RTX下使用ADC，GPIO和USB Mouse

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

代碼簡述	在 RTX 程式下執行 ADC，GPIO 和 USB Mouse。
BSP 版本	NUC230_240 Series BSP CMSIS V3.01.001
開發平台	NuEdu-EVB-NUC240 V2.1 NuEdu-Basic01 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 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 功能介紹

1.1 簡介

當微控制器程式需同時執行許多任務時，可使用實時操作系統(real-time operating system，RTOS)來管理這些任務。RTOS將CPU的處理時間切割成許多時隙 (time slots)，每個時隙進行不同的任務，並且在每個時隙之間進行任務的切換。因為任務間的切換在一秒鐘內可以達到上百次，整體看起來像是這些任務是同時在執行。當程式比較複雜需要許多任務同時執行，就可使用RTOS協助建立和管理這些任務。

Keil MDK (Microcontroller Development Kit) 內建的CMSIS-RTOS，為底層 RTOS和上層應用程式提供一套標準API，包括Keil免費的RTX (Real-Time eXecutive) 核心。當其他RTOS也支援CMSIS-RTOS，透過共同標準的API，則可輕易將程式移植到其他RTOS平台。本範例代碼利用Keil裡提供的CMSIS-RTOS建立RTX核心的系統，然後執行USB、ADC、UART和GPIO任務。

