<u>การทดลองที่ 0</u> เริ่มต้นการใช้งาน (Getting Started)

วัตถุประสงค์

1) สามารถติดตั้งโปรแกรมที่เกี่ยวข้องการการปฏิบัติการ
2) สามารถพัฒนาโปรแกรมโดยใช้ซอฟต์แวร์ที่เกี่ยวข้องได้
3) สามารถดาวน์โหลดโปรแกรมที่พัฒนาลงบอร์ดไมโครคอนโทรลเลอร์ได้

1. การติดตั้งและใช้งานโปรแกรม STM32CubeMX

การพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์ (Microcontroller Unit; MCU) ตระกูล STM32F7 นั้น ขั้นแรกต้องกำหนดหน้าที่ให้กับขาของไอซีไมโครคอนโทรลเลอร์ให้ถูกต้อง เนื่องจากแต่ละขาสามารถทำหน้าที่ได้หลายแบบ กลายเป็นความยากลำบากในการเริ่มต้นพัฒนา เนื่องจากต้องศึกษาเอกสารทางเทคนิคจึงจะสามารถเลือกและกำหนดหน้าที่การทำงานได้อย่างถูกต้อง ดังนั้นทางบริษัทผู้ผลิตไมโครคอนโทรลเลอร์จึงได้พัฒนาโปรแกรมเพื่ออำนวยความความสะดวกในการกำหนดหน้าที่ (Configure) และเริ่มต้นการทำงาน (Initial) ให้กับขาไมโครคอนโทรลเลอร์ที่ผู้พัฒนาต้องการเลือกใช้ ได้แก่โปรแกรม STM32CubeMX

1.1 เลือกไมโครคอนโทรลเลอร์

ให้ติดตั้ง JAVA 8 Runtime ก่อนแล้วจึงติดตั้งโปรแกรม STM32CubeMX โดยสามารถดาวน์โหลดได้จากเว็บไซต์ของบริษัท ST หรือเข้าไปยัง Google Drive ของรายวิชา (ไฟล์ **en.stm32cubemx_v5.4.0.zip**) จากนั้นให้เปิดโปรแกรมขึ้นมาแล้วเลือก New Project จะพบกับหน้าต่างดัง รูปที่ 1.1 จากนั้นเลือกหมายเลขไมโครคอนโทรลเลอร์ให้ตรงกับการใช้งาน สำหรับบอร์ดทดลองที่ใช้ในวิชานี้ได้แก่ ไมโครคอนโทรลเลอร์หมายเลข STM32F767ZI แล้วกดปุ่ม Start Project

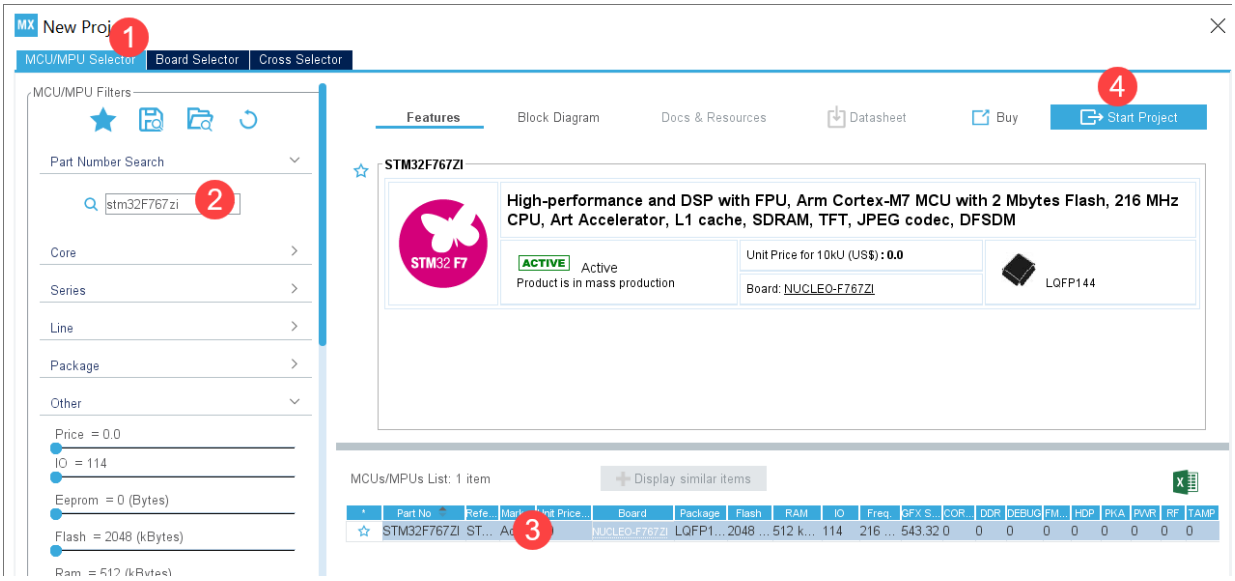
1.2 เลือกขาที่ต้องการใช้งาน

รูปที่ 1.2 แสดงแผนภาพทางด้านบนของบอร์ดซึ่งประกอบไปด้วย User LED ที่ผู้ใช้สามารถเขียนโปรแกรมควบคุมได้จำนวน 3 ดวง และปุ่มกด 2 ปุ่ม โดย LED 3 ดวงมีรายละเอียดดังรูปที่ 1.3 และ รูปที่ 1.4 ดังนี้

