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

วัตถุประสงค์

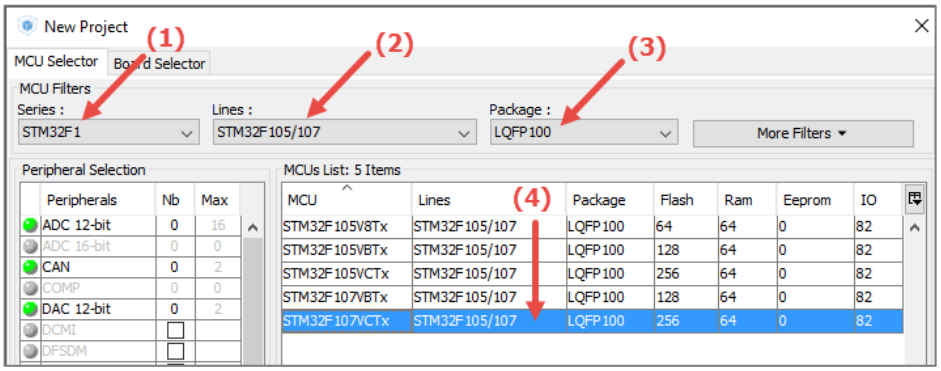
1) สามารถติดตั้งโปรแกรมที่เกี่ยวข้องการการปฏิบัติการ
2) สามารถพัฒนาโปรแกรมโดยใช้ซอฟต์แวร์ที่เกี่ยวข้องได้
3) สามารถดาวน์โหลดโปรแกรมที่พัฒนาลงบอร์ดไมโครคอนโทรลเลอร์ได้

1. การติดตั้งและใช้งานโปรแกรม STM32CubeMX

การพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์ (Microcontroller Unit; MCU) ตระกูล STM32F10X นั้น ขั้นแรกต้องกำหนดหน้าที่ให้กับขาของไอซีไมโครคอนโทรลเลอร์ให้ถูกต้อง เนื่องจากแต่ละขาสามารถทำหน้าที่ได้สูงสุด 3 แบบ กลายเป็นความยากลำบากในการเริ่มต้นพัฒนาเนื่องจากต้องศึกษาเอกสารทางเทคนิคจึงจะสามารถเลือกและกำหนดหน้าที่การทำงานได้อย่างถูกต้อง ดังนั้นทางบริษัทผู้ผลิตไมโครคอนโทรลเลอร์จึงได้พัฒนาโปรแกรมเพื่ออำนวยความความสะดวกในการกำหนดหน้าที่ (Configure) และเริ่มต้นการทำงาน (Initial) ให้กับขาไมโครคอนโทรลเลอร์ที่ผู้พัฒนาต้องการเลือกใช้ ได้แก่โปรแกรม STM32CubeMX

1.1 เลือกไมโครคอนโทรลเลอร์

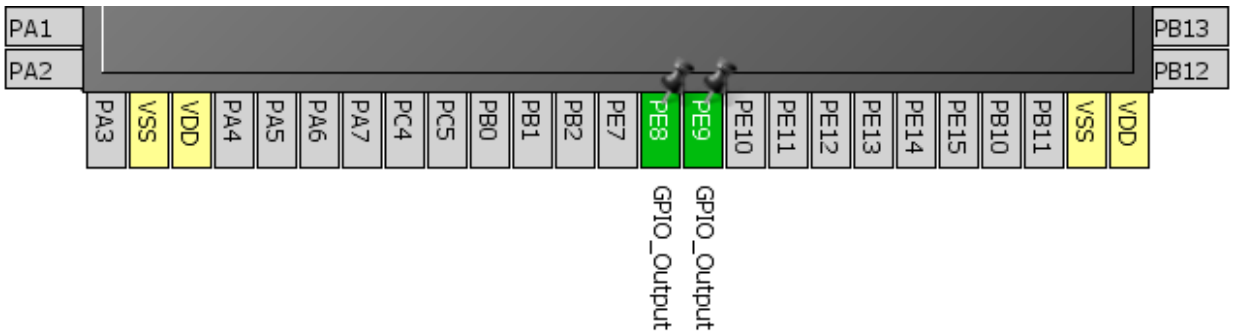
ให้ติดตั้งโปรแกรม STM32CubeMX โดยสามารถดาวน์โหลดได้จากเว็บไซต์ของบริษัท ST หรือเข้าไปยัง "`GDrive\STM32CubeMX\en.stm32cubemx.zip`" หลังจากติดตั้งแล้วให้เปิดโปรแกรมขึ้นมาแล้วเลือก New Project จะพบกับหน้าต่างดัง รูปที่ 1.1 จากนั้นเลือกหมายเลขไมโครคอนโทรลเลอร์ให้ตรงกับการใช้งาน สำหรับบอร์ดทดลองที่ใช้ในวิชาใช้ไมโครคอนโทรลเลอร์หมายเลข STM32F107VCT6 แล้วกดปุ่ม OK



