針旋轉（起始下緣觸發）

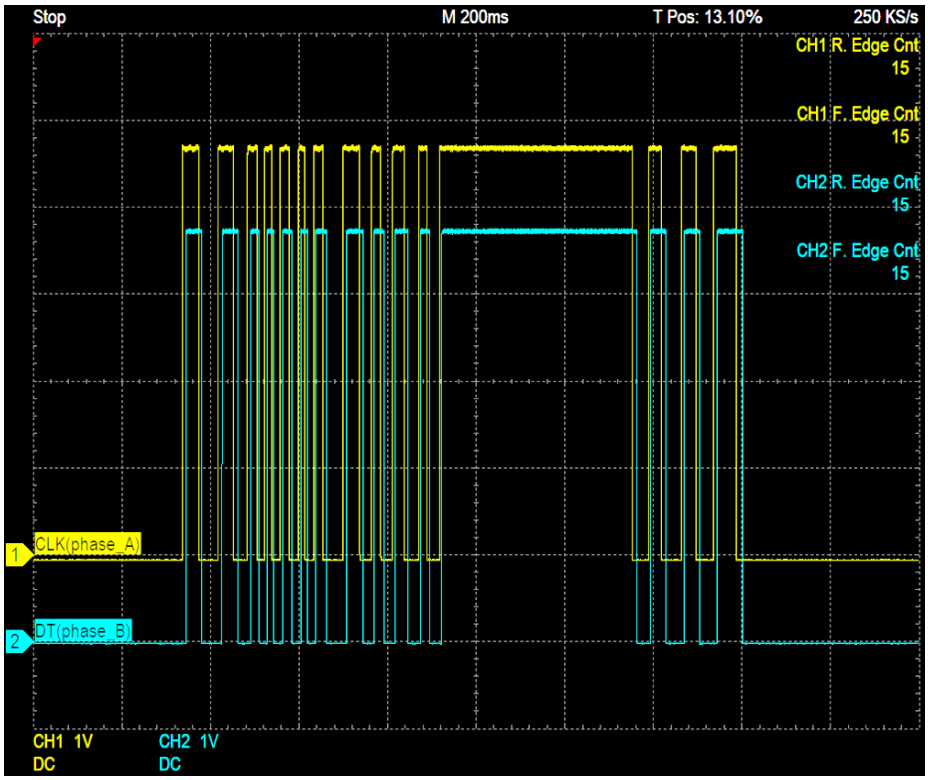

圖 1-6 Rotary Encoder Switch 順時針旋轉一圈

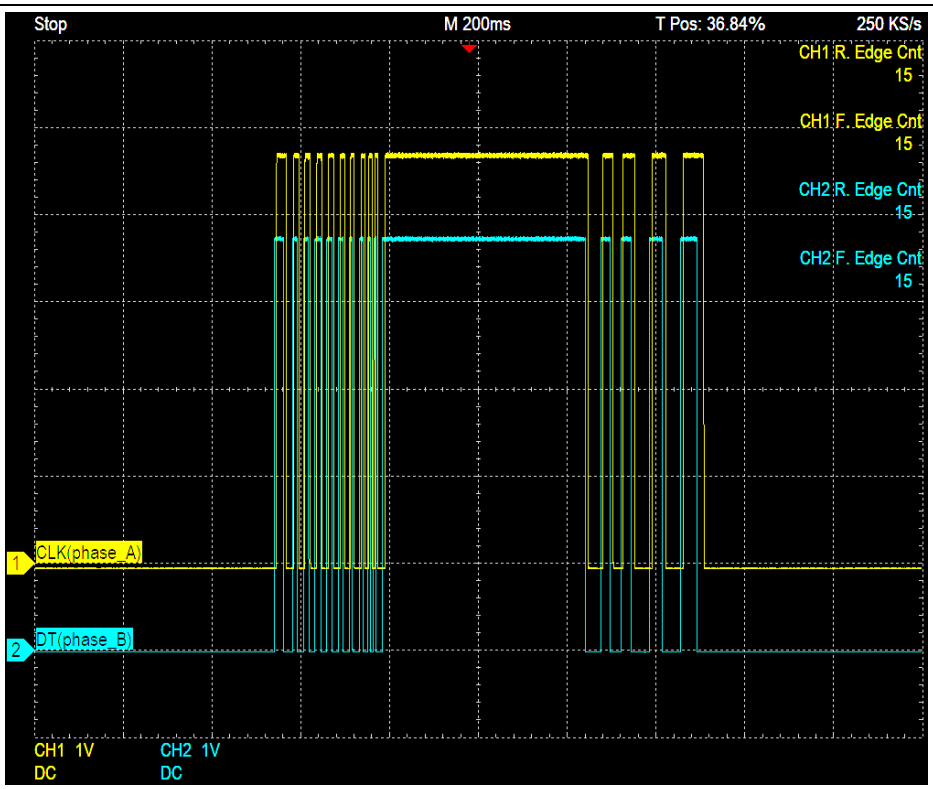

圖 1-7 Rotary Encoder Switch 逆時針旋轉一圈

由於 Rotary Encoder Switch 是機械式開關，所產生的方波 CLK(phase_A)和 DT(phase_B)會不一樣，如圖 1-2 所示 CLK(phase_A)的正脈波寬為 40.57 ms、DT(phase_B) 正脈波寬為 47.83 ms；如圖 1-3 所示 CLK(phase_A)的負脈波寬為 104.84 ms、DT(phase_B) 負脈波寬為 94.54 ms，從上述觀察得知同一次中旋轉產生 CLK(phase_A)和 DT(phase_B)兩方波時間誤差較大不適合使用 PWM Capture 自帶的 CAPDATn(上緣觸發計數值 RCAPDATn、下緣觸發計數值 FCAPDATn)來判斷方波超前或是落後。

故本範例使用 PWM Capture 將 CLK(phase_A)和 DT(phase_B)產生的上緣觸發中斷、下緣觸發中斷搭配旋轉編碼器方向查找表（正交解碼器）且記錄前一個狀態 CLK(phase_A)及 DT(phase_B)，當前狀態 Current state = (phase_A << 1) | phase_B；也就是說將 phase_A 狀態當 MSB，phase_B 狀態當 LSB，組合成 2 Bits 的 Data 就是當前狀態 Current state，如圖 1-8 和圖 1-9 分別為順時針旋轉和逆時針旋轉產生的當前狀態值 Current state。

