

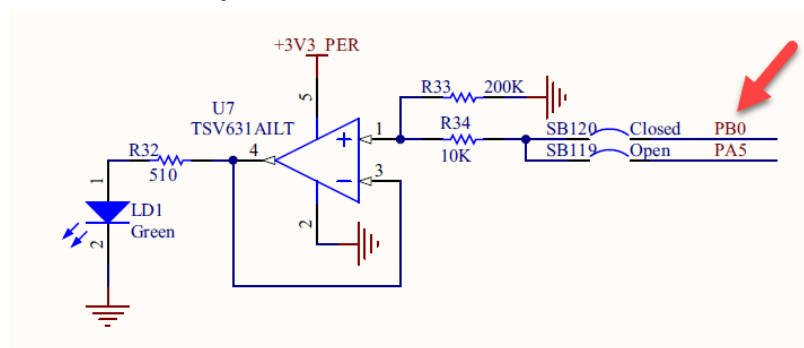
การทดลองที่ 1 การใช้งาน LED บนบอร์ดและการใช้งาน Debug mode

วัตถุประสงค์

- 1) สามารถเขียนโปรแกรมควบคุม LED บนบอร์ดได้
- 2) สามารถใช้ Debug mode ภายในโปรแกรม STM32CubeIDE ได้

1. โครงสร้าง LED บนบอร์ด

บนบอร์ด Nucleo-F767ZI มี User LED LD 1 ที่เชื่อมต่อกับ GPIO พอร์ต B ขา 0 LED จะติดเมื่อป้อนลอจิก “1” (3.3 v) และจะดับเมื่อป้อนลอจิก “0” (0V) ดังรูปที่ 1



รูปที่ 1 การเชื่อมต่อ LED

2. เขียนโปรแกรมเพิ่มเติม

เปิด Project จาก Lab 0 แล้วแก้ไขโปรแกรม ดังนี้

- ประกาศตัวแปรโกลบอล num และ led0 ดังรูปที่ 2
- แก้ไข while loop ดังรูปที่ 3

การเขียนโปรแกรมเพิ่มเติมลงในไฟล์ main.c ควรเขียนให้อยู่ระหว่าง comment `/* USER CODE BEGIN x */` และ `/* USER CODE END x */` เพื่อป้องกันไม่ให้โปรแกรมที่เขียนเพิ่มนั้นโดนลบในกรณีที่สั่ง Generate code ทับ Project เดิม

```

42
43 /* Private variables -----
44
45 /* USER CODE BEGIN PV */
46 uint8_t num = 0;
47 uint8_t led0 = 0;
48 /* USER CODE END PV */
49

```

รูปที่ 2 ประกาศตัวแปร Global Variable

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    //LD1 PB0 ON
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
    led0 = 1;
    //Delay
    HAL_Delay(500);
    //LD1 PB0 OFF
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    led0 = 0;
    //Delay
    HAL_Delay(500);