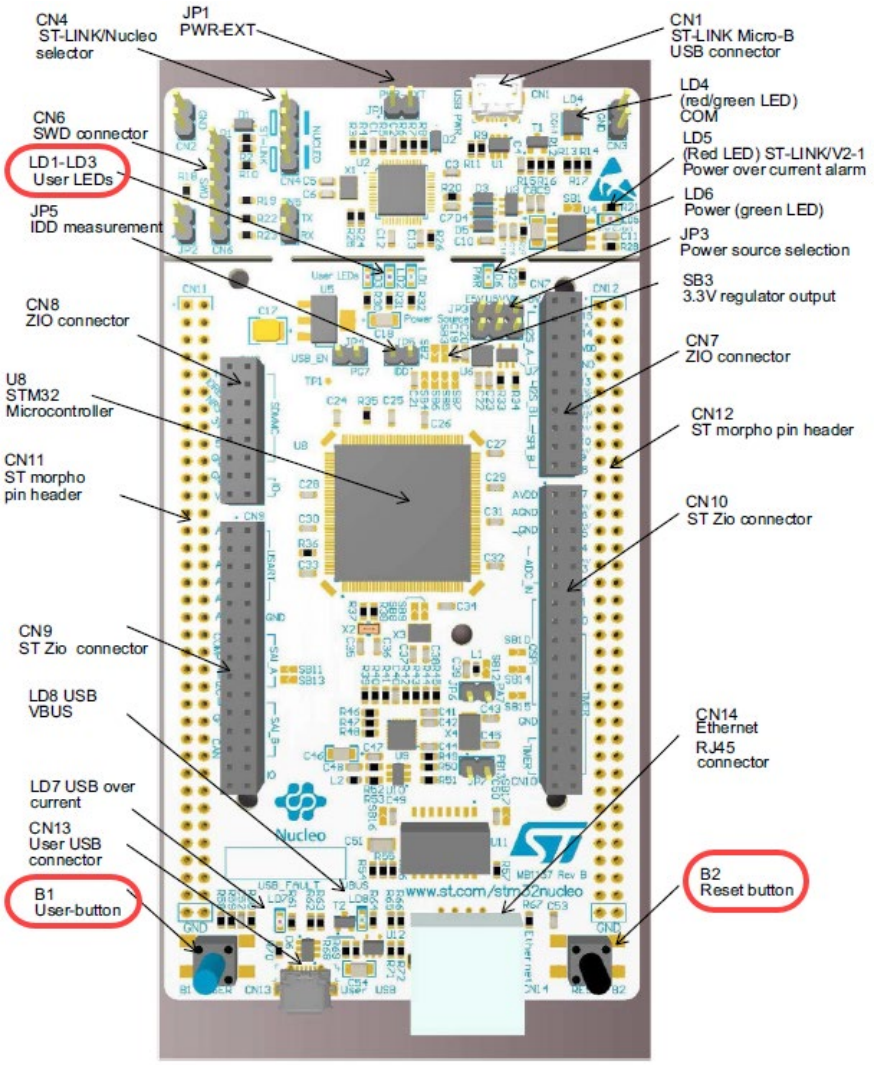
1) LD1 สีเขียว เชื่อมต่อกับไมโครคอนโทรลเลอร์ที่ขา PB0
2) LD2 สีน้ำเงิน เชื่อมต่อกับไมโครคอนโทรลเลอร์ที่ขา PB7
3) LD3 สีแดง เชื่อมต่อกับไมโครคอนโทรลเลอร์ที่ขา PB14

ส่วนปุ่มกด 2 ปุ่มได้แก่

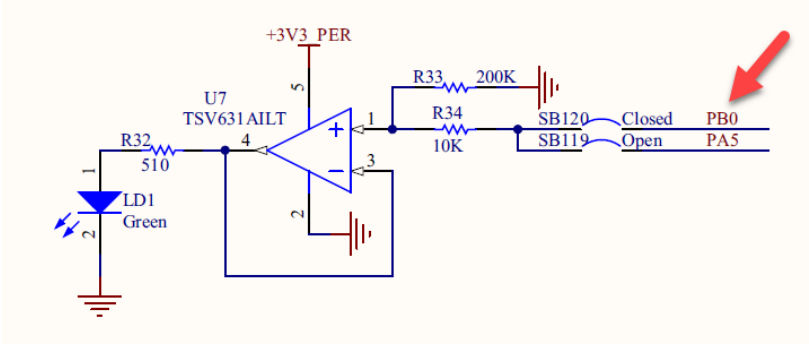
1) B1 (สีน้ำเงิน) คือ ปุ่มที่ผู้ใช้เขียนโปรแกรมควบคุมได้ เชื่อมต่อกับไมโครคอนโทรลเลอร์ที่ขา PC13
2) B2 (สีดำ) คือ ปุ่ม Reset



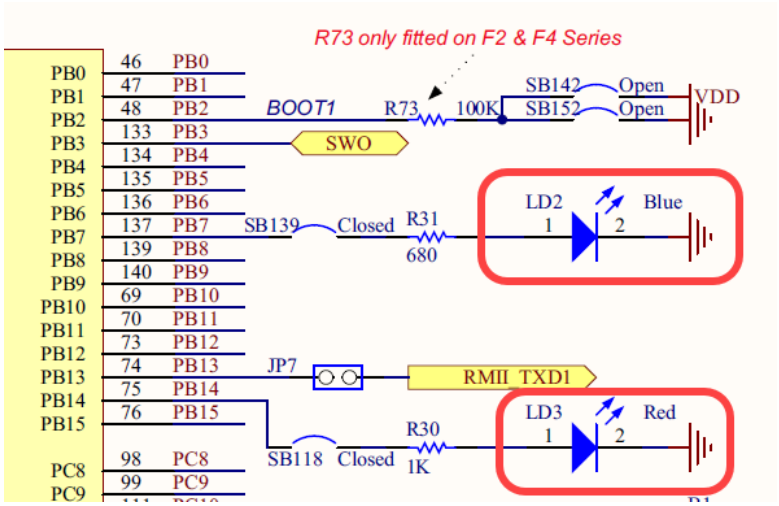
รูปที่ 1.1 หน้าต่าง New Project



รูปที่ 1.2 แผนภาพด้านบนของบอร์ด Nucleo-F767ZI



รูปที่ 1.3 แสดงการเชื่อมต่อ LED LD1 ที่ขา PB0



รูปที่ 1.4 แสดงการเชื่อมต่อ LED LD2 และ LD3 ที่ขา PB7 และ PB14 ตามลำดับ

ส่วนรูปที่ 1.5 แสดงรายชื่อขาของไมโครคอนโทรลเลอร์ที่เชื่อมต่อกับตัวเชื่อมต่อ (Connector) ตำแหน่งต่างๆ เอาไว้สำหรับเชื่อมต่อขาสัญญาณภายนอกไปยัง MCU