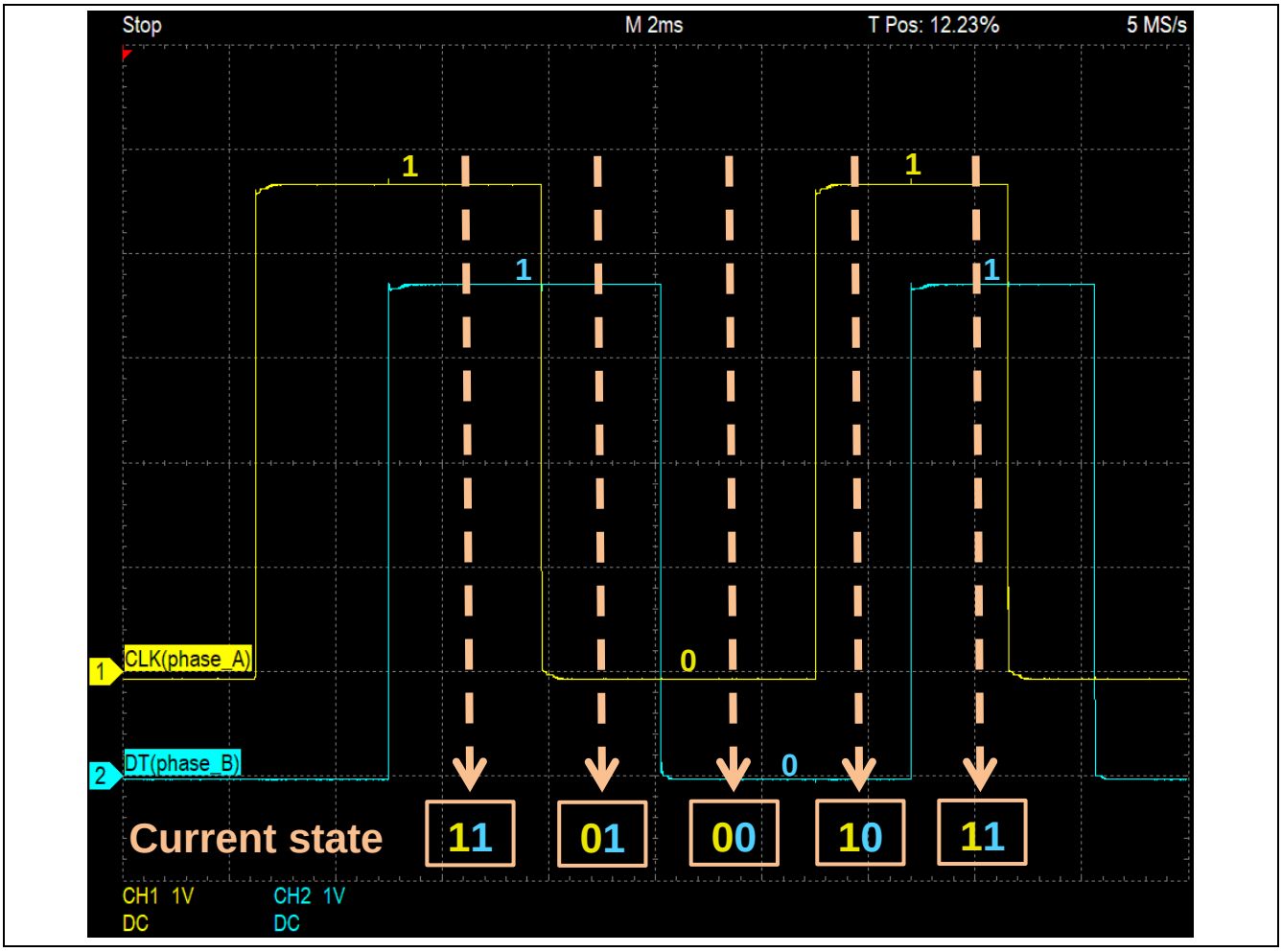

圖 1-8 Rotary Encoder Switch 順時針旋轉

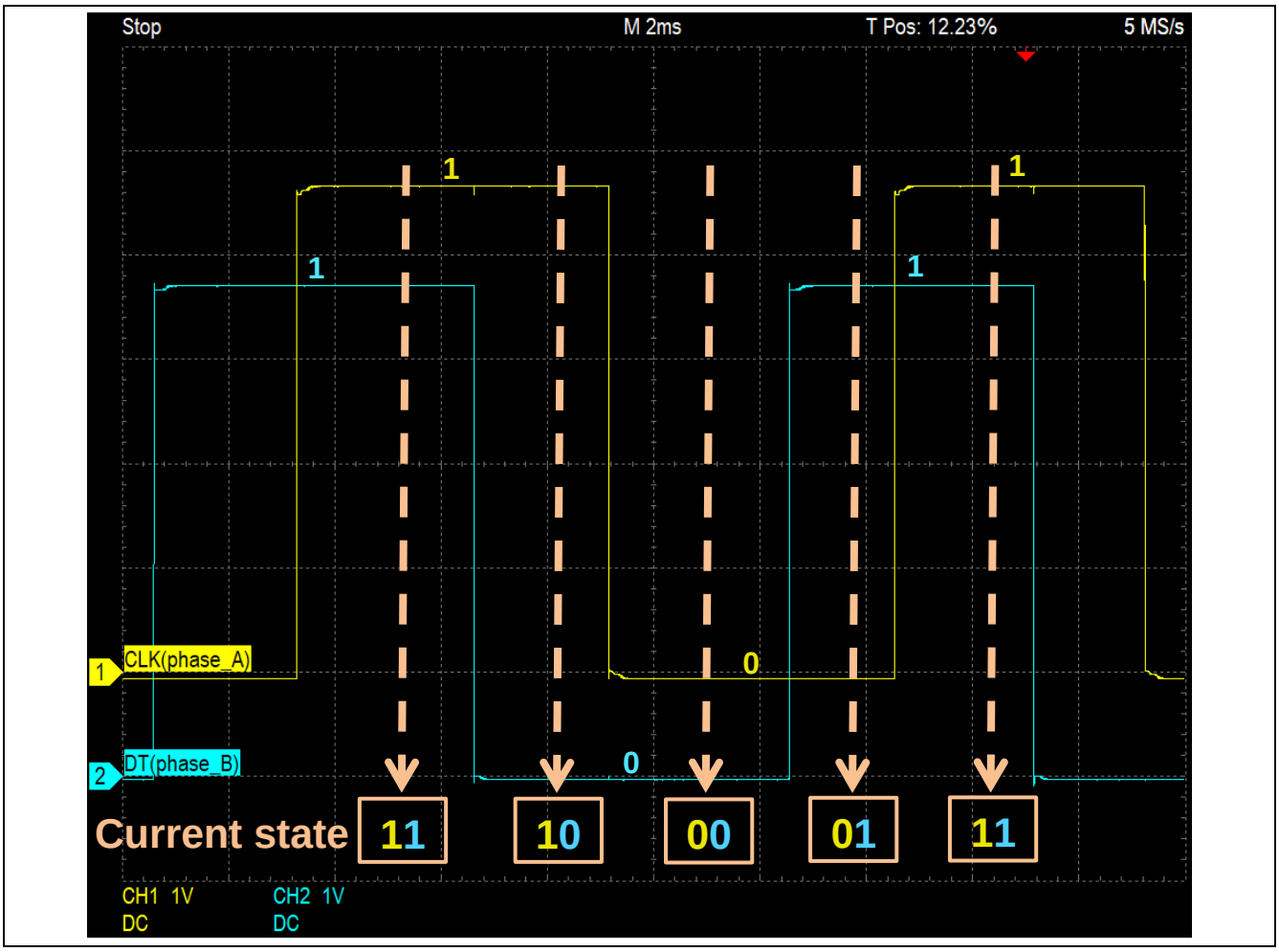

圖 1-9 Rotary Encoder Switch 逆時針旋轉

順時針旋轉當前狀態 Current state 11→01、01→00、00→10、10→11 有 4 種順序，逆時針旋轉 11→10、10→00、00→01、01→11 也是有 4 種順序，順時針旋轉與逆時針旋轉總共有 8 種順序，將其當前狀態 Current state 和上一個狀態 Last state 組合起來，可作為旋轉狀態查表依據如表 1-1；上一個狀態 Last state 的 2 Bits 為 MSB，當前狀態 Current state 的 2 Bits 為 LSB，也就是 Transition = (Last state << 2) | current state 。