รูปที่ 1.1 หน้าต่าง New Project

1.2 เลือกขาที่ต้องการใช้งาน

กำหนดให้ขา PE8 และ PE9 ซึ่งเชื่อมต่ออยู่กับ LED0 และ LED1 บนบอร์ดให้ทำหน้าที่เป็นขาเอาต์พุตโดยคลิกซ้ายที่ขาดังกล่าวแล้วเลือก GPIO_OUT ดังรูปที่ 1.2



รูปที่ 1.2 กำหนดขา PE8 และ PE9 ให้ทำหน้าที่ GPIO_OUT

1.3 กำหนดสัญญาณนาฬิกา

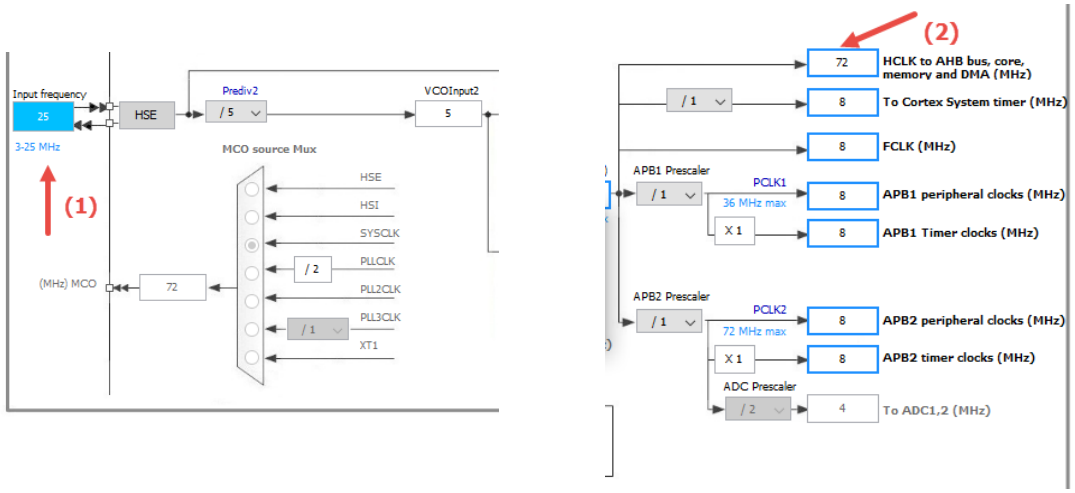
สามารถกำหนดสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์ได้จาก 3 แหล่งจ่ายทั้งภายในไมโครคอนโทรลเลอร์หรือ Crystal Oscillator 25 MHz ที่ติดตั้งอยู่บนบอร์ด โดยที่หน้าต่างด้านซ้ายภายใต้ Peripheral เลือก RCC พร้อมกำหนดค่าดังรูปที่ 1.3 (1) สังเกตว่าทางด้านซ้ายของไมโครคอนโทรลเลอร์จะปรากฏสัญญาณ RCC_OSC_IN และ RCC_OSC_OUT โดยอัตโนมัติ

(1) (2)

รูปที่ 1.3 กำหนดแหล่งจ่ายสัญญาณภายนอกให้กับไมโครคอนโทรลเลอร์

จากนั้นไปที่ Tab Clock Configuration แล้วกำหนด Input frequency ให้เป็น 25 MHz เท่ากับความถี่ของ Oscillator ที่ใช้ จากนั้นกำหนดความถี่ในการทำงานของไมโครคอนโทรลเลอร์ให้เป็น 72 MHz ซึ่งเป็นความถี่สูงสุดที่รับได้

ดังรูปที่ 1.4 แล้วกดปุ่ม OK บนไดอะล็อคที่ปรากฏ จากนั้นโปรแกรมจะกำหนดค่าที่เหมาะสมให้กับวงจรหารและคูณความถี่ภายในให้สอดคล้องกับความถี่ของ Oscillator และความถี่ในการทำงานของไมโครคอนโทรลเลอร์และโมดูลอื่นๆ