    num++;
    if(num > 7)
        num = 0;
}
/* USER CODE END 3 */

```

รูปที่ 3 แก้ไขโปรแกรมใน while loop

3. อธิบายการทำงาน

โปรแกรมจะเริ่มต้นทำงานที่ฟังก์ชัน main โดยจะรันฟังก์ชันดังต่อไปนี้

- ฟังก์ชัน HAL_Init() เพื่อกำหนดค่าเริ่มต้นที่จำเป็นต่อการเริ่มการทำงานให้กับไมโครคอนโทรลเลอร์ โดยโค้ดของฟังก์ชันนี้จะอยู่ในไฟล์ stm32f7xx_hal.c
- ฟังก์ชัน SystemClock_Config() ทำงานต่อจากฟังก์ชัน HAL_Init() เพื่อตั้งค่าวงจรหารและคูณความถี่ภายในไมโครคอนโทรลเลอร์ให้ทำงานตามที่ตั้งค่าไว้จากโปรแกรม STM32CubeMX โดยรายละเอียดของชนิดตัวแปรแบบ Structure และโค้ดของฟังก์ชันที่เรียกใช้ภายในฟังก์ชัน SystemClock_Config() นั้นสามารถศึกษาเพิ่มเติมได้จากไฟล์ stm32f7xx_hal_rcc.h และ stm32f7xx_hal_rcc.c
- ฟังก์ชัน MX_GPIO_Init() ซึ่งมีรายละเอียดดังรูปที่ 4 เป็นฟังก์ชันที่โปรแกรม STM32CubeMX สร้างขึ้นเพื่อกำหนดให้ขา PB0 ทำหน้าที่เป็นเอาต์พุตตามที่กำหนดไว้ในโปรแกรม ขา PB0 จะทำงานได้ต้องจ่ายสัญญาณนาฬิกาไปยังโมดูล GPIO พอร์ต B ด้วยฟังก์ชัน

__HAL_RCC_GPIOB_CLK_ENABLE();

- ตั้งค่าให้ขา PB0 มีระดับลอจิกเริ่มต้นเป็นลอจิก 0 ตามที่ได้กำหนดไว้ในโปรแกรม STM32CubeMX ด้วยคำสั่ง

HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);

- จากนั้น enable ขาที่ต้องการใช้งานซึ่งได้แก่ขา 0 ของ Port B ผ่านตัวแปรแบบโครงสร้าง GPIO_InitStructure ด้วยคำสั่ง

GPIO_InitStructure.Pin = GPIO_Pin_0;

แล้วกำหนดให้ทั้งสองขาทำหน้าที่เป็นขาเอาต์พุตแบบ push pull ที่ความเร็วแบบ High ด้วยคำสั่ง

GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;

GPIO_InitStruct.Pull = GPIO_NOPULL;

```
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
```

จากนั้นจึงทำให้การตั้งค่าเกิดผลด้วยการเรียกฟังก์ชัน

```
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
```

```
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;

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

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);

    /*Configure GPIO pins : PB0 PB14 PB7 */
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

รูปที่ 4 รายละเอียดของฟังก์ชัน MX_GPIO_Init

- ตัวแปร GPIO_InitStruct มีชนิดข้อมูลเป็น GPIO_InitTypeDef ซึ่งเป็นชนิดข้อมูลแบบโครงสร้าง มีรายละเอียดดังนี้

```
typedef struct
{
    uint32_t Pin;           //ระบุนาที่ต้องการตั้งค่า
    uint32_t Mode;          //ระบุนโหมดการทำงานของนาที่ต้องการตั้งค่า
    uint32_t Pull;          //ระบุนการทำงานแบบ Pull-Up หรือ Pull-Down
    uint32_t Speed;         //ระบุนความเร็วเมื่อนาทำงานเป็นนาเอาต์พุต
}GPIO_InitTypeDef;
```

- GPIOB เป็น pointer ที่ถูกสร้างขึ้นด้วยมาโครในไฟล์ stm32f767xx.h

```
#define GPIOB ((GPIO_TypeDef *) GPIOB_BASE)
```