順時針旋轉	逆時針旋轉
1101	1110
0100	1000
0010	0001
1011	0111

表 1-1 順時針旋轉與逆時針旋轉的上一個狀態和當前狀態表

索引 (十進位)	上一個狀態|當前狀態 Transition = (Last state << 2) | Current state (二進位)	有效性	旋轉方向	0=無旋轉/無效 1=順時針旋轉 -1=逆時針旋轉
0	0000	無	無改變	0
1	0001	有	逆時針旋轉	-1
2	0010	有	順時針旋轉	1
3	0011	無	無	0
4	0100	有	順時針旋轉	1
5	0101	無	無	0
6	0110	無	無	0
7	0111	有	逆時針旋轉	-1
8	1000	有	逆時針旋轉	-1
9	1001	無	無	0
10	1010	無	無改變	0
11	1011	有	順時針旋轉	1
12	1100	無	無	0
13	1101	有	順時針旋轉	1
14	1110	有	逆時針旋轉	-1
15	1111	無	無改變	0

表 1-2 上一個狀態和當前狀態組合的 4 Bits 所有情況

如表 1-2 為上一個狀態和當前狀態組合的 4 Bits 所有情況，有 8 種為有效編碼另外 8 種為無效編碼，由於 Rotary Encoder Switch 為機械開關，在旋轉途中可能會有彈跳現象，彈跳產生的雜訊會是上述表 1-2 無效編碼情況，只需記錄有效編碼情況可用於除彈跳現象，最後將順時針旋轉替換為「1」，逆時針旋轉替換為「-1」，無效輸出替換為「0」。

1.2 執行結果

本範例程式碼會有三種執行結果說明，第一種為單純順時針旋轉，第二種為單為單純逆時針旋轉，第三種為順時針旋轉與逆時針交替旋轉。

在程式中 phase_A 和 phase_B 分別使用兩種觸發中斷，分別為上緣觸發中斷和下緣觸發中斷，

如圖 1-10 所示，

- 時序①時為 phase_A 的上緣觸發
- 時序②時為 phase_B 的上緣觸發
- 時序③時為 phase_A 的下緣觸發
- 時序④時為 phase_B 的下緣觸發
- 時序⑤時為 phase_A 的上緣觸發
- 時序⑥時為 phase_B 的上緣觸發
- 時序⑦時為 phase_A 的下緣觸發
- 時序⑧時為 phase_B 的下緣觸發

上述情況使用 phase_A 和 phase_B 都會進入上緣觸發中斷和下緣觸發中斷，所以在連續 2 次偵測 Transition 值都為順時針時才是真的順時針旋轉一格，在 Rotary Encoder Switch 的 Count 值才加 1；反之連續 2 次偵測 Transition 值都為逆時針時才是真的逆時針旋轉一格，在 Rotary Encoder Switch 的 Count 值才減 1。

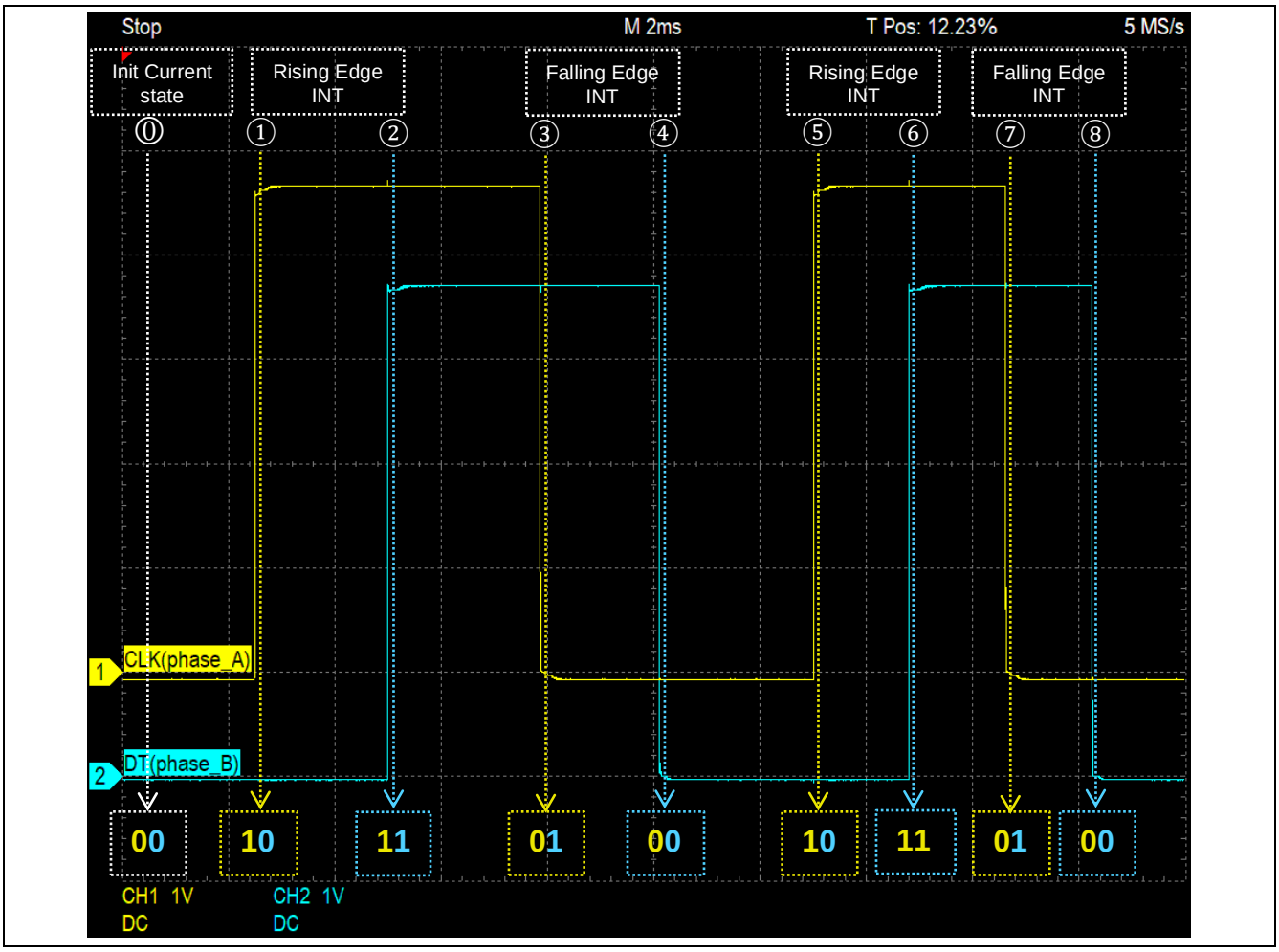

圖 1-10 Rotary Encoder Switch 順時針旋轉

Current state = (phase_A << 1) | phase_B, Transition = (Last state << 2) | Current state