1.2 原理

在RTX環境下，建立以下4個任務，並且同時執行：

- USB

  用 4 個按鍵 SW1~SW4 作為輸入，模擬控制滑鼠的移動方向(上、下、左或右)。

- ADC

  ADC 進行電壓轉換，旋轉可變電阻的旋鈕可改變輸入的電壓值。

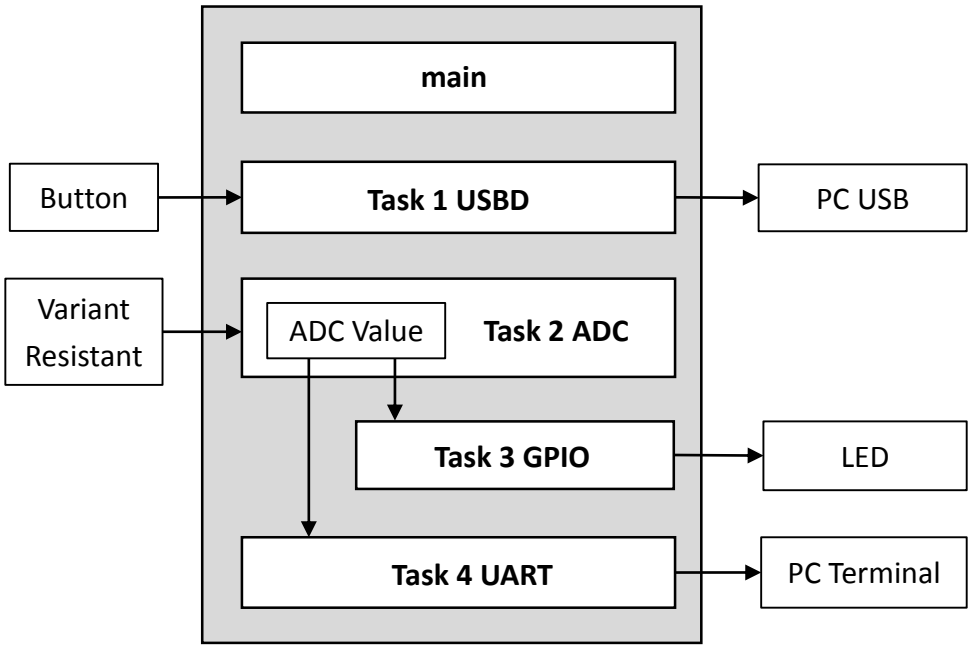
- UART

  完成一次 ADC 電壓轉換，則打印出 ADC 的電壓轉換值。

- GPIO

  根據當時的 ADC 電壓轉換值，當作 LED 閃爍頻率的輸入。

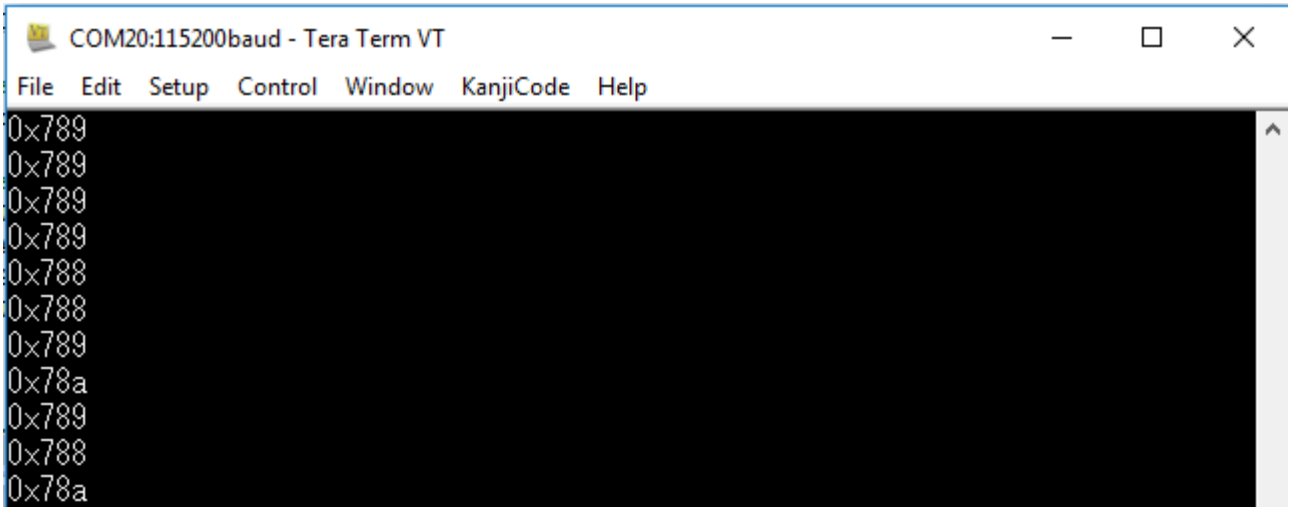
下圖描述這4個同時執行的任務所需要的輸入、輸出周邊和任務間的互動關係。



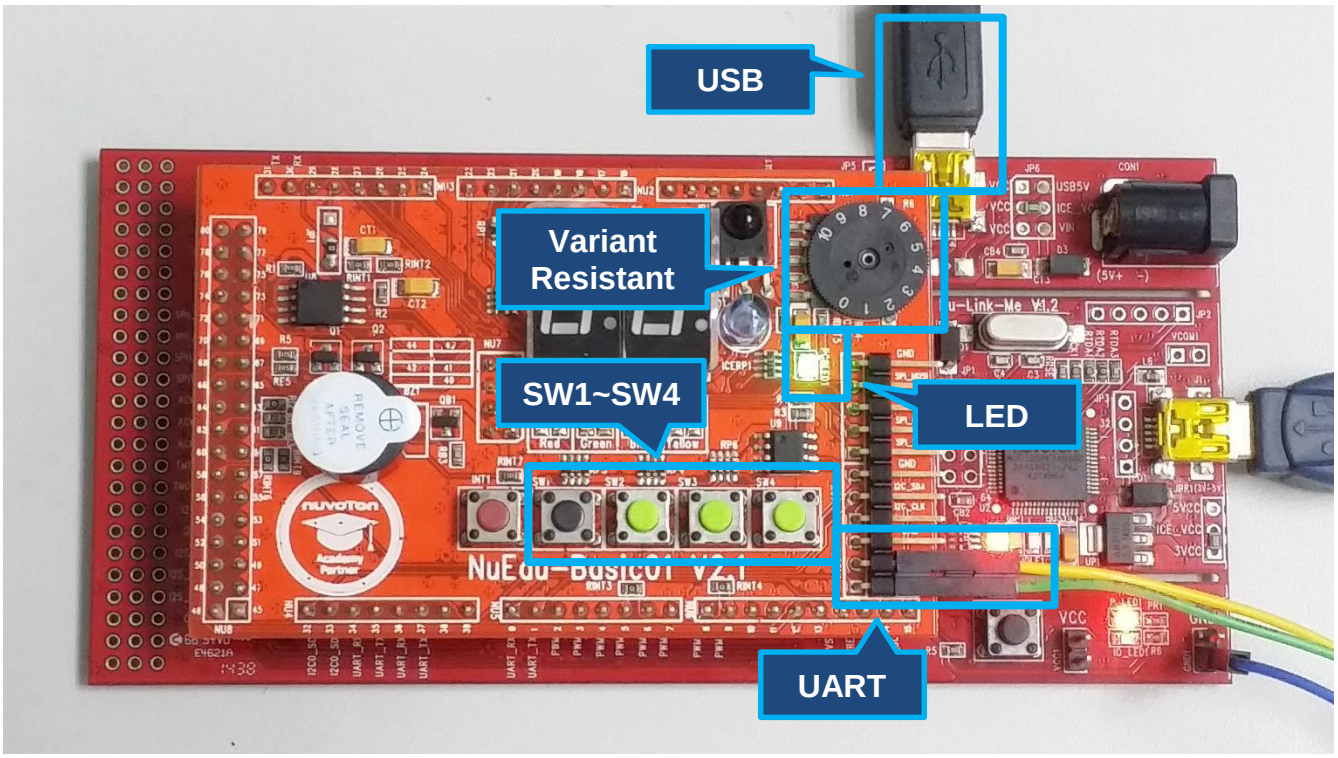
1.3 執行結果

將NUC240開發板的USB連接至電腦，按下開發版上的按鈕SW1~SW4，可控制滑鼠的上、下、左和往右移動。ADC持續對輸入電壓進行電壓轉換，轉換出來的電壓值做為控制LED閃爍頻率的參數，旋轉開發版上的可變電阻旋鈕，改變ADC的輸入電壓值，可看到LED的閃爍速度改變。連接UART至NuBridge，開啟電腦終端機軟體，UART打印ADC的電壓轉換值。

電腦終端機顯示UART打印訊息：



NUC240開發版上所用到的周邊和位置：



2 代碼介紹

2.1 RTX

Thread宣告

本範例代碼有4個平行處理的任務，分別宣告其函式原型，並為每個任務宣告各自的ID，ID的資料格式為osThreadId。

```
/* Thread IDs */
osThreadId t_task1_USB;     // Declare a thread ID for task 1 (USBD mouse)
osThreadId t_task2_ADC;     // Declare a thread ID for task 2 (ADC conversion)
osThreadId t_task3_GPIO;    // Declare a thread ID for task 3 (GPIO LED control)
osThreadId t_task4_UART;    // Declare a thread ID for task 4 (UART print message)

/* Function declaration */
void task1_USB(void const *argument);
void task2_ADC(void const *argument);
void task3_GPIO(void const *argument);
void task4_UART(void const *argument);
```

使用osThreadDef macro宣告每個任務為thread，也可以用來設定任務的優先等級。

```
/* define task as thread function */
osThreadDef(task1_USB, osPriorityNormal, 1, 0);
osThreadDef(task2_ADC, osPriorityNormal, 1, 0);
osThreadDef(task3_UART, osPriorityNormal, 1, 0);
osThreadDef(task4_GPIO, osPriorityNormal, 1, 0);
```

Semaphore宣告

宣告作為任務間內部互相溝通的Semaphore，名字為Sem_AdcDone，以及宣告其ID變數。

```
/* Declare Semaphore */
osSemaphoreDef(Sem_AdcDone);

/* Semaphore ID */
osSemaphoreId Sem_AdcDone_id;
```

