<u>การทดลองที่ 2</u> การใช้งาน GPIO

วัตถุประสงค์

1 เพื่อให้นักศึกษาเขียนโปรแกรมควบคุมการทำงานของ GPIO ได้

2 เพื่อให้นักศึกษาเขียนโปรแกรมเพื่อรับอินพุตจากสวิตช์บนบอร์ดได้

1. Switches

บนบอร์ด ET-STM32F ARM KIT มีสวิตช์อยู่ 3 แบบ ได้แก่ Wakeup, Tamper และ Joy Switch แบบ 5 ทิศทาง สวิตช์ Wakeup จะอยู่ใกล้ UART ส่วนสวิตช์ Tamper จะอยู่ใกล้ปุ่ม Reset โดยสวิตช์แต่จะแบบจะเชื่อมต่อกับ GPIO ที่แตกต่างกัน และเมื่อกดสวิตช์จะได้ลอจิกที่แตกต่างกัน ดังตารางที่ 1.1

ตารางที่ 1.1 รายละเอียดของสวิตช์บนบอร์ทดลอง

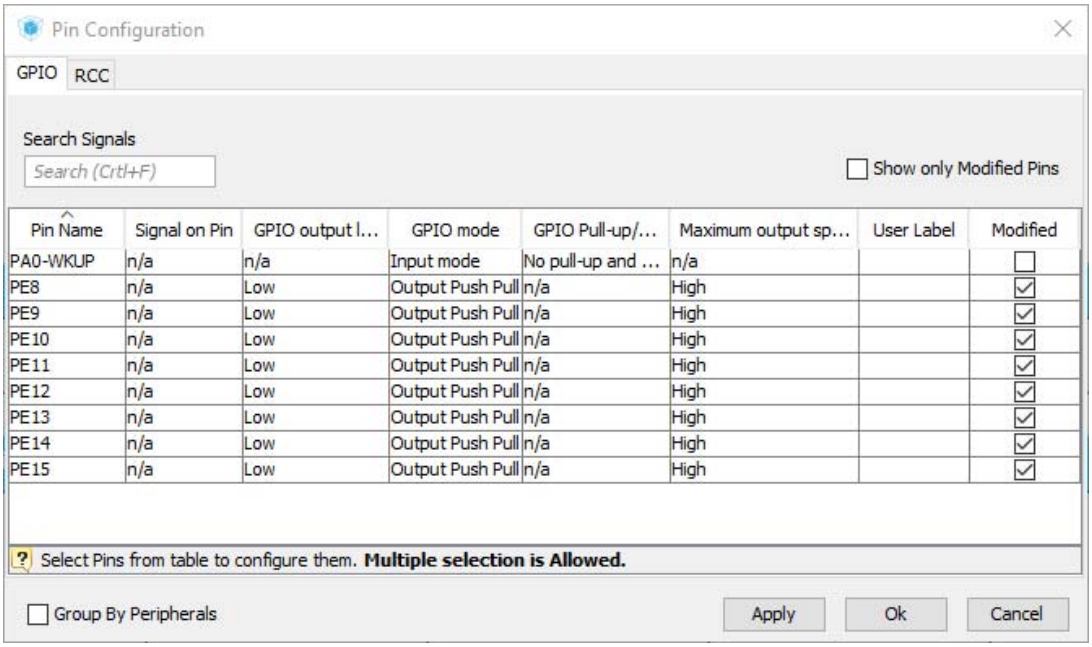
ประเภทสวิตช์	พอร์ต	สถานะลอจิก	
		เมื่อกด	เมื่อปล่อย
Wakeup	PA0	HI	LO
Tamper	PC13	LO	HI
Joy – Up	PD11		
Joy – Left	PD12		
Joy – Down	PD13		
Joy – Right	PD14		
Joy - Center	PD15		