- 時序⓪時初始化 Current state=00(二進位)
- 時序①時 Transition=0010(二進位)，索引= 2(十進位) ，查表 1-2 為順時針旋轉→**“1”**
- 時序②時 Transition=1011(二進位) ，索引=11(十進位)，查表 1-2 為順時針旋轉→**“1”**

→ 連續 2 次出現 “1” → 顯示“ Clockwise(CW) Count= 1”

- 時序③時 Transition=1101(二進位)，索引=13(十進位)，查表 1-2 為順時針旋轉→**“1”**
- 時序④時 Transition=0100(二進位)，索引= 4(十進位) ，查表 1-2 為順時針旋轉→**“1”**

→ 連續 2 次出現 “1” → 顯示“ Clockwise(CW) Count= 2”

- 時序⑤時 Transition=0010(二進位)，索引= 2(十進位) ，查表 1-2 為順時針旋轉→**“1”**
- 時序⑥時 Transition=1011(二進位)，索引=11(十進位)，查表 1-2 為順時針旋轉→**“1”**

→ **連續 2 次出現 “1” → 顯示“ Clockwise(CW) Count= 3”**

- 時序⑦時 Transition=1101(二進位)，索引=13(十進位)，查表 1-2 為順時針旋轉→**“1”**
- 時序⑧時 Transition=0100(二進位)，索引= 4(十進位) ，查表 1-2 為順時針旋轉→**“1”**

