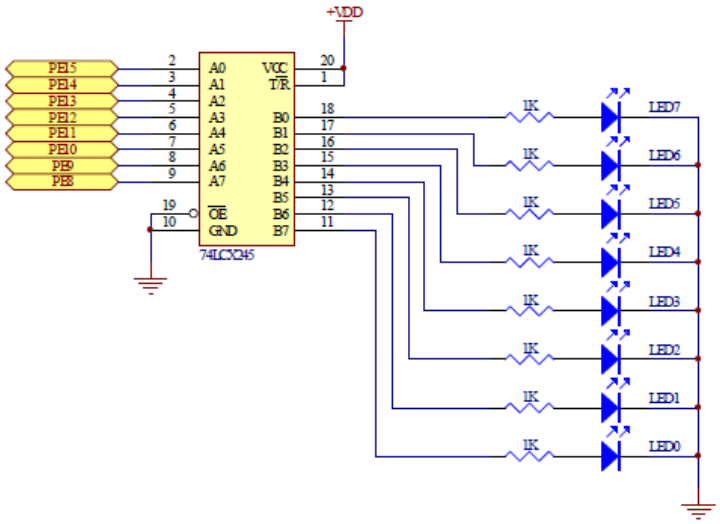
<u>การทดลองที่ 1</u> การใช้งาน LED บนบอร์ดและการใช้งาน Logic Analyzer

วัตถุประสงค์

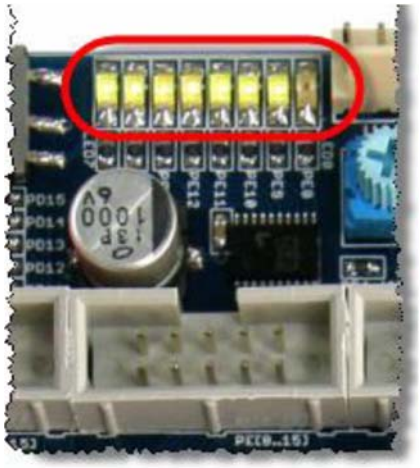
1) สามารถเขียนโปรแกรมควบคุม LED บนบอร์ดได้
2) สามารถใช้ logic analyzer เพื่อวัดสัญญาณดิจิทัลได้
3) เข้าใจว่าการกำหนด optimization level ให้กับคอมไพเลอร์ส่งผลต่อโปรแกรมอย่างไร

1. โครงสร้าง LED บนบอร์ด

บนบอร์ด ET-STM32F ARM KIT มี LED 8 ดวง เชื่อมต่อกับ GPIO พอร์ต E ขา 8 ถึง 15 (PE8 . . . PE15) โดยต่อวงจรแบบขับกระแส (Source Current) โดยใช้กับแหล่งจ่าย +3.3V ทำงานด้วยลอจิก “1” (+3V3) และหยุดทำงานด้วยโลจิก “0” (0V) ดังรูปที่ 1 และ 2 โดย LED0 เชื่อมต่อกับขา PE8 ส่วน LED7 เชื่อมต่อกับ PE15



รูปที่ 1 การเชื่อมต่อ LED



รูปที่ 2 LED บนบอร์ด

2. เขียนโปรแกรมเพิ่มเติม

เปิด Project จาก Lab 0 แล้วแก้ไขโปรแกรมใน while loop ให้เป็นดังรูปที่ 3 และเพิ่มฟังก์ชัน delay ลงในไฟล์ main.c ดังรูปที่ 4 พร้อมประกาศฟังก์ชัน Prototype การเขียนโปรแกรมเพิ่มเติมลงในไฟล์ main.c ควรเขียนให้อยู่ระหว่าง comment `/* USER CODE BEGIN x */` และ `/* USER CODE END x */` เพื่อป้องกันไม่ให้โปรแกรมที่เขียนเพิ่มนั้นหายในกรณีที่สั่ง Generate code ทับ Project เดิม

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

/* USER CODE BEGIN 3 */
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, GPIO_PIN_RESET);
  delay(500);

  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, GPIO_PIN_SET);
  delay(500);

  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, GPIO_PIN_RESET);
  delay(500);

  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, GPIO_PIN_SET);
  delay(500);
}
/* USER CODE END 3 */
```

รูปที่ 3 แก้ไขโปรแกรมใน while loop