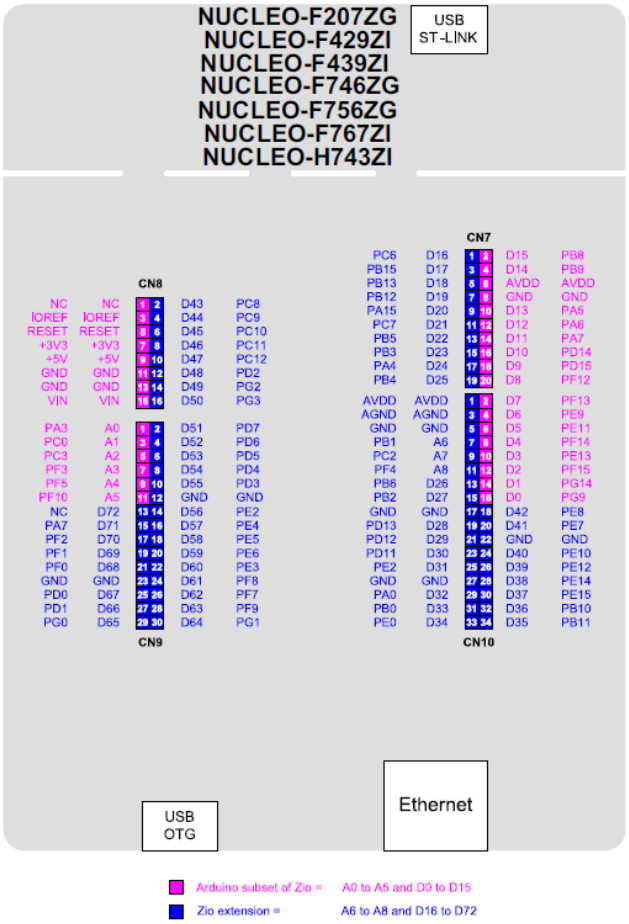
หากต้องการเขียนโปรแกรมควบคุม LED LD1 สีเขียว ต้องกำหนดให้ขา PB0 ซึ่งเชื่อมต่ออยู่กับ LD1 ให้ทำหน้าที่เป็นขาเอาต์พุตโดยคลิกซ้ายที่ขาดังกล่าวแล้วเลือก GPIO_Output ดังรูปที่ 1.6

1.3 กำหนดสัญญาณนาฬิกา

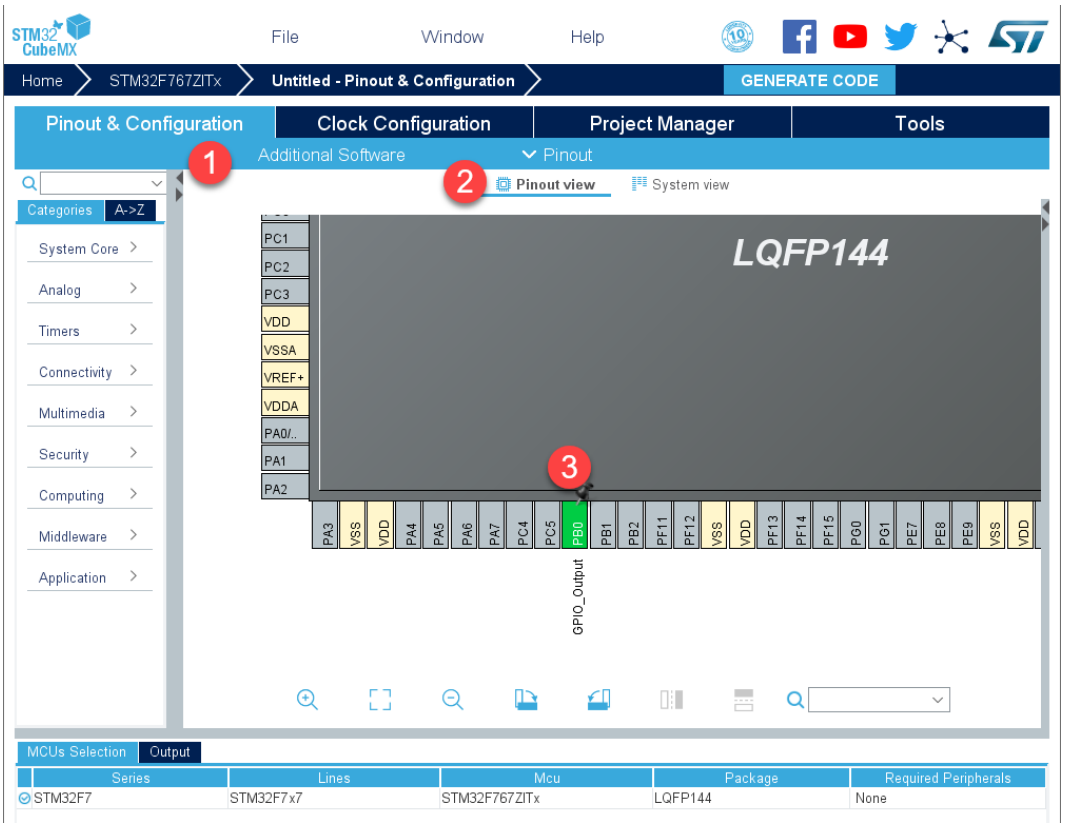
สามารถกำหนดสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์ได้จาก 2 แหล่งจ่าย ได้แก่