OS Kernel 初始化

呼叫osKernelInitialize函式，開始進行OS Kernel初始化。

```
    /* Initialize OS Kernel */
    printf("RTX Start ...\n");
    osKernelInitialize();
```

Semaphore建立

呼叫osSemaphoreCreate函式建立新的Semaphore，並將傳回值指定給該Semaphore的ID。一開始沒有任何token，當ADC完成一次電壓轉換，會釋出一個token，UART則等ADC轉換完成後，打印電壓轉換結果。

```
    /* Create Semaphore without token */
    Sem_AdcDone_id = osSemaphoreCreate(osSemaphore(Sem_AdcDone), 0);
```

Thread建立

用osThreadCreate函式為每個任務建立新的thread，傳回值指定給該任務的ID。

```
    /* Create task and assign thread ID to t_task */
    t_task1_USB = osThreadCreate(osThread(task1_USB), NULL);
    t_task2_ADC = osThreadCreate(osThread(task2_ADC), NULL);
    t_task3_GPIO = osThreadCreate(osThread(task3_GPIO), NULL);
    t_task4_UART = osThreadCreate(osThread(task4_ UART), NULL);
```

開始執行OS Kernel

呼叫osKernelStart函式開始執行OS Kernel以及任務切換。

```
    /* Start the OS kernel and begin thread switching */
    osKernelStart();
```

2.2 USB

初始化按鍵，將相對應的GPIO (PC.13、PE.8、PC14和PC.15) 設定為input mode。

```
    /* Initial GPIO for key input */
    Initial_Key_Input();
```

在USB任務裡，當End Point 2事件發生，檢查目前是否有按鍵被按下，將資料寫入USB SRAM。資料傳回電腦USB控制滑鼠的移動。

```
void task1_USB(void const *argument)
{
    while(1) {

        uint8_t *buf;

        if(g_u8EP2Ready) {
            buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));

            /* Update new report data */
            buf[0] = 0x00;
            buf[1] = 0x00;
            buf[2] = 0x00;
            buf[3] = 0x00;

            if(Get_Key_Input() & 0x01)
                buf[1] = 16;
            if(Get_Key_Input() & 0x02)
                buf[1] = (uint8_t)-16;
            if(Get_Key_Input() & 0x04)
                buf[2] = 16;
            if(Get_Key_Input() & 0x08)
                buf[2] = (uint8_t)-16;

            g_u8EP2Ready = 0;

            /* Set transfer length and trigger IN transfer */
            USBD_SET_PAYLOAD_LEN(EP2, 4);
        }
    }

}
```

2.3 ADC

初始化ADC，設定ADC時鐘，多功能管腳，操作模式和開啟電源。

```
    /* Open Volume Knob Device */
    Open_ADC_Knob();
```

在ADC任務裡，延遲10ms後做一次ADC電壓轉換，得到的數值存到g_u32AdcValue，然後釋出一個semaphore token。

```
void task2_ADC(void const *argument)
{
    while(1) {
        osDelay(10);                            /* delay 10 msec */
        g_u32AdcValue = Get_ADC_Knob();         /* Get Volume Knob data(0~4095) */
        osSemaphoreRelease(Sem_AdcDone_id);     /* Increase semaphore token */
    }
}
```

2.4 GPIO

RGB LED分別連接到GPIO (PA.12、PA.13和PA.14)。ADC轉換出來的電壓值g_u32AdcValue作為LED閃爍頻率的輸入，當轉換出來的數值越大，表示LED的閃爍頻率越快，延遲的時間越短，每次輪流點亮一個LED。

```
void Write_LED_Color_Flash(uint32_t u32Speed)
{
    uint32_t i;
    static uint32_t u32LED_Count=0;
    volatile uint32_t *p32LED[LED_Color_Count] = {&LED_B, &LED_R, &LED_G};

    /* LED Delay Time Control */
    if(u32Speed>4096)  u32Speed = 4096;
    for(i=0; i<((4096-u32Speed)*500); i++);

    for(i=0; i<LED_Color_Count; i++) {
        if(u32LED_Count==i)
            *p32LED[i] = 0; //LED ON
        else
```

```
            *p32LED[i] = 1; //LED OFF
    }

    u32LED_Count++;
    if(u32LED_Count==LED_Color_Count)
        u32LED_Count = 0;
}

void task3_GPIO(void const *argument)
{
    while(1) {
        Write_LED_Color_Flash(g_u32AdcValue);
    }
}
```