→ **連續 2 次出現 “1” → 顯示“ Clockwise(CW) Count= 4”**

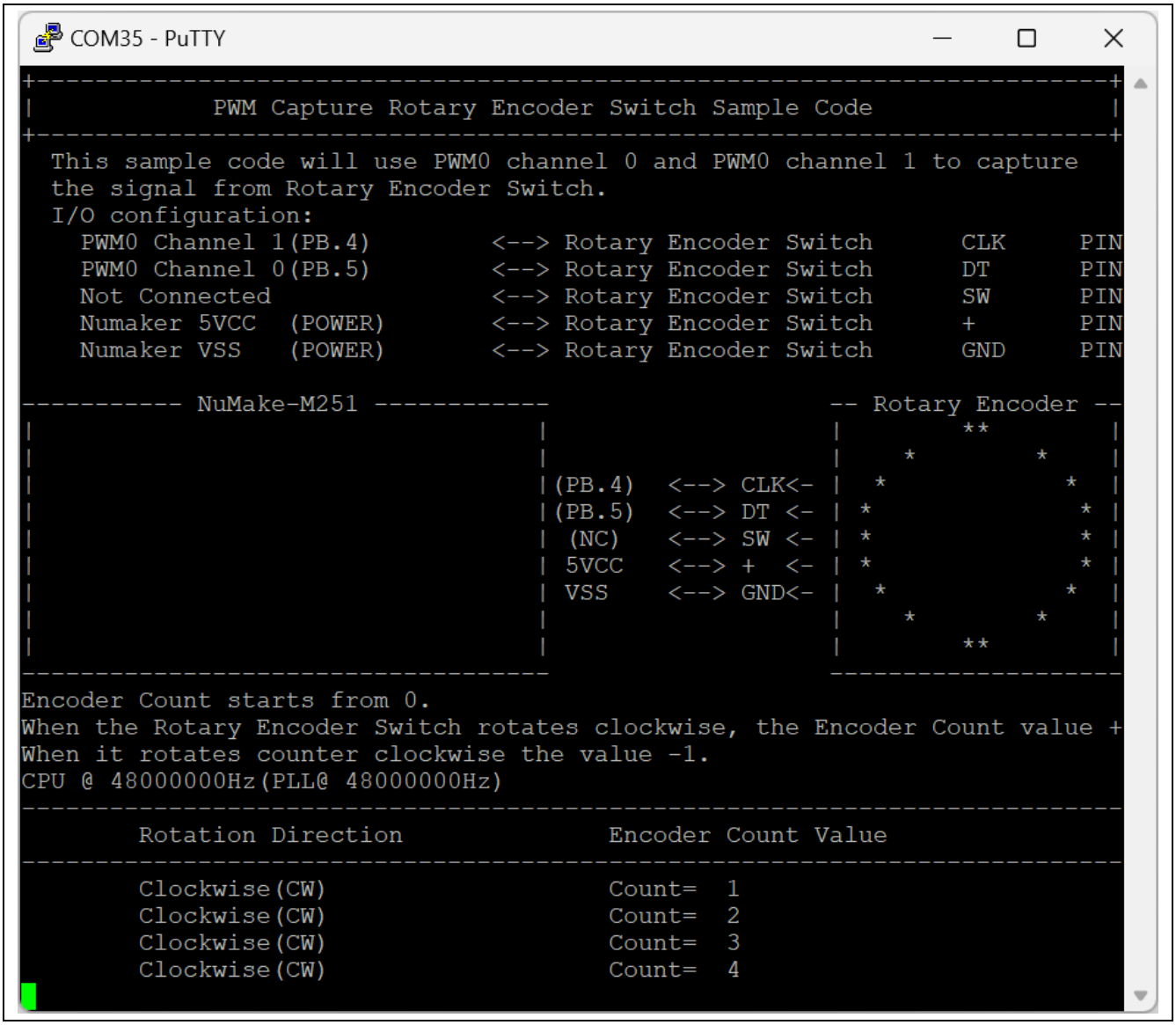

圖 1-11 順時針旋轉執行結果

如圖 1-10 Rotary Encoder Switch 順時針旋轉時序⓪→⑧執行結果為圖 1-11 所示。

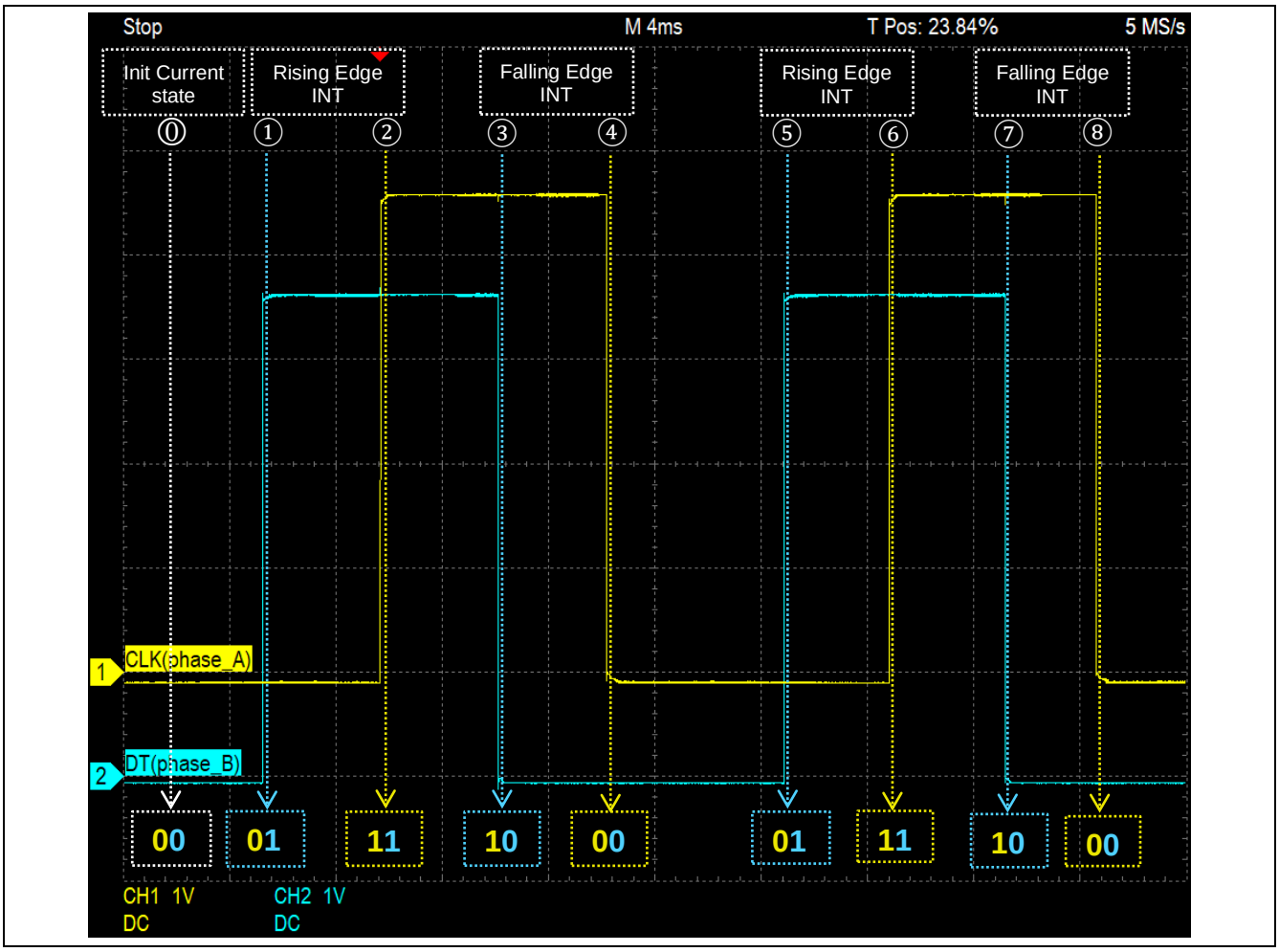

圖 1-12 Rotary Encoder Switch 逆時針旋轉

Current state = (phase_A << 1) | phase_B, Transition = (Last state << 2) | Current state

- 時序⓪時初始化 Current state=00(二進位)
- 時序①時 Transition=0001(二進位)，索引= 1(十進位) ，查表 1-2 為逆時針旋轉→“-1”
- 時序②時 Transition=0111(二進位)，索引= 7(十進位) ，查表 1-2 為逆時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= -1”

- 時序③時 Transition=1110(二進位)，索引=14(十進位)，查表 1-2 為逆時針旋轉→“-1”
- 時序④時 Transition=1000(二進位)，索引= 8(十進位) ，查表 1-2 為逆時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= -2”

- 時序⑤時 Transition=0001(二進位)，索引= 1(十進位) ，查表 1-2 為逆時針旋轉→“-1”
- 時序⑥時 Transition=0111(二進位)，索引= 7(十進位) ，查表 1-2 為逆時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= -3”

- 時序⑦時 Transition=1110(二進位)，索引=14(十進位)，查表 1-2 為逆時針旋轉→“-1”
- 時序⑧時 Transition=1000(二進位)，索引= 8(十進位) ，查表 1-2 為順時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= -4”