1) สัญญาณนาฬิกาภายในไอซีไมโครคอนโทรลเลอร์แบบความถี่สูง (HSI) ขนาด 16 MHz
2) Crystal Oscillator 8 MHz ที่ติดตั้งอยู่บนบอร์ด (HSE)

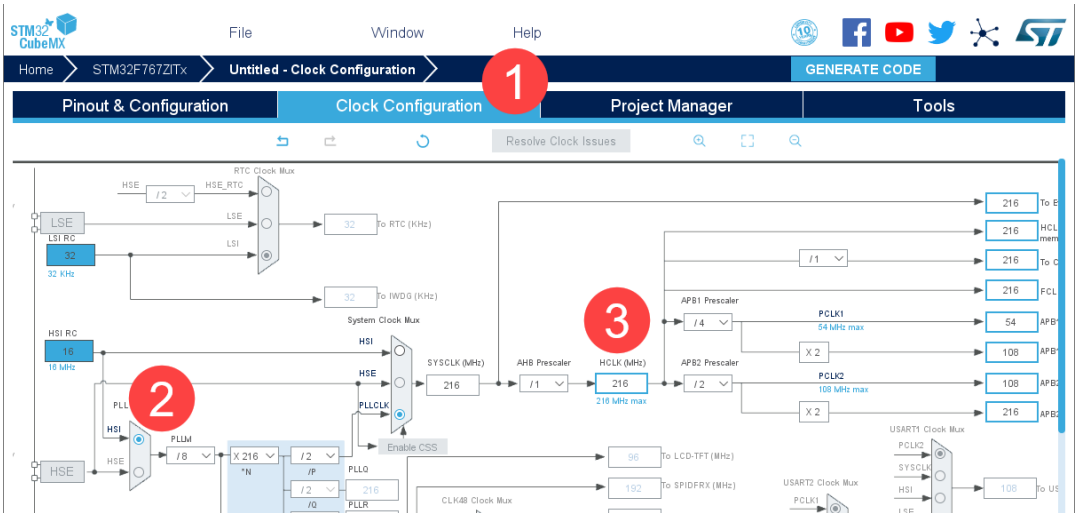
การทดลองนี้จะเลือกใช้การตั้งค่าที่กำหนดไว้เป็นค่าเริ่มต้นอยู่แล้วได้แก่ สัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์ความถี่ 16 MHz โดยสังเกตที่ Tab Clock Configuration แสดง HSI RC มีความถี่ 16 MHz จากนั้นกำหนดความถี่ในการทำงานของไมโครคอนโทรลเลอร์ให้เป็น 216 MHz ซึ่งเป็นความถี่สูงสุดที่รับได้ดังรูปที่ 1.7 แล้วกดปุ่ม OK บนไดอะล็อคที่ปรากฏ จากนั้นโปรแกรมจะคำนวณค่าที่เหมาะสมให้กับวงจรหารและวงจรคูณความถี่ภายในให้สอดคล้องกับความถี่ของแหล่งจ่ายและความถี่ในการทำงานของไมโครคอนโทรลเลอร์และโมดูลอื่นๆ



รูปที่ 1.5 แสดงรายชื่อขาของไมโครคอนโทรลเลอร์ที่เชื่อมต่อกับตัวเชื่อมต่อตำแหน่งต่างๆ



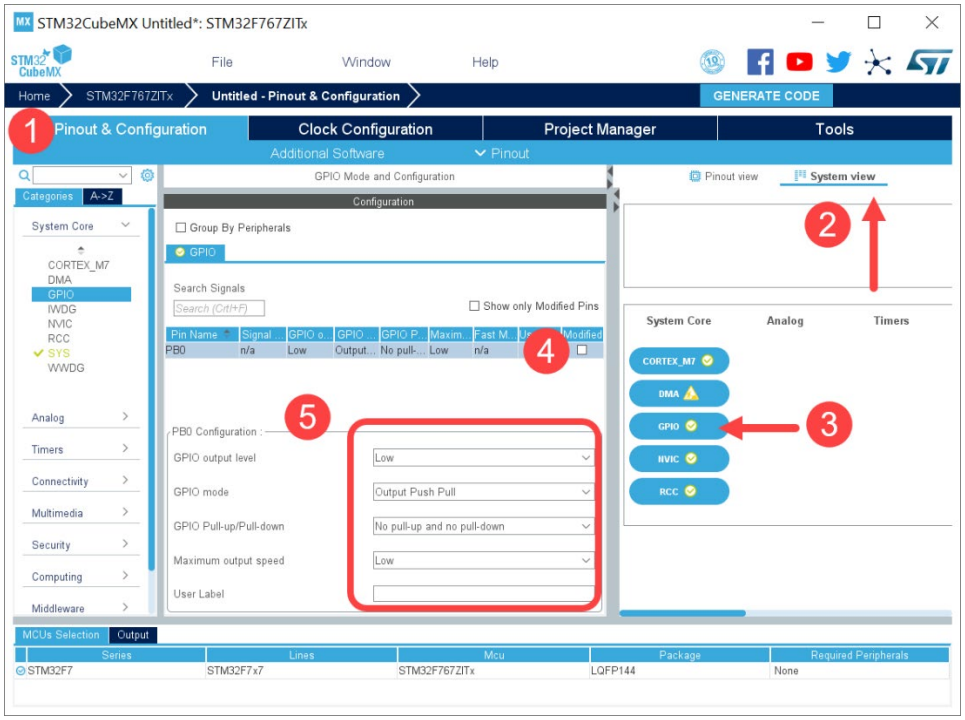
รูปที่ 1.6 กำหนดขา PB0 ให้ทำหน้าที่ GPIO_Output