2. การอ่านค่าอินพุต

การเริ่มต้นใช้งาน GPIO นั้นต้องมีการจ่ายสัญญาณนาฬิกาไปยังพอร์ตที่ต้องการใช้งาน พร้อมทั้งกำหนดโหมดการทำงานให้กับแต่ละขาว่าจะให้เป็นอินพุตหรือเอาต์พุตประเภทใด อีกทั้งยังสามารถกำหนดความเร็ว (speed) เมื่อทำหน้าที่เป็นเอาต์พุตได้ว่าจะให้มีความเร็ว LOW, MEDIUM หรือ HIGH ซึ่งจะมีความเร็ว 2, 10 หรือ 50 MHz ตามลำดับ

หากต้องการเขียนโปรแกรมเพื่อตรวจสอบว่าเมื่อสวิตช์ Wakeup ถูกกด ให้ LED0 และ LED7 ติดพร้อมกันนาน 1 วินาทีแล้วดับพร้อมกัน มีขั้นตอนดังนี้

1) ใช้โปรแกรม STM32CubeMX สร้างโปรเจ็คขึ้นมาเช่นเดียวกับการทดลองที่ 1 แต่เปลี่ยนไปใช้ขา GPIO จำนวน 9 ขา ดังรูปที่ 2.1
2) พิมพ์โค้ดใน while loop ดังรูปที่ 2.2



รูปที่ 2.1 การกำหนด GPIO

```
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

  //Check whether Wakeup switch at PA0 is pressed
  if (HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET)
  {
    // Turn ON LED0 and LED7 at PE8 and PE15 respectively
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8 | GPIO_PIN_15, GPIO_PIN_SET);
    // Delay 1000 millisecond
    HAL_Delay(1000);

    // Turn OFF LED0 and LED7 at PE8 and PE15 respectively
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8 | GPIO_PIN_15, GPIO_PIN_RESET);
    // Delay 1000 millisecond
    HAL_Delay(1000);
  }

}
```

รูปที่ 2.2 โค้ดในฟังก์ชัน main

3. อธิบายการทำงาน

ฟังก์ชัน MX_GPIO_init ( )

- เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นมา เพื่อตั้งค่า GPIO บนไมโครคอนโทรลเลอร์ให้สอดคล้องกับที่กำหนดไว้ในโปรแกรม
- เริ่มต้นด้วยการ Enable สัญญาณนาฬิกาให้ GPIOA (สำหรับสวิตช์ Wakeup) และ GPIOE (สำหรับ LED)

  __HAL_RCC_GPIOA_CLK_ENABLE();

  __HAL_RCC_GPIOE_CLK_ENABLE();

- กำหนดให้ PA0 ซึ่งเชื่อมต่ออยู่กับสวิตช์ Wakeup เป็นอินพุตแบบ Floating

```
GPIO_InitStruct.Pin = GPIO_PIN_0;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

- กำหนดให้ PE8 ถึง PE15 เป็นเอาต์พุตแบบ Push Pull ที่ Speed 50 MHz