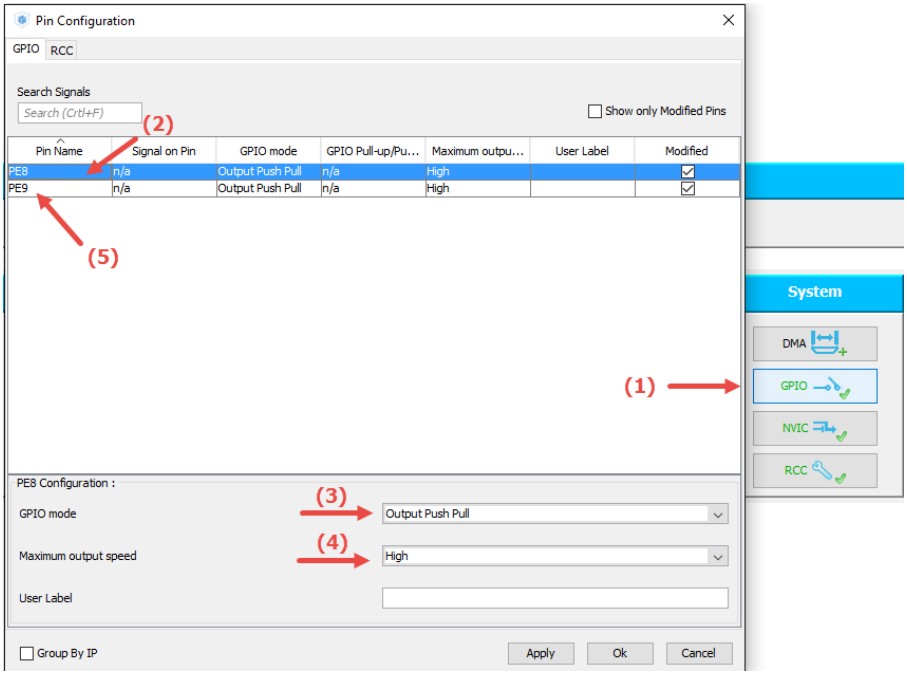


(1) (2)

รูปที่ 1.4 การกำหนดความถี่ของสัญญาณนาฬิกา

1.4 กำหนดรายละเอียดของโมดูลที่เลือกใช้

ไปที่ Tab Configuration ที่หน้าต่างขวามือเลือก GPIO แล้วจะพบกับไดอะล็อคให้กำหนดรายละเอียดดังรูปที่ 1.5 ให้กำหนดรายละเอียด GPIO mode, Maximum output speed ให้กับ PE8 และ PE9 ดังรูปที่ 1.5