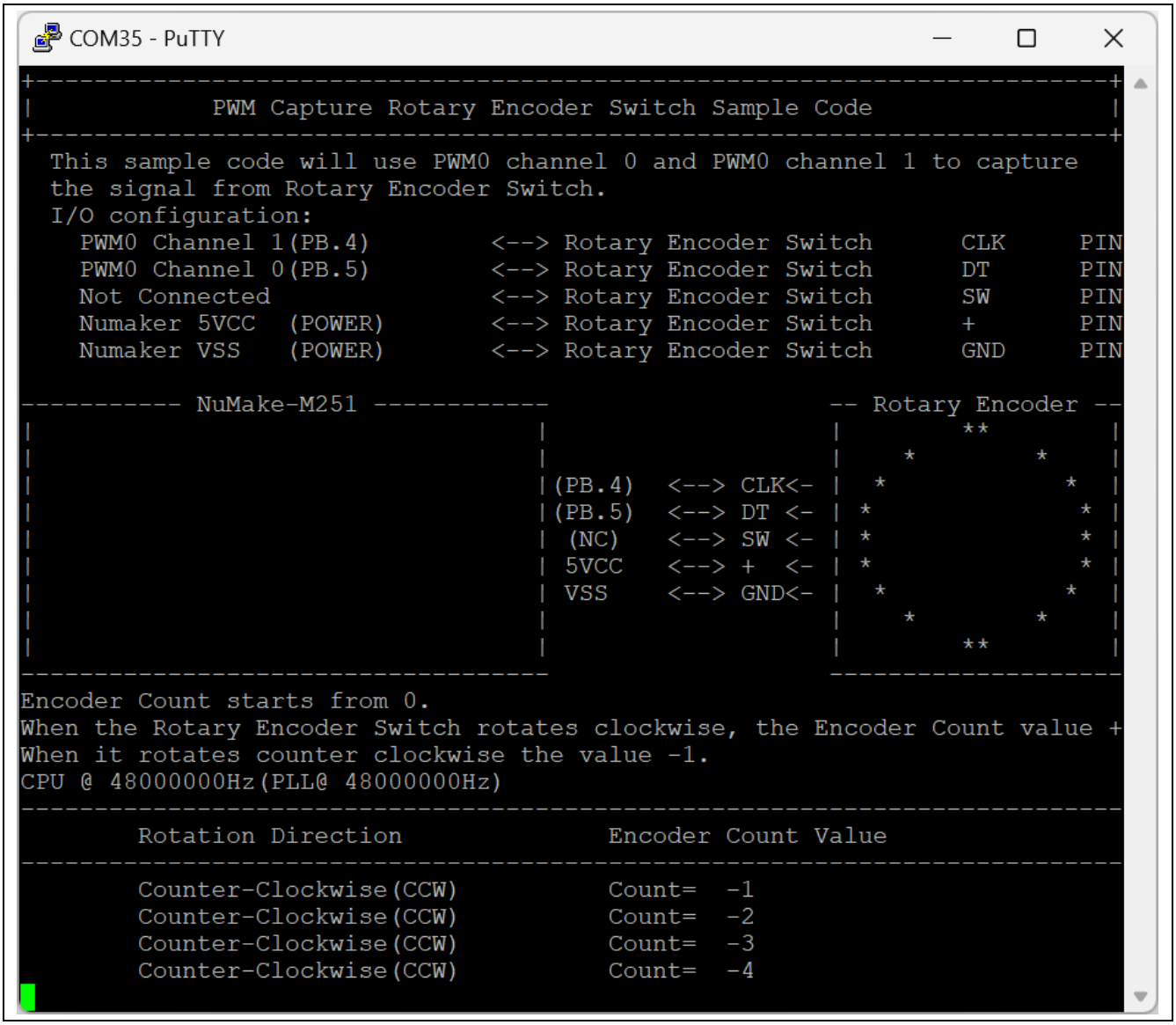

圖 1-13 逆時針旋轉執行結果

如圖 1-12 Rotary Encoder Switch 逆時針旋轉時序⓪→⑧執行結果為圖 1-13 所示。

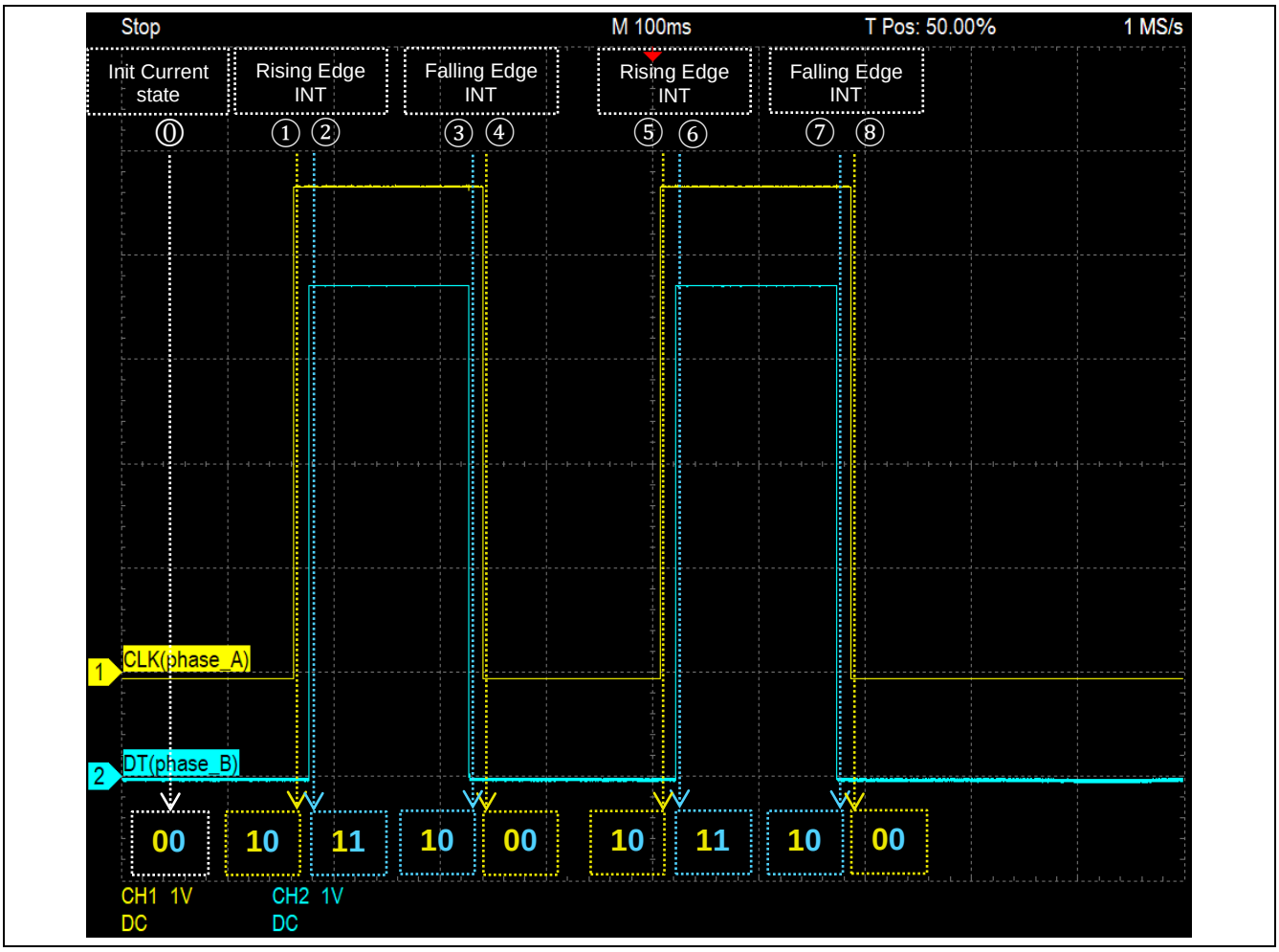

圖 1-14 Rotary Encoder Switch 順時針旋轉與逆時針旋轉交替

Current state = (phase_A << 1) | phase_B, Transition = (Last state << 2) | Current state

- 時序⓪時初始化 Current state=00(二進位)
- 時序①時 Transition=0010(二進位)，索引= 2(十進位) ，查表 1-2 為順時針旋轉→“1”
- 時序②時 Transition=1011(二進位)，索引=11(十進位)，查表 1-2 為順時針旋轉→“1”

→ 連續 2 次出現“1” → 顯示“ Clockwise(CW) Count= 1”

- 時序③時 Transition=1110(二進位)，索引=14(十進位)，查表 1-2 為逆時針旋轉→“-1”
- 時序④時 Transition=1000(二進位)，索引= 8(十進位) ，查表 1-2 為逆時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= 0”

- 時序⑤時 Transition=0010(二進位)，索引= 2(十進位) ，查表 1-2 為順時針旋轉→“1”
- 時序⑥時 Transition=1011(二進位)，索引=11(十進位)，查表 1-2 為順時針旋轉→“1”

→ 連續 2 次出現“1” → 顯示“ Clockwise(CW) Count= 1”

- 時序⑦時 Transition=1110(二進位)，索引=14(十進位)，查表 1-2 為逆時針旋轉→“-1”
- 時序⑧時 Transition=1000(二進位)，索引= 8(十進位) ，查表 1-2 為逆時針旋轉→“-1”

→ 連續 2 次出現“-1” → 顯示“ Counter-Clockwise(CCW) Count= 0”