รูปที่ 1.7 การกำหนดความถี่ของสัญญาณนาฬิกา

1.4 กำหนดรายละเอียดของโมดูลที่เลือกใช้

ไปที่ Tab Pinout & Configuration ที่หน้าต่างขวามือเลือก GPIO แล้วจะพบกับหน้าต่างให้กำหนดรายละเอียดดังรูปที่ 1.8 ให้กำหนดรายละเอียด GPIO output level, GPIO mode, GPIO Pull-up/Pull-down และ Maximum output speed ให้กับ PB0 ดังรูปที่ 1.8

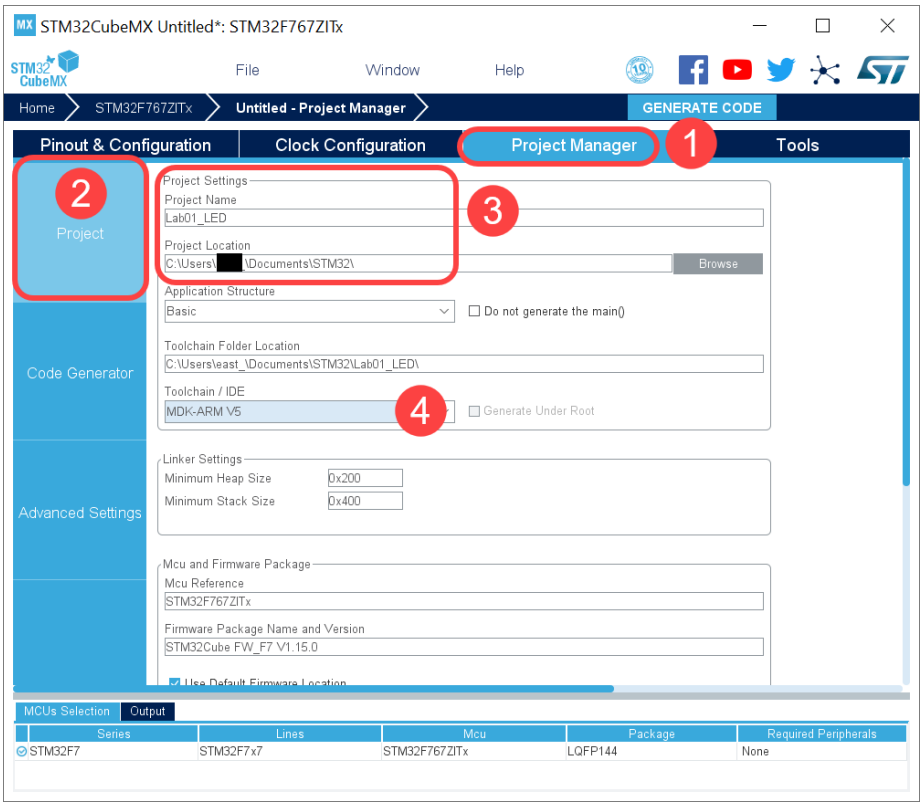


รูปที่ 1.8 กำหนาดรายเอียดของ GPIO

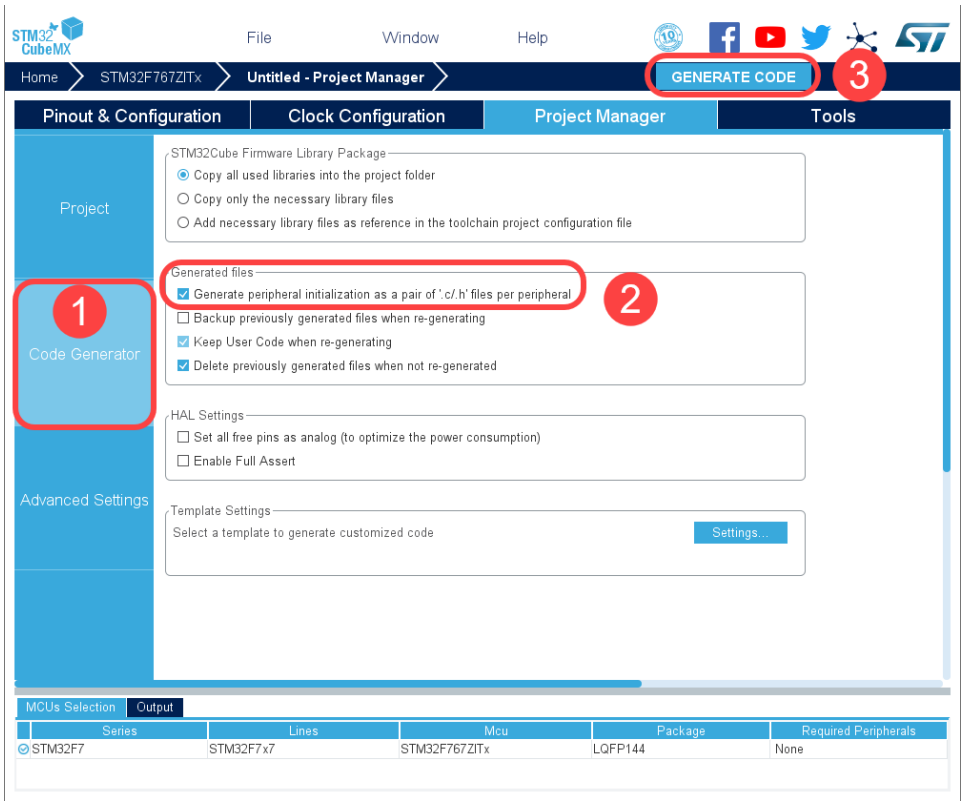
1.5 ติดตั้ง Package

สามารถข้ามขั้นตอนข้อนี้ไปได้ โดยเมื่อข้ามไปทำขั้นตอน 1.6 โปรแกรมจะดาวน์โหลด Package ที่เกี่ยวข้องให้โดยอัตโนมัติ แต่เพื่อความสะดวกสามารถติดตั้ง Package ด้วยตนเองได้ที่เมนู Help -> Manage embedded software

packages ที่มุมล่างซ้ายกดปุ่ม "From Local" เลือกไฟล์ **stm32cube_fw_f7_v1150.zip** จาก Google Drive รายวิชา