```
GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                      |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
```

ฟังก์ชัน main ( )

- เริ่มต้นการทำงานด้วยฟังก์ชัน HAL_Init() , SystemClock_Config() และ MX_GPIO_Init();
- อ่านสถานะของสวิตช์ Wakeup โดยใช้คำสั่ง

  HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0)
- ตรวจสอบเงื่อนไขเพื่อดูว่าสวิตช์ถูกกดหรือไม่ ด้วยคำสั่ง

  HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == GPIO_PIN_SET

สามารถศึกษารายละเอียดฟังก์ชันที่สามารถใช้งานกับโมดูล GPIO เพิ่มเติมได้จากไฟล์ “**GDrive\STM32CubeMX\User Manual - UM1850 - Description of STM32F1xx HAL drivers**”

4. การทดลอง

1. จงเขียนโปแกรมที่มีการทำงานดังนี้ **เมื่อสวิตช์ Wakeup ถูกกด ให้ LED0 และ LED7 ติดสว่างนาน 1 วินาที พร้อมกัน แล้วดับพร้อมกัน**

2. จงแก้ไขโปรแกรมในการทดลองข้อ 1 ให้ทำงานได้ดังนี้ **เมื่อสวิตช์ Wakeup หรือสวิตช์ Tamper ถูกกด** ให้ LED0 และ LED7 ติดสว่างนาน 1 วินาทีพร้อมกัน แล้วดับพร้อมกัน โดยกำหนดให้ **GPIO มี Speed ในการทำงานที่ 10 MHz**

3. จงเขียนตัวอย่างโค้ดเพื่อตรวจสอบว่า **LED0 ติดอยู่หรือไม่**

---

4. เขียนโปรแกรมเพื่อควบคุมการทำงาน LED ด้วย Joy Switch บนบอร์ด โดยให้สามารถทำงานได้ดังตารางที่ 4.1

ตารางที่ 4.1 การแสดงผลด้วย LED เมื่อใช้งาน Joy Switch

Joy Switch Direction	การทำงานของ LED
UP	เมื่อโยก Joy Switch Up ให้ LED ติดค้างทีละดวงต่อการโยก 1 ครั้ง ถ้าหากไม่มี LED ดวงใดติดอยู่เลยให้เริ่มต้นที่ LED7 ไปยัง LED0 แล้ววนไป LED7 อีกครั้ง **เริ่มต้น** Joy Switch UP -> LED7 ติดค้าง -> Joy Switch UP -> LED7 ดับ และ LED6 ติดค้าง -> Joy Switch UP -> LED6 ดับ และ LED5 ติดค้าง . . . LED1 ดับ และ LED0 ติดค้าง -> Joy Switch UP -> LED0 ดับ และ LED7 ติดค้าง แต่ถ้ามี LED ติดค้างอยู่แล้วให้เริ่มต้นที่ LED ดวงนั้น
Down	เช่นเดียวกับ UP แต่มีทิศทางตรงกันข้าม (LED0 ไปยัง LED7)
Left	เมื่อโยกสวิตช์ 1 ครั้ง LED จะไล่ติดดับทีละ 1 คู่ ทั้งหมด 4 ครั้ง เริ่มต้นจากคู่นอกสุด เข้ามายังคู่ในสุด โดย LED7 ติดคู่กับ LED0, LED6 ติดคู่กับ LED1, LED5 ติดคู่กับ LED2 และ LED4 ติดคู่กับ LED3 โดยหน่วงเวลาตามความเหมาะสมไม่เร็วหรือช้าจนเกินไป
Right	เหมือนกับ Left แต่เริ่มต้นติดจากคู่ในออกไปยังคู่นอก
Center	เมื่อกดให้ LED7 LED5 LED3 LED1 ติดค้างชั่วครู่แล้วดับ จากนั้นให้ LED อีก 4 ดวงที่เหลือติดค้างชั่วครู่แล้วดับ

การทดลองพิเศษ (เลือกทำข้อ 1 ข้อ)

1. จงดัดแปลงการทดลองข้อ 4 ในส่วนของ Joy Switch Up และ Down ให้ LED ติดมากกว่า 1 ดวง สามารถวนกลับมาที่ตำแหน่งเดิมและเปลี่ยนทิศทางได้เช่นเดียวกับการทดลองข้อ 3 และสามารถปรับเพิ่มหรือลดความเร็วในการติดดับได้โดยใช้ Joy Switch Left และ Right โดยให้เชื่อมต่อขา GPIO จำนวน 3 ขาเข้ากับ LED ภายนอกจำนวน 3 ดวง (LED ต้องต่ออนุกรมกับตัวต้านทานเพื่อจำกัดกระแสที่ไหลผ่าน LED ป้องกันไม่ให้เกิดความเสียหาย) โดยให้แสดงผลดังนี้