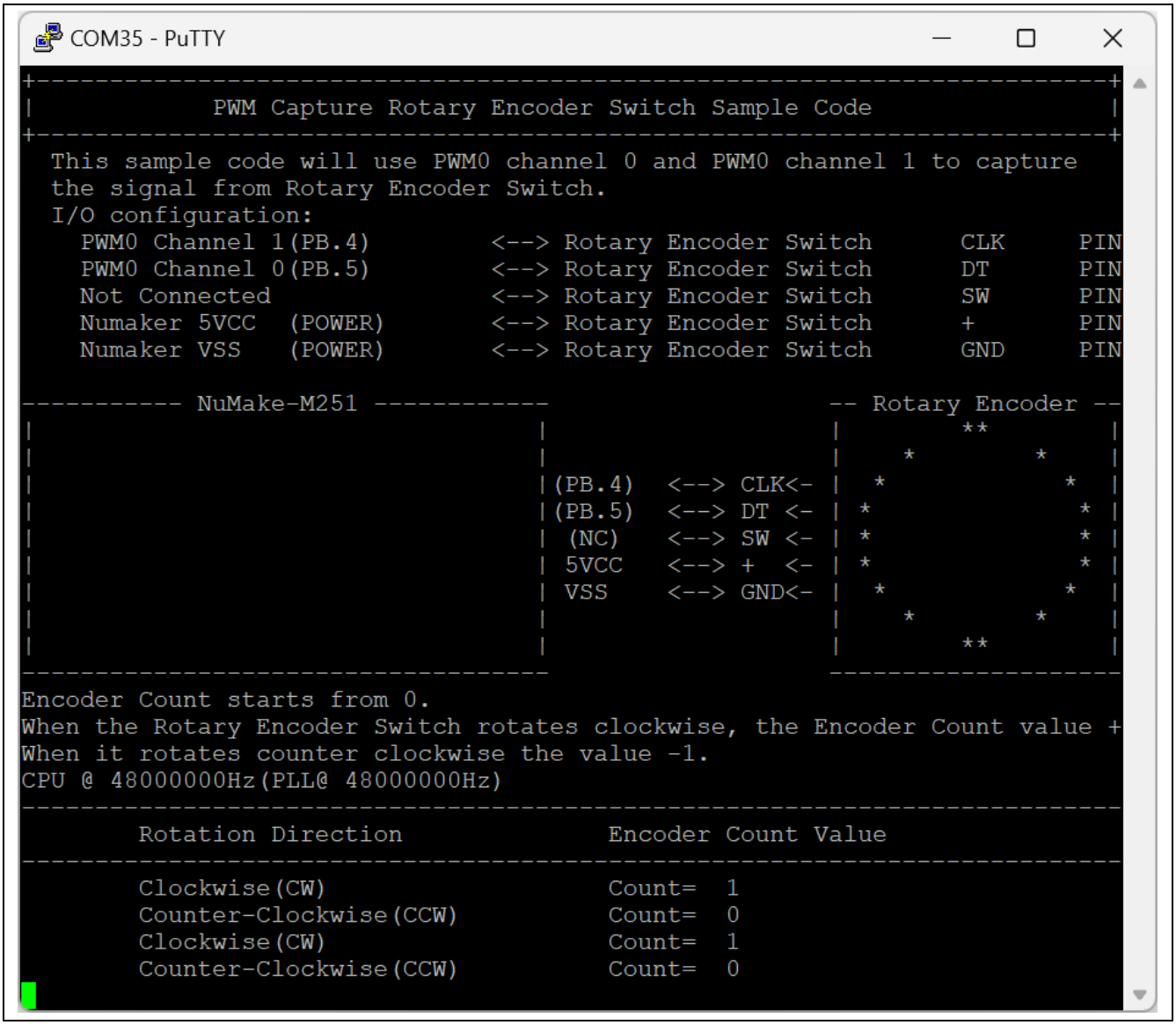

圖 1-15 順時針旋轉與逆時針旋轉交替執行結果

如圖 1-14 Rotary Encoder Switch 順時針旋轉與逆時針旋轉交替時序⓪→⑧執行結果為

圖 1-15 所示。

2. 程式碼介紹

程式碼工作流程如下：

1. main.c 中初始 M251/M252/M254/M256/M258 系列相關外設，包含 HIRC、UART0、PWM0，並在 main.c 主循環中去判斷 Rotary Enconder Switch 是否有旋轉。
2. 使用 PWM0 CH0 和 CH1 的 Capture 中斷檢測 Rotary Enconder Switch A/B 相位的上緣/下緣中斷。
3. 通過狀態表判斷方向變化，並累加變化量以濾除抖動干擾。
4. 每次穩定的增減 2 次變化才判定為一次有效旋轉，並記錄事件。
5. 使用迴圈佇列 event_queue 存儲方向和計數資訊。

在 while(1){...} 主循環中，M251/M252 會去從事件佇列中取出 1 筆 Rotary Enconder Switch 資料並顯示方向與計數值，用來即時顯示旋轉方向與對應的計數。

```
int main(void)
{
    /* Initialize system and clocks */
    SYS_Init();

      /* Initialize UART for printing */
    UART0_Init();

    /* Initialize PWM capture for encoder */
    PWM0_Capture_Init();

    EncoderEvent evt;
    printf("+-------------------------------------------------------------------------+\n");
    printf("|            PWM Capture Rotary Encoder Switch Sample Code                |\n");
    printf("+-------------------------------------------------------------------------+\n");
    ...
    printf("-------------------------------------------------------------------------\n");
    printf("\tRotation Direction\t\tEncoder Count Value\n");
    printf("-------------------------------------------------------------------------\n");
    while (1)
    {
        /* If there is an event in the queue, print it */
        if (dequeue_event(&evt))
        {
            if (evt.Direction == 1)
            {
                printf("\tClockwise(CW)\t\t\tCount=\t%d\n", evt.Count);
            }
            else if (evt.Direction == -1)
            {
                printf("\tCounter-Clockwise(CCW)\t\tCount=\t%d\n", evt.Count);
            }
        }
    }
}
```

PWM0 channel 0 and channel 1 判斷上緣觸發中斷和下緣觸發中斷，進入中斷後先初始 phases A 和 phases B 狀態，然後將當前狀態 current_state 記錄下來(2 Bits 值)，下一個中斷進來後將目前current_state和剛剛上一個中斷紀錄起來current_state結合在一起為4 Bits值(transition)，將進入中斷得到的 transition 值去查表(transition_table)便可知旋轉方向，連續 2 次中斷進入後判斷為旋轉同方向才為此旋轉方向。

```
void PWM0_P0_IRQHandler(void)
{
    /* Get capture interrupt flags for channel 0 and channel 1 */
    uint32_t flag_state_ch0 = PWM_GetCaptureIntFlag(PWM0, 0);
    uint32_t flag_state_ch1 = PWM_GetCaptureIntFlag(PWM0, 1);

    /* Check if channel 0 has a rising edge capture interrupt */
    if (flag_state_ch0 & 0x01)
    {
        /* Handle encoder event on channel 0 rising edge */
        HandleEncoderEvent();

        /* Clear channel 0 rising edge capture interrupt flag */
        PWM_ClearCaptureIntFlag(PWM0, 0, PWM_CAPTURE_INT_RISING_LATCH);
    }
    /* Check if channel 0 has a falling edge capture interrupt */
    if (flag_state_ch0 & 0x02)
    {
        /* Handle encoder event on channel 0 falling edge */
        HandleEncoderEvent();

        /* Clear channel 0 falling edge capture interrupt flag */
        PWM_ClearCaptureIntFlag(PWM0, 0, PWM_CAPTURE_INT_FALLING_LATCH);
    }
    /* Check if channel 1 has a rising edge capture interrupt */
    if (flag_state_ch1 & 0x01)
    {
        /* Handle encoder event on channel 1 rising edge */
        HandleEncoderEvent();

        /* Clear channel 1 rising edge capture interrupt flag */
        PWM_ClearCaptureIntFlag(PWM0, 1, PWM_CAPTURE_INT_RISING_LATCH);
    }
    /* Check if channel 1 has a falling edge capture interrupt */
    if (flag_state_ch1 & 0x02)
    {
        /* Handle encoder event on channel 1 falling edge */
        HandleEncoderEvent();

        /* Clear channel 1 falling edge capture interrupt flag */
        PWM_ClearCaptureIntFlag(PWM0, 1, PWM_CAPTURE_INT_FALLING_LATCH);
    }
}

void HandleEncoderEvent()
{
    /* Read current states of encoder phases A and B from pins */
    uint8_t phase_A = PB4 & 1;
    uint8_t phase_B = PB5 & 1;

    /* Encode current state */
    uint8_t current_state = (phase_A << 1) | phase_B;

     /* Form 4-bit transition key from last state and current state */
```

```
    uint8_t transition = (last_state << 2) | current_state;

    /* Lookup direction using transition table */
    int8_t dir = transition_table[transition];

    /* If movement is valid */
    if (dir != 0)
    {
      transition_sum += dir;
      if (transition_sum >= 2)
      {
        /* Increase count on clockwise rotation */
        encoder_count++;
        enqueue_event(1, encoder_count);
        transition_sum = 0;
      }
      else if (transition_sum <= -2)
      {
        /* Decrease count on counter-clockwise rotation */
        encoder_count--;
        enqueue_event(-1, encoder_count);
        transition_sum = 0;
      }
    }
    /* Update last state */
    last_state = current_state;
}
```