2.5 UART

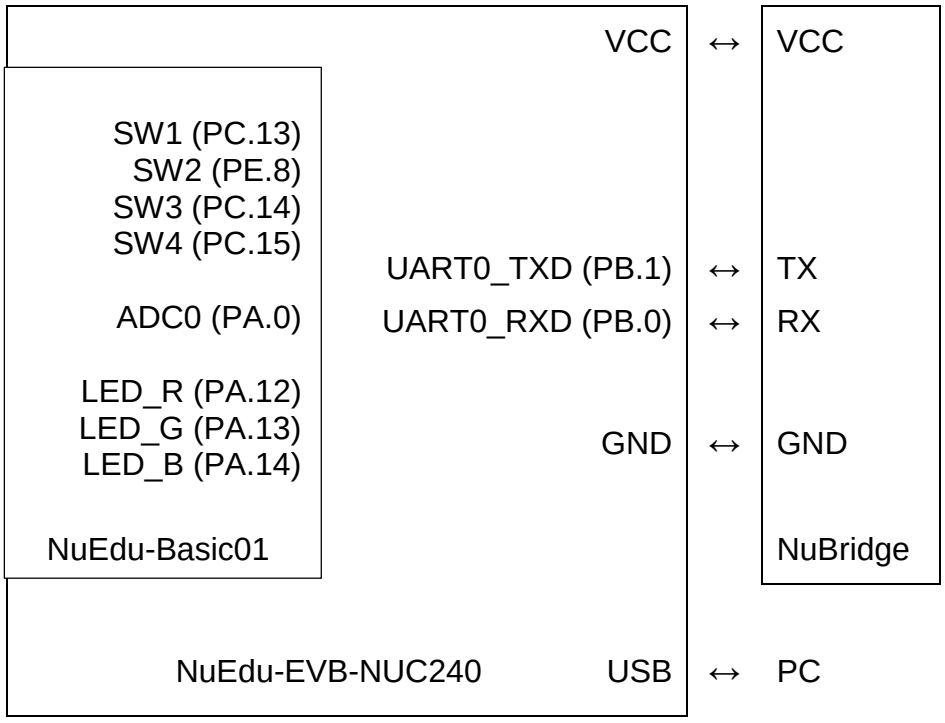
UART任務，等每一次ADC電壓轉換完成，釋出一個token，打印轉換出來的ADC電壓值。

```
void task4_UART(void const *argument)
{
    while(1) {
        printf("0x%x\n\r", g_u32AdcValue);
        osSemaphoreWait(Sem_AdcDone_id, osWaitForever);     /* Wait for semaphore token */
    }
}
```

3 軟體與硬體環境

- **軟體環境**
  - BSP 版本
    - NUC230_240 Series BSP CMSIS V3.01.001
  - IDE 版本
    - Keil uVersion 5.28
- **硬體環境**
  - 電路元件
    - NuEdu-EVB-NUC240 V2.1
    - NuEdu-Basic01 V2.1
  - 示意圖

NuEdu-EVB-NUC240 開發版上接 NuEdu-Basic01 版子、NuBridge 和 USB。



4 目錄資訊

- EC_NUC240_RTX_ADC_GPIO_USB_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
    - NuEdu — Library for NuEdu-SDK-NUC240 board
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 如何執行範例程式

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 NUC240_RTX_ADC_GPIO_USB.uvprojx。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

6 修訂紀錄

Date	Revision	Description
Aug. 31, 2019	1.00	1. 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.