- เมื่อ LED ติดดับด้วยความเร็วสูงสุดให้ LED ภายนอกดวงแรกติด (ใช้ LED สีแดง)
- ขณะที่ LED กำลังทำงานติดดับ (งูเคลื่อนที่) ให้ LED ภายนอกดวงที่ 2 ติด (ใช้ LED สีน้ำเงิน)
- เมื่อ LED ติดดับด้วยความเร็วต่ำสุดให้ LED ภายนอกดวงสุดท้ายติด (ใช้ LED สีเขียว)

จำนวนขั้นความเร็วที่สามารถปรับได้ จำนวน LED ที่ติด และขา GPIO ที่ใช้ทั้งสามขา ให้ TA เป็นผู้กำหนดก่อนลงมือปฏิบัติ

2. จงสร้างโปรแกรมเปรียบเทียบ (Comparator) เพื่อใช้เปรียบเทียบเลขฐาน 2 ความยาว 4 บิต 2 จำนวน ได้แก่ LED0 – LED3 เป็นตัวตั้ง และ LED4 - LED7 เป็นตัวเปรียบเทียบ และเชื่อมต่อ LED ภายนอกจำนวน 3 ดวงเพื่อแสดงผลลัพธ์จากการเปรียบเทียบ ดังนี้

- เมื่อตัวตั้งมีค่ามากกว่าตัวเปรียบเทียบ ให้ LED สีน้ำเงินติด
- เมื่อตัวตั้งมีค่าเท่ากับตัวเปรียบเทียบ ให้ LED สีเขียวติด
- เมื่อตัวตั้งมีค่าน้อยกว่าตัวเปรียบเทียบ ให้ LED สีแดงติด

ให้ TA เป็นผู้กำหนด GPIO ที่ใช้ก่อนลงมือปฏิบัติ แสดงตำแหน่งปัจจุบันด้วยการทำให้ LED กระพริบ เมื่อเริ่มโปรแกรมให้ LED0 คือตำแหน่งปัจจุบันและทุกบิตมีลอจิก 0 กำหนดให้สวิตช์ต่างๆ ทำงานดังตารางที่ 4.2

ตารางที่ 4.2 การแสดงผลด้วย LED เมื่อใช้งาน Joy Switch

Switch	การทำงานของ LED
Temper	Toggle ค่าลอจิกของตำแหน่งปัจจุบัน
UP	ใช้เลื่อนตำแหน่งปัจจุบันไปทาง LED0 1 ตำแหน่ง และให้ LED0 เป็นตำแหน่งสุดท้ายที่สามารถเลื่อนไปได้ ไม่ต้องวนไป LED7
Down	เช่นเดียวกับอัพ แต่ทิศทางตรงกันข้าม (เลื่อนไปทาง LED7)
Left	ให้ทำการ Shift Left ชุดตัวเลขที่ตำแหน่งปัจจุบันชี้อยู่ 1 ตำแหน่ง (ตัวตั้งหรือตัวเปรียบเทียบ)
Right	ให้ทำการ Shift Right ชุดตัวเลขที่ตำแหน่งปัจจุบันชี้อยู่ 1 ตำแหน่ง (ตัวตั้งหรือตัวเปรียบเทียบ)
Center	ให้ค่าทั้งตัวตั้งและตัวเปรียบเทียบวิ่งไปทาง LED0 และวนมาทาง LED7 และหยุดเมื่อกลับมาสู่ตำแหน่งเดิม
WakeUp	กด 1 ครั้งเพื่อทำการเปรียบเทียบพร้อมแสดงผลลัพธ์ที่ LED ภายนอกตามที่กำหนด

ใบตรวจการทดลองที่ 2

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

1. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

2. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

3. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลายเซ็นผู้ตรวจ

การทดลองข้อ 2 & 3 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ _____

การทดลองข้อ 4 ผู้ตรวจ _____ สัปดาห์ที่ตรวจ _____

คำถามท้ายการทดลอง

1. บอร์ด ET-STM32F ARM KIT มี GPIO จำนวน ..... พอร์ต ขนาดพอร์ตละ .....บิต

2. จงเขียนคำสั่งเพียง 1 คำสั่งที่ทำให้ LED0 LED1 และ LED2 ติดพร้อมกัน