```
/* USER CODE BEGIN 4 */

/* Delay in ms by using 2 nested loop

*/
void delay(uint32_t ms)
{
  volatile uint32_t i, j;

  for (i=0; i<ms; i++)
    for (j=0; j<25000; j++)
      ;
}
/* USER CODE END 4 */
```

รูปที่ 4 ฟังก์ชัน delay

3. อธิบายการทำงาน

โปรแกรมจะเริ่มต้นทำงานที่ฟังก์ชัน main โดยจะรันฟังก์ชันดังต่อไปนี้

- ฟังก์ชัน HAL_Init() เพื่อกำหนดค่าเริ่มต้นที่จำเป็นต่อการเริ่มการทำงานให้กับไมโครคอนโทรลเลอร์ โดยโค้ดของฟังก์ชันนี้จะอยู่ในไฟล์ stm32f1xx_hal.c
- ฟังก์ชัน SystemClock_Config() ทำงานต่อจากฟังก์ชัน HAL_Init() เพื่อตั้งค่าวงจรหารและคูณความถี่ภายในไมโครคอนโทรลเลอร์ให้ทำงานตามที่ตั้งค่าไว้จากโปรแกรม STM32CubeMX โดยรายละเอียดของชนิดตัวแปรแบบ Structure และโค้ดของฟังก์ชันที่เรียกใช้ภายในฟังก์ชัน SystemClock_Config() นั้นสามารถศึกษาเพิ่มเติมได้จากไฟล์ stm32f1xx_hal_rcc.h และ stm32f1xx_hal_rcc.c
- ฟังก์ชัน MX_GPIO_Init() ซึ่งมีรายละเอียดดังรูปที่ 5 เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นเพื่อกำหนดให้ขา PE8 และ PE9 ทำหน้าที่เป็นเอาต์พุตตามที่กำหนดไว้ในโปรแกรม ขา PE8 และ PE9 จะทำงานได้ต้องจ่ายสัญญาณนาฬิกาไปยังโมดูล GPIO พอร์ต E ด้วยฟังก์ชัน

`__HAL_RCC_GPIOE_CLK_ENABLE();`

- จากนั้น enable ขาที่ต้องการใช้งานซึ่งได้แก่ PE8 และ PE9 สำหรับควบคุม LED0 และ LED1 ตามลำดับ ผ่านตัวแปรแบบโครงสร้าง GPIO_InitStructure ด้วยคำสั่ง

  GPIO_InitStructure.Pin = GPIO_Pin_8 | GPIO_Pin_9;

  แล้วกำหนดให้ทั้งสองขาทำหน้าที่เป็นขาเอาต์พุตแบบ push pull ที่ความเร็ว 50 MHz ด้วยคำสั่ง

  GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;

  GPIO_InitStructure.GPIO_Speed = GPIO_SPEED_HIGH;

  จากนั้นจึงทำให้การตั้งค่าเกิดผลด้วยการเรียกฟังก์ชัน

  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

```
void MX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

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

  /*Configure GPIO pins : PE8 PE9 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

}
```

รูปที่ 5 รายละเอียดของฟังก์ชัน MX_GPIO_Init

- ตัวแปร GPIO_InitStruct มีชนิดข้อมูลเป็น GPIO_InitTypeDef ซึ่งเป็นชนิดข้อมูลแบบโครงสร้าง มีรายละเอียดดังนี้

```
typedef struct
{
  uint32_t Pin;          //ระบุขาที่ต้องการตั้งค่า
  uint32_t Mode;         //ระบุโหมดการทำงานของขาที่ต้องการตั้งค่า
  uint32_t Pull;         //ระบุการทำงานแบบ Pull-Up หรือ Pull-Down
  uint32_t Speed;        //ระบุความเร็วเมื่อทำงานเป็นขาเอาต์พุต
}GPIO_InitTypeDef;
```

- GPIOE เป็น pointer ที่ถูกสร้างขึ้นด้วยมาโครในไฟล์ stm32f107xc.h

  #define GPIOE ((GPIO_TypeDef *) GPIOE_BASE)

- GPIOE จึงมีสถานะเป็น pointer ที่ชี้ไปยังหน่วยความจำ ณ ตำแหน่งเริ่มต้นของพอร์ต E โดยมีชนิดของข้อมูลเป็น struct GPIO_TypeDef ซึ่งมีข้อมูลย่อยภายใน struct เป็นรีจิสเตอร์ทั้งหมดของพอร์ต E มีรายละเอียดดังนี้