(1)



(2)

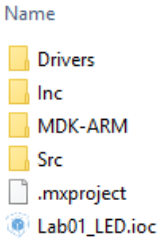
รูปที่ 1.9 การตั้งค่าสำหรับการ Generate

1.6 Generating Code

จากหลังที่เลือกใช้และกำหนดรายละเอียดให้กับโมดูลที่เลือกใช้งานแล้ว ขั้นตอนต่อไปจะเป็นการ generate code เพื่อนำไปพัฒนาต่อในโปรแกรม Keil โดยไปที่ Tab Project Manager ด้านซ้ายมือเลือก Tab Project แล้วกำหนดชื่อ Project เป็น Lab01_LED แล้วเลือกโฟลเดอร์ตามที่ต้องการ และเลือกประเภทของ IDE ให้เป็น MDK-ARM V5 ดังรูปที่ 1.9 (1) และที่ Tab Code Generator กำหนดตัวเลือกตรงกับรูปที่ 1.9 (2) แล้วกดปุ่ม Generate Code ด้านบน เพื่อ Generate Code สำหรับตั้งค่าไมโครคอนโทรลเลอร์ตามที่ได้เลือกไว้ และยังเป็นการบันทึก STM32CubeMX Project ในคราวเดียว ซึ่ง Project ที่ถูกเซพจะมีนามสกุลไฟล์เป็น **.ioc** ซึ่งสามารถกลับมาแก้ไขภายหลังแล้ว Generate Code ซ้ำได้

2. การใช้งานโปรแกรม Keil

สามารถติดตั้งโปรแกรม Keil โดยศึกษาจากเอกสาร **Lab-00-Keil Installation** เมื่อติดตั้งเรียบร้อยแล้วให้ไปยังโฟลเดอร์ที่โปรแกรม STM32CubeMX ได้ Generate Project ไว้ ดังรูปที่ 2.1 โดยมีรายละเอียดดังตารางที่ 2.1



รูปที่ 2.1 โครงสร้างโฟลเดอร์

ตารางที่ 2.1 รายละเอียดของโฟลเดอร์ที่ถูกสร้างจากโปรแกรม STM32CubeMX

โฟลเดอร์	รายละเอียดไฟล์ที่บรรจุอยู่ภายใน
Drivers	Firmware CMSIS และ STM32
Inc	ไฟล์ Header ที่เกี่ยวข้อง
MDK-ARM	ไฟล์ที่เกี่ยวข้องกับโปรแกรม Keil
Src	ไฟล์ภาษา C ที่เกี่ยวข้อง
ไฟล์ Lab01_LED.ioc	ไฟล์ของโปรแกรม STM32CubeMX สาสมารถเปิดเพื่อแก้ไขแล้ว Generate ใหม่ได้

2.1 เปิด Project

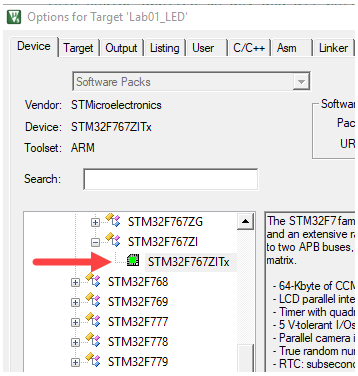
เปิด Project โดยเข้าไปที่โฟลเดอร์ MDK-ARM แล้วเปิดไฟล์นามสกุล .uvprojx จะพบกับหน้าต่าง Project ที่แสดงโครงสร้างโฟลเดอร์และไฟล์ใน Project จากนั้นในโฟลเดอร์ Application/User ให้เปิดไฟล์ main.c และ gpio.c จะพบฟังก์ชันที่สำคัญดังตารางที่ 2.2

ตารางที่ 2.2 รายละเอียดของฟังก์ชันในไฟล์ main.c และ gpio.c

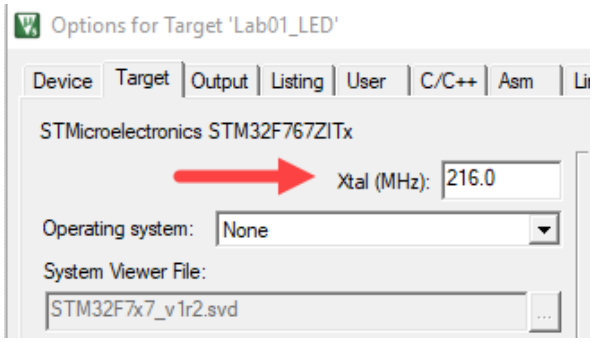
ฟังก์ชัน	รายละเอียด
main.c\main	ใช้เขียน User Code
gpio.c\MX_GPIO_Init	ตั้งค่าขา PB0 ให้ทำหน้าที่เอาต์พุตตามที่กำหนดใน STM32CubeMX
main.c\SystemClock_Config	ตั้งค่าสัญญาณนาฬิกาให้กับโมดูลต่างๆ ในไมโครคอนโทรลเลอร์

