<u>การทดลองที่ 5</u> การใช้งาน ADC

วัตถุประสงค์

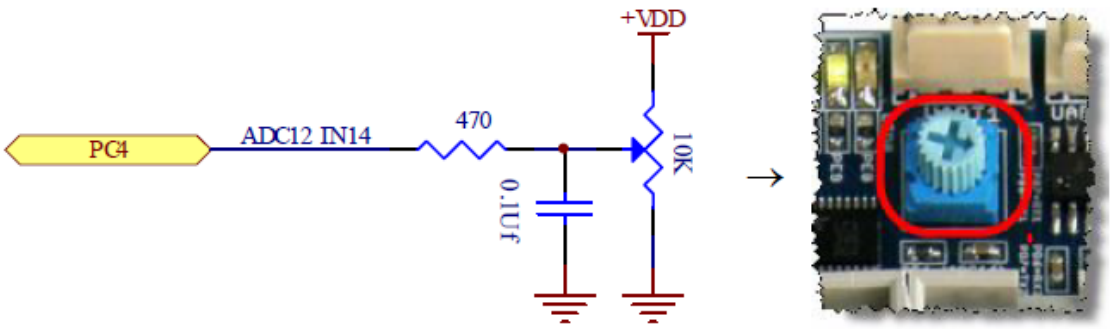
1) เข้าใจการทำงานของ Analog to Digital Converter
2) สามารถเขียนโปรแกรมควบคุมการทำงานของ Analog to Digital Converter

1. Analog to Digital Converter (ADC)

STM32F107xx มี ADC 2 โมดูล ได้แก่ ADC1 และ ADC2 เชื่อมต่อกับบัส APB2 รองรับสัญญาณนาฬิกาสูงสุด 14 MHz แปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิตอลด้วยวิธี successive approximation โดยมีความละเอียดในการแปลง 12 บิต โมดูล ADC แต่ละโมดูลมีช่องสัญญาณ 18 ช่อง แบ่งเป็นมัลติเพล็กซ์ 16 ช่อง ส่วนอีก 2 ช่องคือตัววัดอุณหภูมิและแหล่งจ่ายแรงดันอ้างอิงภายในไมโครคอนโทรลเลอร์ แรงดันสัญญาณอ้างอิงสูงสุดคือ 3.6 V

การใช้งานวงจรปรับแรงดัน (0-3V3)

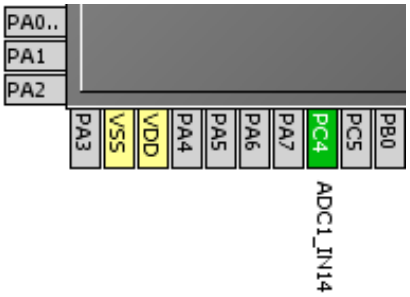
วงจรปรับแรงดันจะใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าชนิดมีแกนหมุนสำหรับปรับค่า โดยวงจรนี้ใช้กับแหล่งจ่าย +3.3V โดยจะให้ Output เป็นแรงดันซึ่งมีค่าระหว่าง 0V ถึง +3.3V ตามการปรับค่าของตัวต้านทาน โดยแรงดันไฟฟ้าที่ได้จะถูกป้อนให้กับขาสัญญาณ PC4 (ADC12_IN14) เพื่อใช้เป็นแรงดันอินพุตสำหรับทดสอบการทำงานของวงจร ADC (PC4) ดังรูปที่ 1.1