```
typedef struct
{
  __IO uint32_t CRL;  //Control Register Low
  __IO uint32_t CRH;  //Control Register High
  __IO uint32_t IDR;  //Input Data Register
  __IO uint32_t ODR;  //Output Data Register
  __IO uint32_t BSRR;  //Bit Set/Reset Register
  __IO uint32_t BRR;  //Bit Reset Register
  __IO uint32_t LCKR;  //Configuration Lock Register
} GPIO_TypeDef;
```

- ฟังก์ชัน HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, GPIO_PIN_RESET) ใน while loop คือการทำให้ขา PE9 ส่ง**ลอจิก 0** ออกมาส่งผลให้ LED1 ที่เชื่อมต่ออยู่**ดับ**
- ฟังก์ชัน HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, GPIO_PIN_SET) คือการทำให้ขา PE9 ส่ง**ลอจิก 1** ออกมาส่งผลให้ LED1 ที่เชื่อมต่ออยู่**ติด**
- ฟังก์ชัน HAL_GPIO_WritePin และฟังก์ชันอื่นๆ ที่เกี่ยวข้องกับโมดูล GPIO สามารถศึกษาเพิ่มเติมได้จากไฟล์ stm32f1xx_hal_gpio.h และ stm32f1xx_hal_gpio.c หรือศึกษาจากเอกสารคู่มือไฟล์ PDF ที่โฟลเดอร์ "`GDrive\STM32CubeF1`" ซึ่งมีรายละเอียดของฟังก์ชันนี้อยู่ที่หน้า 250 ดังรูปที่ 6

18.2.8 HAL_GPIO_WritePin

Function Name	**void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)**
Function Description	Sets or clears the selected data port bit.
Parameters	• **GPIOx:** where x can be (A..G depending on device used) to select the GPIO peripheral • **GPIO_Pin:** specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15). • **PinState:** specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values: GPIO_BIT_RESET: to clear the port pin GPIO_BIT_SET: to set the port pin
Return values	• None
Notes	• This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.

รูปที่ 6 รายละเอียดของฟังก์ชัน `HAL_GPIO_WritePin`

- ฟังก์ชัน void delay (uint32_t ms) เป็นฟังก์ชันหน่วงเวลาเพื่อหยุดการทำงานของไมโครโพรเซสเซอร์ชั่วคราวด้วยการไม่ให้ไป execute คำสั่งอื่น มีการทำงานเป็นการวนลูป 2 ลูปซ้อนกัน ระยะเวลาของการหน่วงเวลาขึ้นอยู่กับจำนวนของการวนลูปนอกซึ่งถูกกำหนดค่าผ่านตัวแปร ms และการกำหนด optimization level ตอนคอมไพล์โปรแกรม

การทดลอง

1. ให้ตรวจสอบการหน่วงเวลาจากการเรียกฟังก์ชัน delay(500) ว่าหน่วงเวลาเป็นระยะเวลา 500 ms หรือไม่ ถ้าไม่ใช่ให้เปลี่ยนเงื่อนไขของลูปข้างใน (inner loop) ให้สามารถหน่วงเวลาได้ 500 ms แล้วบันทึกผล

2. เปลี่ยน optimization level ให้เป็น level 3 แล้วตรวจสอบดูว่าผลของการเรียกฟังก์ชัน delay(500) เปลี่ยนแปลงหรือไม่ อย่างไร เพราะสาเหตุใด ถ้ามีการเปลี่ยนแปลงแล้วเงื่อนไขของลูปในควรเปลี่ยนแปลงอย่างไร เพื่อให้ผลลัพธ์ของการเรียกฟังก์ชันเหมือนกับการทดลองที่ 1

3. ให้เปลี่ยน optimization level กลับมาเป็น level 0 พร้อมกับใช้เงื่อนไขของลูปข้างในตามผลการทดลองที่ 1 จากนั้นจงเขียนโปรแกรมให้ LED 3 ดวง ได้แก่ LED2 LED1 และ LED0 แสดงผลลัพธ์ของวงจรนับขึ้น 3 บิต (0b000 – 0b111) โดยหน่วงเวลาที่ 300 ms และทำให้ LED ที่เหลือดับ พร้อมใช้ logic analyzer ตรวจจับสัญญาณ เมื่อพบว่า LED ทั้งสามดวงมีค่า 0b110 ให้แสดงผลค้างไว้ 10 µs แล้วจึงให้ logic analyzer ทำงานต่อ

ใบตรวจการทดลองที่ 1

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

1. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

2. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

3. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลายเซ็นผู้ตจรวจ

การทดลองข้อ 3 ผู้ตรวจ _____ วันที่ตรวจ _____

คำถามท้ายการทดลอง

1. ไฟล์ hex ของการทดลองที่ 3 มีขนาด _____ KB โดยได้ข้อมูลจาก _____

2. GPIOE เป็นมาโคร pointer เพื่อชี้ไปยังตำแหน่งเริ่มต้นในหน่วยความจำของโมดูล GPIO พอร์ต E จงหาตำแหน่งเริ่มต้นนี้ว่าอยู่ที่ตำแหน่งที่เท่าไร (คำตอบเป็นตัวเลข) และถูกจัดเป็นส่วนไหนใน memory space โดยศึกษาจากไฟล์ stm32f107xc.h

   _____

   _____

   _____

   _____