2.2 การตรวจสอบการตั้งค่า

ก่อนการสร้างไฟล์สำหรับโปรแกรมลงบอร์ดให้ตรวจสอบการตั้งค่าดังรูปที่ 2.2 โดยให้เชื่อมต่อบอร์ดเข้ากับเครื่องคอมพิวเตอร์ด้วยสาย micro USB จากนั้นไปที่เมนู Project-> Option for Target ‘Lab01_LED Configuration’ แล้วทำการตรวจสอบว่าการตั้งค่าใน Tab ต่างๆ นั้นตรงกับรูปที่ 2.2 - รูปที่ 2.6

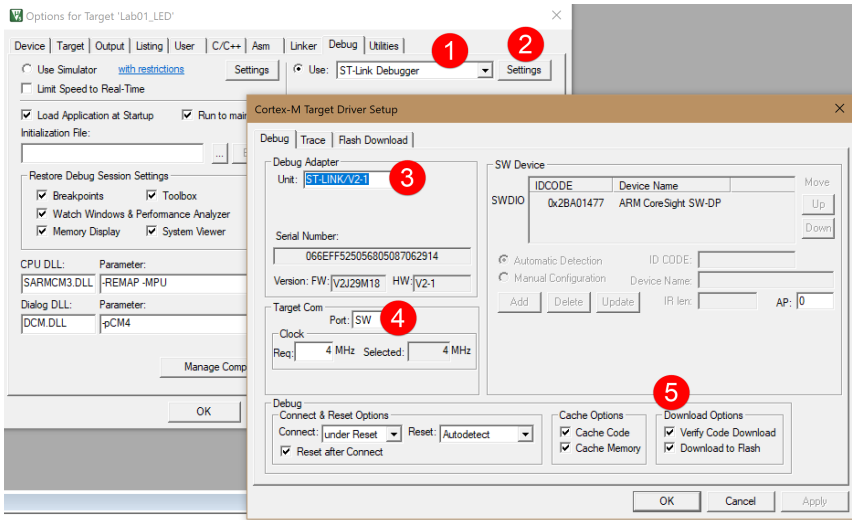


(1) Tab Device เลือก STM32F767ZITx

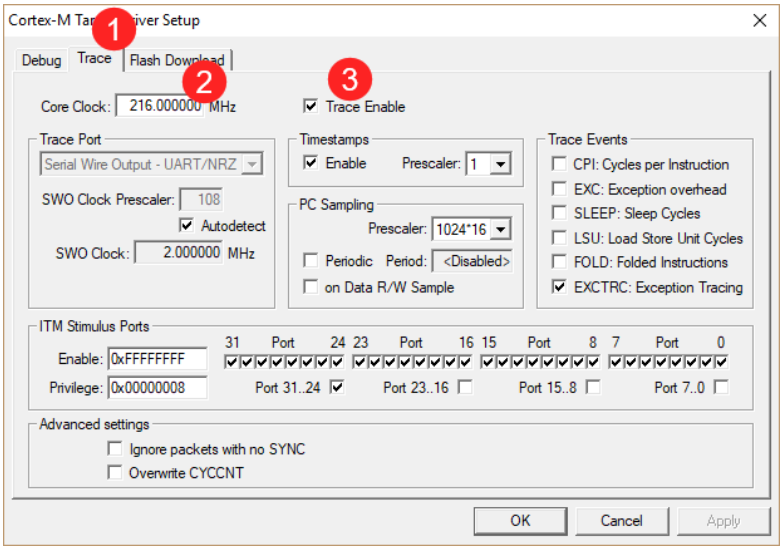


(2) Tab Target กำหนดความถี่ 216 MHz

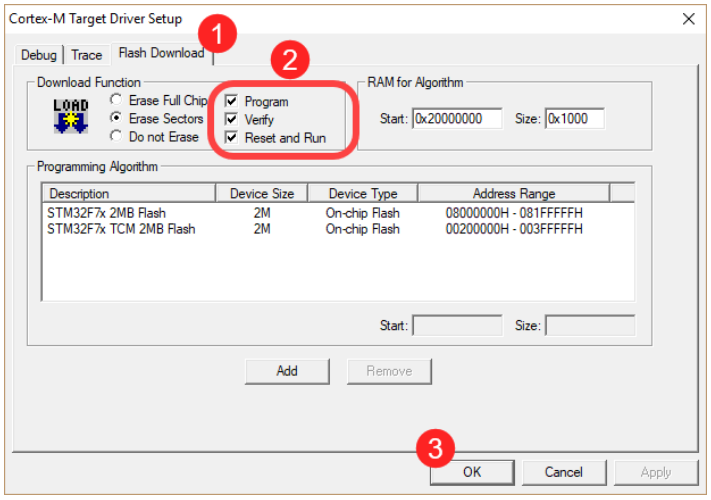
รูปที่ 2.2 การตั้งค่าก่อนการ Build



รูปที่ 2.3 Tab Debug เลือก ST-Link Debugger กด Setting เลือก ST-LINK/V2-1



รูปที่ 2.4 เลือก Tab Trace ระบุ Core Clock ให้เท่ากับความถี่ของ MCU แล้ว Check Trace Enable