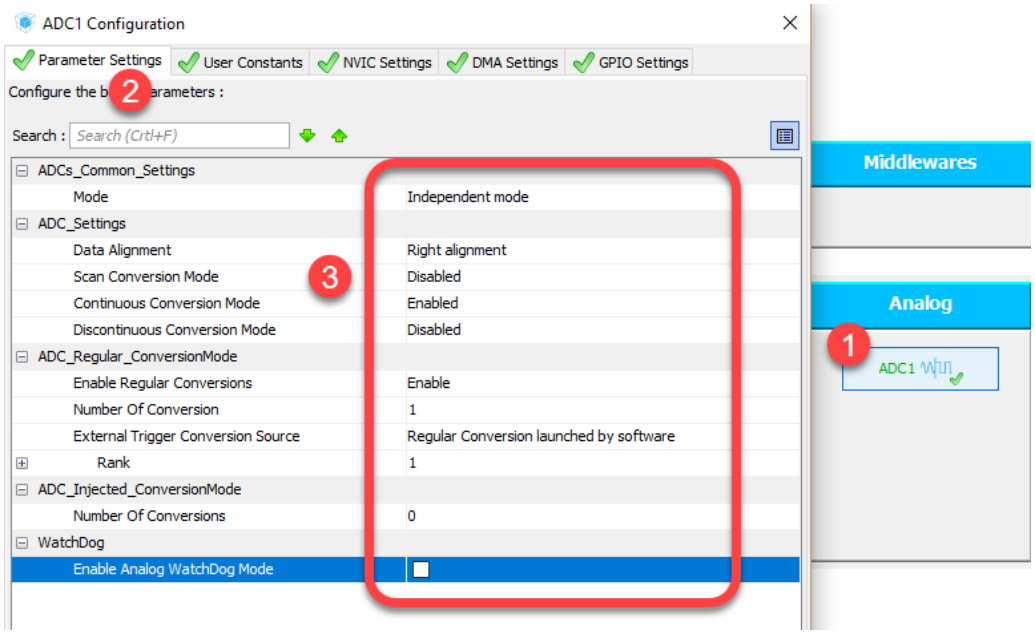
รูปที่ 1.1 วงจรปรับแรงดันไฟฟ้าเชื่อมต่อเข้ากับขา PC4 หรือ ADC12_IN14

2. การตั้งค่าในโปรแกรม STM32CubeMX

การตั้งค่าสำหรับการทดลองครั้งนี้ต้องกำหนดให้ขา PC4 ทำหน้าที่เป็น ADC ดังรูปที่ 2.1 เนื่องจากขา PC4 เชื่อมต่ออยู่กับวงจรปรับแรงดันไฟฟ้า และตั้งค่า ADC ตามรูปที่ 2.2



รูปที่ 2.1 แสดงการตั้งค่าให้ PC4 ให้ทำหน้าที่ ADC



รูปที่ 2.2 แสดงการตั้งค่าให้ ADC

3. อธิบายการทำงานของ ADC

โปรแกรม STM32CubeMX ตั้งค่า ADC ด้วยฟังก์ชัน `MX_ADC1_Init` ในไฟล์ `main.c` ดังรูปที่ 3.1 ซึ่งกำหนดการทำงานของ ADC ให้ทำงานแบบ Regular จัดเรียงข้อมูลชิดขวา (`ADC_DATAALIGH_RIGHT`) แล้วจ่ายสัญญาณนาฬิกาให้กับ GPIO พอร์ต C ในฟังก์ชัน `MX_GPIO_Init` ดังรูปที่ 3.2 ส่วนการกำหนดให้ขา PC4 ทำหน้าที่เป็น Analog Input อยู่ในฟังก์ชัน `HAL_ADC_MspInit` ในไฟล์ `stm32f1xx_hal_msp.c` ดังรูปที่ 3.3

```
/* ADC1 init function */
void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

    /**Common config
    */
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  HAL_ADC_Init(&hadc1);

    /**Configure Regular Channel
    */
  sConfig.Channel = ADC_CHANNEL_14;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

}
```

รูปที่ 3.1 แสดงการตั้งค่า ADC ในฟังก์ชัน `MX_ADC1_Init()`

```
void MX_GPIO_Init(void)
{

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();

}
```

รูปที่ 3.2 แสดงการจ่ายสัญญาณนาฬิกาให้ GPIOC

```
/* ADC1 init function */
void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

    /**Common config
    */
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  HAL_ADC_Init(&hadc1);

    /**Configure Regular Channel
    */
  sConfig.Channel = ADC_CHANNEL_14;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

}
```