- GPIOB จึงมีสถานะเป็น pointer ที่ชี้ไปยังหน่วยความจำ ณ ตำแหน่งเริ่มต้นของพอร์ต B โดยมีชนิดของข้อมูลเป็น struct GPIO_TypeDef ซึ่งมีข้อมูลย่อยภายใน struct เป็นรีจิสเตอร์ทั้งหมดของพอร์ต B 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;
```

- เมื่อเข้าสู่ Infinite Loop คำสั่งแรก คือ HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET) เป็นการสั่งขา PB0 มีค่าลอจิก 1 ซึ่งจะทำให้ LD0 ติด
- ตัวแปรแบบโกลบอล led0 มีจุดประสงค์สำหรับแสดงการติดดับ ภายในโปรแกรม STM32CubeIDE โดยจะมีค่าตามค่าลอจิกของขา PB0 ดังนั้นเมื่อขา PB0 มีค่าลอจิก 1 จึงทำให้ตัวแปร led0 มีค่า 1 ด้วยคำสั่ง led0 = 1
- คำสั่ง HAL_Delay(500) ใช้หน่วยเวลาการทำงาน 500 ms ทำให้สามารถมองเห็น LED ติดสว่างได้นาน 500 ms
- คำสั่ง คือ HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET) เป็นการสั่งขา PB0 มีค่าลอจิก 0 ซึ่งจะทำให้ LD0 ดับ
- คำสั่ง led0 = 0 เปลี่ยนแปลงค่าตัวแปร led0 ให้มีค่าเป็น 0 สอดคล้องกับค่าลอจิกของขา PB0 ที่เปลี่ยนไป
- คำสั่ง HAL_Delay(500) ใช้หน่วยเวลาการทำงาน 500 ms ทำให้สามารถมองเห็น LED ดับนาน 500 ms
- ฟังก์ชัน HAL_GPIO_WritePin และฟังก์ชันอื่นๆ ที่เกี่ยวข้องกับโมดูล GPIO สามารถศึกษาเพิ่มเติมได้จากไฟล์ stm32f7xx_hal_gpio.h และ stm32f7xx_hal_gpio.c หรือศึกษาจากเอกสารคู่มือจากไฟล์ UM1905 Description of STM32F7 HAL and low-layer drivers ดังรูปที่ 5

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..K) 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_PIN_RESET: to clear the port pin
 - GPIO_PIN_SET: to set the port pin

Return values

- **None:**

รูปที่ 5 รายละเอียดของฟังก์ชัน HAL_GPIO_WritePin

4. การใช้งาน Debug mode ภายในโปรแกรม STM32CubeIDE

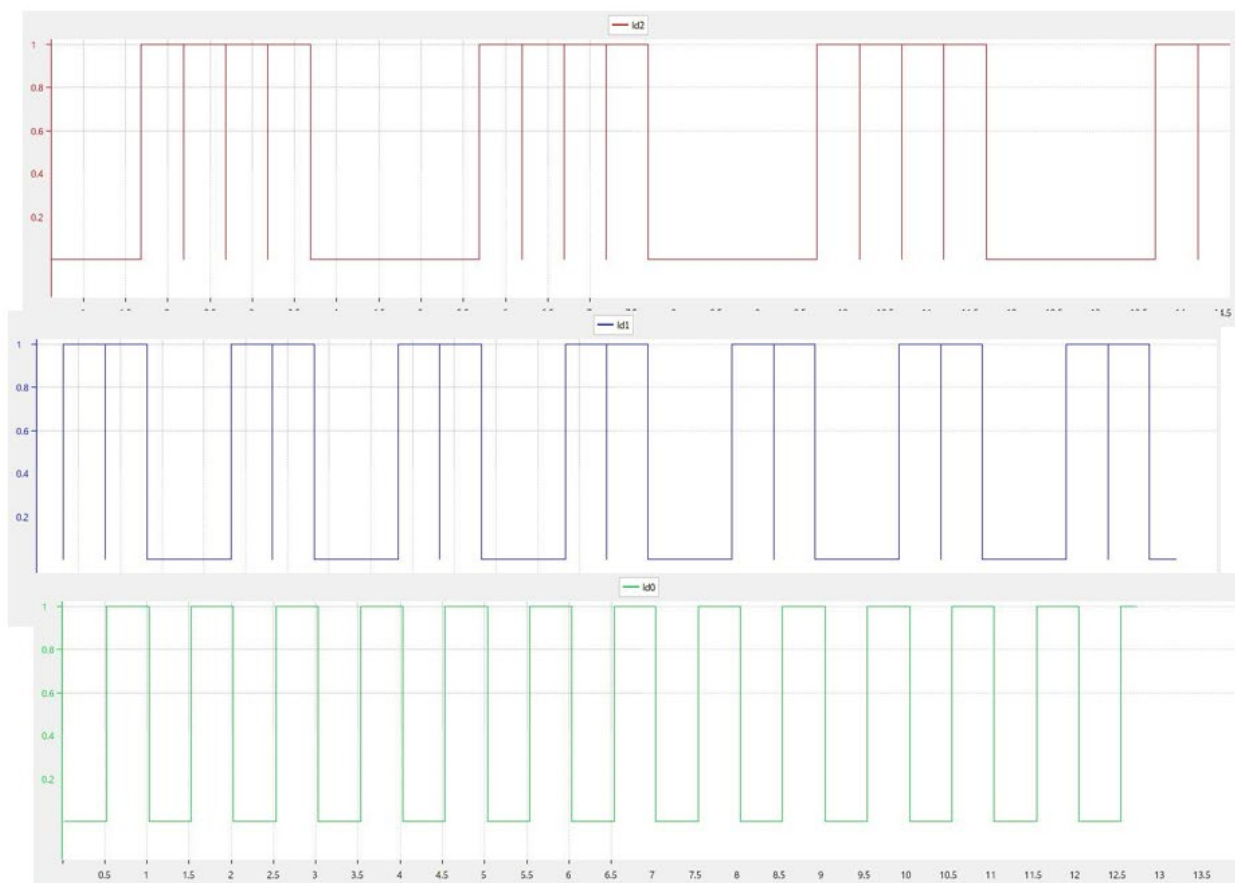
การพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์นั้นสามารถใช้งานในโหมดดีบั๊กได้เช่นเดียวกับการพัฒนาโปรแกรมโดยทั่วไป เพียงแต่การประมวลผลข้อมูลเกิดขึ้นที่ไมโครคอนโทรลเลอร์ แล้วจึงนำผลลัพธ์จากการประมวลผลมาแสดงภายในโปรแกรมที่ใช้พัฒนามบนเครื่องคอมพิวเตอร์

โปรแกรมสำหรับพัฒนาไมโครคอนโทรลเลอร์หลายๆ โปรแกรมมีฟังก์ชันสำหรับการดีบั๊กหลายอย่างด้วยกัน เช่น โปรแกรม STM32CubeIDE สามารถตั้งค่า Breakpoint, อ่านค่าตัวแปร, อ่านค่าจากหน่วยความจำที่ตำแหน่งต่างๆ, รีจิสเตอร์ทั่วไป และแสดงค่าที่อ่านได้แบบ Timing Diagram โดยใช้ Serial wire viewer (SWV) เป็นต้น

ศึกษาการใช้งาน Debug mode ได้จากคลิป ดังต่อไปนี้ <https://youtu.be/7jw-9-8j2ok>

การทดลอง

1. จงแก้ไขโค้ดจาก Lab 0 ดังรูป 2 และ 3 ศึกษาการใช้งาน Debug mode จนกระทั่งแสดง SWV Data Trace Timeline Graph ได้
2. จงแก้ไขโปรแกรมในการทดลองข้อ 1 เพื่อแสดงระดับสัญญาณวงจรม้วนขึ้น 3 บิต โดยใช้ขา PB14 (MSB) PB7 และ PB0 (LSB) โดยให้หน่วยเวลาที่ 300 ms โดยแสดงผลใน SWV Data Trace Timeline Graph แล้ว Save รูป จัดเรียงดังตัวอย่างต่อไปนี้



ใบตรวจการทดลองที่ 1

Microcontroller Application and Development 2566

วัน/เดือน/ปี _____ กลุ่มที่ _____

1. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____
2. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____
3. รหัสนักศึกษา _____ ชื่อ-นามสกุล _____

ลายเซ็นผู้ตรวจ

การทดลองข้อ 1 ผู้ตรวจ _____ วันที่ตรวจ ☐ W ☐ W+1

การทดลองข้อ 2 ผู้ตรวจ _____ วันที่ตรวจ ☐ W ☐ W+1

คำถามท้ายการทดลอง

1. GPIOB เป็นมาโคร pointer เพื่อชี้ไปยังตำแหน่งเริ่มต้นในหน่วยความจำของโมดูล GPIO พอร์ต B จงหาตำแหน่งเริ่มต้นนี้ว่าอยู่ที่ตำแหน่งที่เท่าไร (ตอบเป็นตัวเลข) และถูกจัดเป็นส่วนไหนใน memory space