รูปที่ 2.5 เลือก Tab Flash Download แล้ว Check Reset and Run

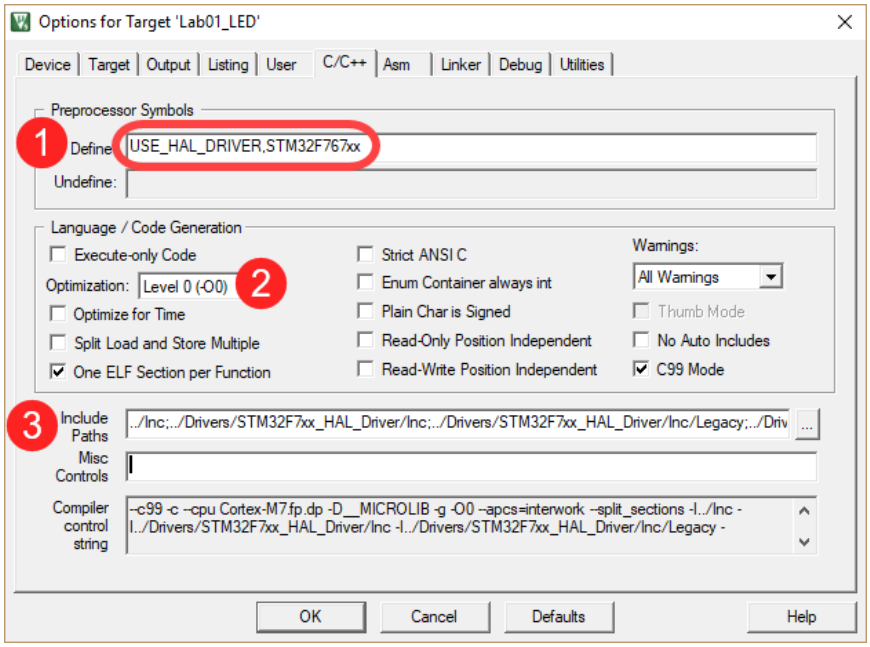
2.3 การตั้งค่าให้กับคอลไพเลอร์

ที่ Tab C/C++ จะเป็นส่วนที่ใช้ตั้งค่าต่างๆ ให้กับคอลไพเลอร์ ดังรูปที่ 2.6 โดย

- หมายเลข 1 เป็นการกำหนด Preprocessor Directives ที่จำเป็นให้กับคอมไพเลอร์ทราบ เนื่องจากไมโครคอนโทรลเลอร์แต่ละแบบมีโครงสร้างภายในแตกต่างๆ กัน จึงต้องระบุรายละเอียดให้กับคอมไพเลอร์เพื่อให้สามารถคอมไฟล์โปรแกรมที่สร้างขึ้นได้ รายละเอียดของไดเรคทีฟแสดงดังตารางที่ 2.3

ตารางที่ 2.3 รายละเอียดของ Directives ที่ใช้

ไดเรคทีฟ	รายละเอียด
USE_HAL_DRIVER	เพื่อให้คอมไพเลอร์รู้ว่ามีการใช้ไลบรารี่ STM32CubeF7 HAL
STM32F767xx	เพื่อให้คอมไพเลอร์รู้ว่าไมโครคอนโทรลเลอร์ที่ใช้อยู่ในตระกูล STM32F767



รูปที่ 2.6 การตั้งค่าให้กับคอมไพเลอร์

- หมายเลข 2 เป็นการกำหนดระดับ Optimization อ้างอิงจาก Keil และ ARMCC รายละเอียดเป็นดังนี้
  - Default: ใช้ค่า default ของคอมไพเลอร์ หรือค่าที่ถูกกำหนดจาก Target หรือ Group ระดับบน
  - Level 0 (-O0): Minimize Optimization เหมาะสำหรับกับการคอมไพล์เพื่อดีบัก (Debug)
  - Level 1 (-O1): Restricted Optimization
  - Level 2 (-O2): High optimization (default level)
  - Level 3 (-O3): Maximum optimization

  สำหรับการทดลองนี้ให้เลือก Optimization Level 0
- หมายเลข 3 ใช้ระบุพาธ (Path) ของไลบรารี่ต่างๆ ซึ่งสามารถกำหนดเพิ่มเติมได้เมื่อใช้ไลบรารี่เพิ่มเติม

2.4 การเพิ่มโค้ดในฟังก์ชัน main

ให้เพิ่มโค้ดลงในฟังก์ชัน main ใน while loop ดังรูปที่ 2.7 ซึ่งจะทำให้ User LED LD1 กระพริบ (ติด 500 ms ดับ 500 ms)

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

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  //Toggle User LED LD1 on PB0 pin
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);

  //Delay 500 ms
  HAL_Delay(500);

}
/* USER CODE END 3 */
```

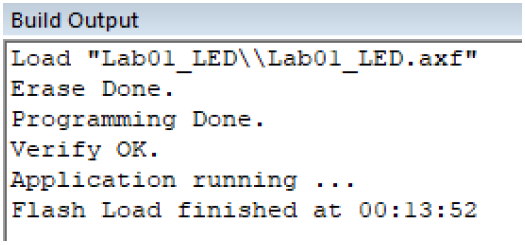
รูปที่ 2.7 ส่วนของโปรแกรมในฟังก์ชัน main

2.5 การ Build Target

ขั้นตอนสุดท้ายคือการ Build Target เพื่อสร้างไฟล์สำหรับโปรแกรมลงบอร์ด โดยไปที่เมนู Project -> Build Target หรือกด F7 จากนั้นตรวจสอบหน้าต่าง Build Output ด้านล่างว่า Build สำเร็จหรือไม่ ดังรูปที่ 2.8 หากไม่สำเร็จให้ศึกษาข้อความแจ้งเตือนแล้วกลับไปแก้ไขโปรแกรมให้ถูกต้อง แล้วทำการ Flash ลงบอร์ด โดยไปที่เมนู Flash -> Download หรือกด F8 สังเกตหน้าต่าง Build Output ด้านล่างว่า Flash สำเร็จหรือไม่ ดังรูปที่ 2.9 หากสำเร็จบอร์ดจะเริ่มต้นทำงานที่ฟังก์ชัน main โดยอัตโนมัติตามที่ได้ตั้งค่าไว้

```
Build Output
compiling stm32f4xx_hal_pwr_ex.c...
compiling system_stm32f4xx.c...
linking...
Program Size: Code=3552 RO-data=464 RW-data=16 ZI-data=1024
"Lab01_LED\Lab01_LED.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:31
```

รูปที่ 2.8 แสดงการ Build สำเร็จ โดยไม่มีข้อผิดพลาดเกิดขึ้น



รูปที่ 2.9 แสดงการ Flash สำเร็จ