รูปที่ 3.3 แสดงตั้งค่าให้ PC4 ทำหน้าที่รับสัญญาณอินพุตแบบแอนะล็อก

4. การอ่านค่าที่ได้จากการแปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิตอล

ก่อนการใช้งาน ADC จะต้องทำการ calibrate และสั่งให้ ADC เริ่มต้นแปลงสัญญาณเป็นสัญญาณดิจิตอลด้วยฟังก์ชัน `HAL_ADCEx_Calibration_Start` และฟังก์ชัน `HAL_ADC_Start` ตามลำดับ

ฟังก์ชัน `HAL_ADC_GetValue` ใช้สำหรับการอ่านค่าดิจิตอลที่โมดูล ADC แปลงได้ โดยจะส่งค่าที่แปลงได้เป็นตัวเลขจำนวนเต็มไม่มีเครื่องหมายขนาด 32 บิต หรือ `uint32_t` ดังรูปที่ 4.1

HAL_ADC_GetValue

Function Name	**uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)**
Function Description	Get ADC regular group conversion result.
Parameters	• **hadc:** ADC handle
Return values	• Converted value
Notes	• Reading DR register automatically clears EOC (end of conversion of regular group) flag.

รูปที่ 4.1 แสดงรายละเอียดของฟังก์ชัน `HAL_ADC_GetValue`

ก่อนการเรียกใช้ฟังก์ชัน `HAL_ADC_GetValue` ต้องตรวจสอบว่าโมดูล ADC ได้แปลงสัญญาณเสร็จสิ้นแล้ว ด้วยการเรียกฟังก์ชัน `HAL_ADC_PollForConversion` ซึ่งจะส่งค่าสถานะ `HAL_OK` กลับมาหากโมดูล ADC ทำการแปลงสัญญาณเสร็จเรียบร้อยแล้ว แสดงตัวอย่างการอ่านค่าจากโมดูล ADC ได้ดังรูปที่ 4.2 โดยค่า 100 เป็นค่า Timeout ของฟังก์ชัน

```
volatile uint32_t adc_val;

HAL_ADCEx_Calibration_Start(&hadc1);
HAL_ADC_Start(&hadc1);

while (1){
  while ( HAL_ADC_PollForConversion(&hadc1, 100) != HAL_OK ){}
  adc_val = HAL_ADC_GetValue(&hadc1);
}
```

รูปที่ 4.2 แสดงวิธีการอ่านค่าจากโมดูล ADC

5. การทดลอง

1. แสดงตัวเลขในรูปเลขฐาน 16 ทาง UART2

จงสร้างฟังก์ชัน `displayHEX` ขึ้นมา โดยมี Function Prototype ดังนี้

```
void displayHEX(uint16_t);
```

เพื่อแปลงเลขจำนวนเต็มขนาด 16 บิตที่รับเข้ามาแล้วแสดงผลออกทาง UART2 ในรูปเลขฐาน 16 จำนวน 4 หลัก ดังตัวอย่างต่อไปนี้

```
uint16_t hex1 = 501;
displayHEX(hex1);
```

โปรแกรมจะแสดง **0x01F5** ออกมาทาง UART2

2. การอ่านค่าที่แปลงได้จากโมดูล ADC

ใช้โปรแกรม STM32CubeMX สร้างโปรเจ็คขึ้นมา โดยตั้งค่าให้ขา PC4 ดังรูปที่ 2.1 - รูปที่ 2.2 เพื่อให้ทำหน้าที่รับสัญญาณอินพุตแบบแอนะล็อก จากนั้นให้เขียนโปรแกรมตามรูปที่ 4.2 เพื่ออ่านค่าผลการแปลงสัญญาณจากโมดูล ADC ที่ขา PC4 แล้วแสดงค่าที่แปลงได้ในโปรแกรม Tera Term ผ่าน UART2 ด้วยฟังก์ชัน `displayHEX()` ที่สร้างจากการทดลองที่ 1 กำหนดให้แสดงผลทุกๆ 300 ms ให้ทดลองหมุนปรับตัวต้านทานปรับค่าได้ในวงจรปรับแรงดันไฟฟ้า สังเกตและบันทึก

ค่าที่น้อยที่สุดที่แปลงได้ คือ .....