transition_table 為判斷 Rotary Encoder Switch 旋轉方向有效性。

```
const int8_t transition_table[16] =
{
     0,  /*  0: 00 -> 00 (no change) */
    -1,  /*  1: 00 -> 01 (CCW)       */
     1,  /*  2: 00 -> 10 (CW)        */
     0,  /*  3: 00 -> 11 (invalid)   */

     1,  /*  4: 01 -> 00 (CW)        */
     0,  /*  5: 01 -> 01 (no change) */
     0,  /*  6: 01 -> 10 (invalid)   */
    -1,  /*  7: 01 -> 11 (CCW)       */

    -1,  /*  8: 10 -> 00 (CCW)       */
     0,  /*  9: 10 -> 01 (invalid)   */
     0,  /* 10: 10 -> 10 (no change) */
     1,  /* 11: 10 -> 11 (CW)        */

     0,  /* 12: 11 -> 00 (invalid)   */
     1,  /* 13: 11 -> 01 (CW)        */
    -1,  /* 14: 11 -> 10 (CCW)       */
     0   /* 15: 11 -> 11 (no change) */
};
```

PWM0_Capture_Init 設定 PWM0 使用 PIN，如使用 PB4 和 PB5 作為偵測 Rotary Enconder Switch 產生的脈波，並初始化 PB4 和 PB5 的值。

```
void PWM0_Capture_Init(void)
{
    /* Enable PWM0 module clock */
    CLK_EnableModuleClock(PWM0_MODULE);

    /* Set PWM0 clock source to PCLK0 with no divider */
    CLK_SetModuleClock(PWM0_MODULE, CLK_CLKSEL2_PWM0SEL_PCLK0, 0);

    /* Set multi-function pins PB.4 and PB.5 for PWM0 CH1 and CH0 */
    SYS->GPB_MFPL &= ~(SYS_GPB_MFPL_PB4MFP_Msk | SYS_GPB_MFPL_PB5MFP_Msk);
    SYS->GPB_MFPL |= (SYS_GPB_MFPL_PB4MFP_PWM0_CH1 | SYS_GPB_MFPL_PB5MFP_PWM0_CH0);

    /* Configure PWM0 channel 0 as capture input with 1000 unit time and 0 prescaler */
    PWM_ConfigCaptureChannel(PWM0, 0, 1000, 0);

    /* Enable capture function and interrupts for channel 0 */
    PWM_EnableCapture(PWM0, PWM_CH_0_MASK);
    PWM_EnableCaptureInt(PWM0, 0, PWM_CAPTURE_INT_RISING_LATCH |
PWM_CAPTURE_INT_FALLING_LATCH);

    /* Configure PWM0 channel 1 as capture input with 1000 unit time and 0 prescaler */
    PWM_ConfigCaptureChannel(PWM0, 1, 1000, 0);

    /* Enable capture function and interrupts for channel 1 */
    PWM_EnableCapture(PWM0, PWM_CH_1_MASK);
    PWM_EnableCaptureInt(PWM0, 1, PWM_CAPTURE_INT_RISING_LATCH |
PWM_CAPTURE_INT_FALLING_LATCH);

    /* Enable PWM0 interrupt in NVIC */
    NVIC_EnableIRQ(PWM0_P0_IRQn);

     /* Set the interrupt priority for PWM0_P0_IRQn to level 1
     (lower number means higher priority) */
    NVIC_SetPriority(PWM0_P0_IRQn, 1);

    /* Initialize last_state to the current state of the encoder */
    last_state = (PB4 ? 1 : 0) << 1 | (PB5 ? 1 : 0);
}
```

使用 EncoderEvent struct 記錄旋轉編碼器（Rotary Enconder Switch）事件的資料，紀錄旋轉的方向和計數值。

```
typedef struct
{
    volatile int8_t Direction;              /* Direction of rotation */
    volatile int Count;                     /* Encoder count */
} EncoderEvent;
```

旋轉編碼器的事件（旋轉方向與計數值）加入到事件佇列（queue）中。

```
void enqueue_event(int8_t Direction, int Count)
{
    uint8_t next_tail = (queue_tail + 1) % QUEUE_SIZE;
    if (next_tail != queue_head)
    {
        event_queue[queue_tail].Direction = Direction;
        event_queue[queue_tail].Count = Count;
        queue_tail = next_tail;
    }
}
```

從佇列中取出 1 筆 Rotary Enconder Switch 事件資料（旋轉方向與計數值）。

```
int dequeue_event(EncoderEvent* event)
{
    if (queue_head == queue_tail)
        return 0;
    *event = event_queue[queue_head];
    queue_head = (queue_head + 1) % QUEUE_SIZE;
    return 1;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本

  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007

- IDE 版本

  - Keil uVision V5.38.0.0

3.2 硬體需求

- 電路元件

  - NuMaker-M252SD V1.1

- 線路示意圖及 Rotary Encoder 接法

  - 將 NuMaker-M252SD 的相關 PIN 與 Rotary Encoder Switch 相關 PIN 對接如下:

PWM0 Channel 1(PB.4)	↔ Rotary Encoder Switch	CLK	PIN
PWM0 Channel 0(PB.5)	↔ Rotary Encoder Switch	DT	PIN
Not Connected	↔ Rotary Encoder Switch	SW	PIN
Numaker 5VCC (POWER)	↔ Rotary Encoder Switch	+	PIN
Numaker VSS (POWER)	↔ Rotary Encoder Switch	GND	PIN

  - 將 NuMaker-M252SD 中 Nu-Link2-Me 的 USB 與 PC USB 對接，以顯示範例程式碼的執行結果。

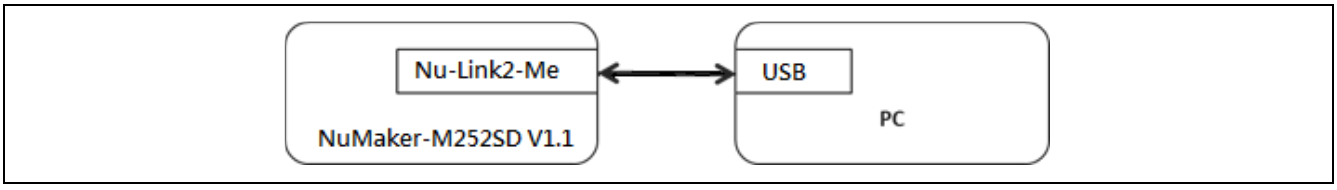

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

EC_M251_M252_PWM_Capture_Rotary_Encoder_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	
M251_M252_PWM_Capture_Rotary_Encoder	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 *M251_M252_PWM_Capture_Rotary_Encoder.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

6. 修訂紀錄

Date	Revision	Description
2025.06.16	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.