ค่าที่มากที่สุดที่แปลงได้ คือ .....

ทำไมค่าที่แปลงได้สูงสุดจึงไม่ใช่ 0xFFFF

.....

.....

.....

.....

.....

.....

.....

3. การแสดงผลที่ได้จากโมดูล ADC เป็นช่วงๆ ด้วย LED

จงเขียนโปรแกรมเพื่อแสดงระดับของสัญญาณที่ได้จาก ADC ออกทาง LED บนบอร์ดทดลอง โดยให้แบ่งระดับสัญญาณที่เป็นไปได้ออกเป็น 9 ระดับ เมื่อสัญญาณอยู่ระดับใดก็ให้ LED ติดดังตารางที่ 5.1 และให้ส่งค่าที่แปลงได้ออกทางพอร์ต UART ดังเช่นในการทดลองที่ 2

ตารางที่ 5.1 แสดงรายละเอียดของ NVIC_PriorityGroup แต่ละกลุ่ม

ระดับ	ผล
1	ไม่มี LED ติด
2	LED0 ติด
3	LED0 LED1 ติด
4	LED0 LED1 LED2 ติด
5	LED0 LED1 LED2 LED3 ติด
6	LED0 LED1 LED2 LED3 LED4 ติด
7	LED0 LED1 LED2 LED3 LED4 LED5 ติด
8	LED0 LED1 LED2 LED3 LED4 LED5 LED6 ติด
9	LED0 LED1 LED2 LED3 LED4 LED5 LED6 LED7 ติด

6. การทดลองพิเศษ (เลือก 1 ข้อ โดยให้อาจารย์หรือ TA ปี 4 เป็นผู้ตรวจเท่านั้น)

1. จงสร้างโปรแกรมเครื่องคิดเลขอย่างง่ายที่คำนวณในรูปของเลขฐาน 2 ขนาด 4 บิต กำหนดให้

- LED0 - LED3 เป็นตัวตั้งกระทำ โดยมี LED0 เป็น MSB ของตัวตั้งกระทำ
- LED4 - LED8 เป็นตัวถูกกระทำ โดยมี LED4 เป็น MSB ของตัวถูกกระทำ
- Joy Switch ทุกทิศทางเป็นตัวดำเนินการ โดยตัวดำเนินการนั้นมีด้วยการอยู่ 5 ชนิด ดังนี้

Joy Switch Direction	การทำงานของ LED
UP	การคูณ (*)
Down	การหาร (/)
Left	การลบ (-)
Right	การบวก (+)
Center	การหารเอาเศษ (%)

การทำงานของโปรแกรมมีอยู่ว่าเมื่อ Switch Wakeup ถูกกดลงจะเป็นการเตรียมกำหนดค่าของตัวตั้งกระทำ แต่ถ้า Switch Tamper ถูกกดลงก็จะเป็นการเตรียมกำหนดค่าตัวถูกกระทำ และหลังจากกดปุ่มใดปุ่มหนึ่งแล้วก็ให้ทำการกำหนดค่าโดยใช้ตัวต้านทานปรับค่าได้แบบเกือกม้า ชนิดมีแกนหมุนสำหรับปรับค่าได้ และหลังจากกำหนดค่าให้แก่ตัวตั้งกระทำและตัวถูกกระทำเรียบร้อยแล้ว ก็ให้ทำการโยก Joy Switch ในแต่ละทิศทางเพื่อหาผลลัพธ์จากกการกระทำระหว่างตัวตั้งกระทำและตัวถูกกระทำ ซึ่งเมื่อทำการโยกแล้วก็ให้ส่งผลลัพธ์ที่ได้ออกทางพอร์ต UART ออกทางหน้าจอ โดยผลลัพธ์ที่แสดงออกหน้าจอนั้นให้มีการแสดงผลลัพธ์ในรูปแบบของเลขฐาน 2 ฐาน 10 และฐาน 16

ตัวอย่างของโปรแกรม

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7

หมายความว่า

- ตัวตั้งกระทำ (LED0 - LED3) มีค่าเป็น 1010_2 หรือ 10
- ตัวถูกกระทำ (LED4 - LED7) มีค่าเป็น 1101_2 หรือ 13

ผลลัพธ์ที่ได้จะแสดงออกทางหน้าจอดังนี้ (สมมติว่าโยก Joy Switch ที่เป็นการบวก)

```
First Operand: 1010
Operand: +
Second Operand: 1110
Result:
        Binary: 10111
        Decimal: 23
        Hexadecimal: 17
```

2. จงสร้างโปรแกรมแสดงการปรับเพิ่มลดระดับสัญญาณอย่างง่าย โดยแบ่งระดับสัญญาณออกเป็น 20 ระดับ เพื่อแสดงผลทาง LED และ UART กำหนดให้

- ใช้ LED0 ถึง LED4 แสดงระดับปัจจุบันในรูปเลขฐานสอง โดยให้ LED0 เป็น MSB และ LED4 เป็น LSB
- เมื่อมีการปรับตัวต้านทานปรับค่าได้ให้มีการแสดงขั้นปัจจุบันของสัญญาณ พร้อมทั้งให้มีการส่งส่งผลลัพธ์ที่ได้ออกทางพอร์ต UART แล้วแสดงผลทางหน้าจอ

ตัวอย่างของโปรแกรม

LED0	LED1	LED2	LED3	LED4

ระดับปัจจุบัน คือ 19

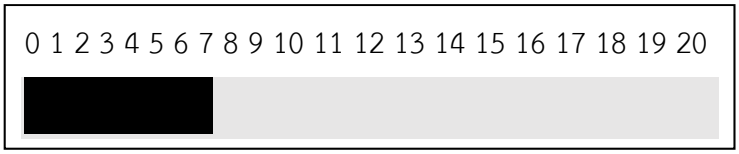
ผลลัพธ์ที่ได้จะแสดงออกทางหน้าจอดังนี้

เมื่อมีการปรับตัวต้านทานอีกครั้ง

LED0	LED1	LED2	LED3	LED4

ระดับปัจจุบัน คือ 7

ผลลัพธ์ที่ได้จะแสดงออกทางหน้าจอดังนี้



การแสดงระดับสัญญาณทาง UART ให้ใช้ตัวอักษรที่มีรหัส ASCII 178 และ 219 ตัวอักษรทั้งสองตัวมีค่ารหัสเกิน 127 ส่งผลให้แสดงผลแตกต่างกันในแต่ละโปรแกรม ดังนั้นหากใช้โปรแกรมอื่นแทนโปรแกรม Hyper Terminal ควรตรวจสอบว่าโปรแกรมที่ใช้นั้นแสดงผลตัวอักษรทั้งสองตัวนี้ได้ถูกต้องตามตัวอย่างหรือไม่

ใบตรวจการทดลองที่ 5

วัน/เดือน/ปี _____ ☐ **Sec 1** ☐ **Sec 2 กลุ่มที่** _____

1. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

2. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

3. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลายเซ็นผู้ตรวจ

การทดลองข้อ 2 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ ☐ W ☐ W+1

การทดลองข้อ 3 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ ☐ W ☐ W+1

คำถามท้ายการทดลอง

1. หากต้องแปลงสัญญาณ Analog ที่ channel 15 (ADC12_IN15) ต้องเชื่อมสัญญาณเข้ามาที่ขาใด

2. หากเปลี่ยน Data Alignment ในรูปที่ 2.2 เป็น Left Alignment จะส่งผลอย่างไรต่อโปรแกรม แล้วต้องเรียกใช้งานฟังก์ชัน `displayHEX` อย่างไรเพื่อให้แสดงผลลัพธ์ได้แบบเดียวกับการทดลองที่ 2