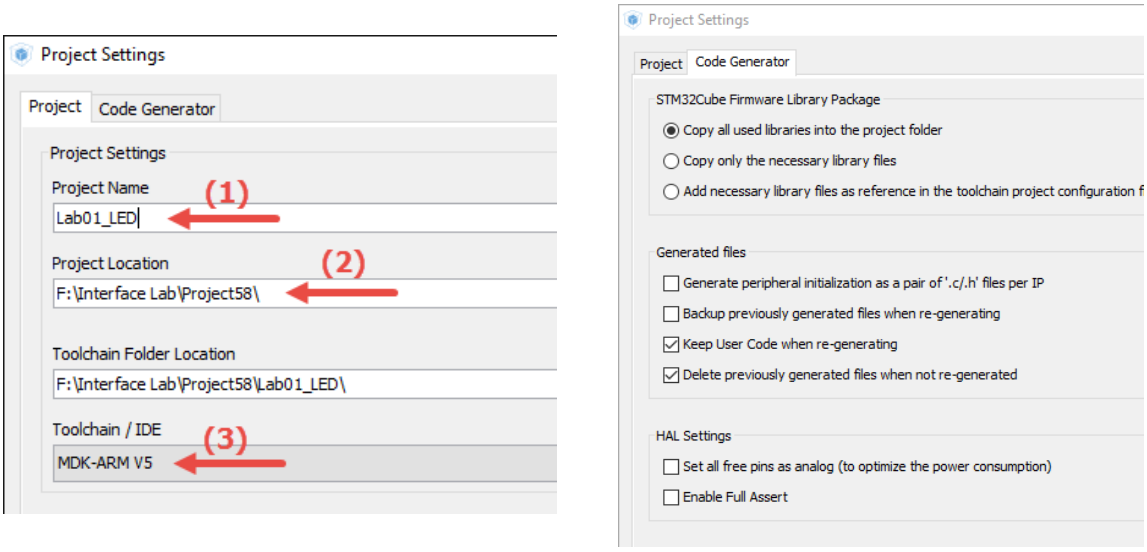
รูปที่ 1.5 กำหนาดรายเอียดของ GPIO

1.5 ติดตั้ง Firmware

สามารถข้ามขั้นตอนข้อนี้ไปได้โดยเมื่อข้ามไปทำขั้นตอน 1.6 โปรแกรมจะดาวน์โหลด Firmware ที่เกี่ยวข้องให้โดยอัตโนมัติ แต่เพื่อความสะดวกสามารถติดตั้ง Firmware ด้วยตนเองได้ที่เมนู Help -> Install new libraries ที่มุมล่างซ้ายกดปุ่ม “`From Local`” เลือกไฟล์ `en.stm32cubef1.zip` จากโฟลเดอร์ “`GDrive\STM32CubeF1`”

1.6 Generating Code

จากหลังที่เลือกใช้และกำหนดรายละเอียดให้กับโมดูลที่เลือกใช้งานแล้ว ขั้นตอนต่อไปจะเป็นการ generate code เพื่อนำไปพัฒนาต่อในโปรแกรม Keil โดยไปที่เมนู Project -> Settings แล้วกำหนดชื่อ Project เป็น Lab01 แล้วเลือกโฟลเดอร์ตามที่ต้องการ และเลือกประเภทของ IDE ให้เป็น MDK-ARM V5 ดังรูปที่ 1.6 (1) และที่ Tab Code Generator กำหนดตัวเลือกตรงกับรูปที่ 1.6 (2) แล้วเลือกเมนู Project -> Generate Code เพื่อ Generate แล้วกดปุ่ม Save the current project เพื่อบันทึกข้อมูล

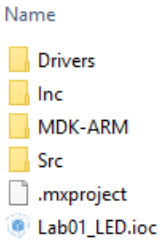


(1) (2)

รูปที่ 1.6 การตั้งค่าสำหรับการ Generate

2. การใช้งานโปรแกรม Keil

สามารถติดตั้งโปรแกรม Keil โดยศึกษาจากเอกสารเพิ่มเติม 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

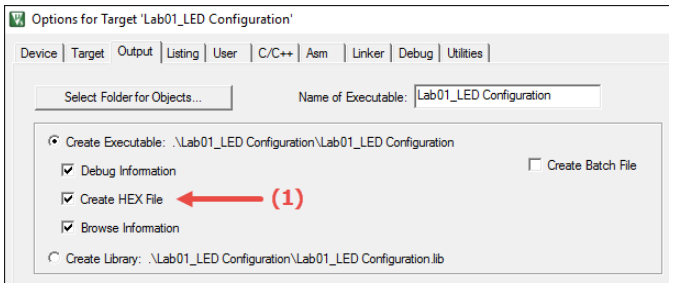
เปิด Project โดยเข้าไปที่โฟลเดอร์ MDK-ARM แล้วเปิดไฟล์นามสกุล .uvprojx จะพบกับหน้าต่าง Project ที่แสดงโครงสร้างโฟลเดอร์และไฟล์ใน Project จากนั้นให้เปิดไฟล์ main.c ซึ่งมีฟังก์ชันที่สำคัญดังตารางที่ 2.2

ตารางที่ 2.2 รายละเอียดของฟังก์ชันในไฟล์ main.c

ฟังก์ชัน	รายละเอียด
`main`	ใช้เขียน User Code
`MX_GPIO`	ตั้งค่าขา PE8 และ PE9 ให้ทำหน้าที่เอาต์พุตตามที่กำหนดใน STM32CubeMX
`SystemClock_Config`	ตั้งค่าสัญญาณนาฬิกาให้กับโมดูลต่างๆ ในไมโครคอนโทรลเลอร์

2.2 การสร้าง Hex File

การโปรแกรมลงบอร์ดต้องใช้ไฟล์นามสกุล .Hex ซึ่งค่าเริ่มต้นของโปรแกรม Keil จะไม่สร้างไฟล์ชนิดนี้มาให้ ต้องตั้งค่าเพิ่มเติมโดยไปที่เมนู Project-> Option for Target จะปรากฏหน้าต่างดังรูปที่ 2.2 โดยให้ไปที่ Tab Output แล้วเลือกให้สร้าง Hex File ดังรูปที่ 2.2



รูปที่ 2.2 การตั้งค่าก่อนการ Build

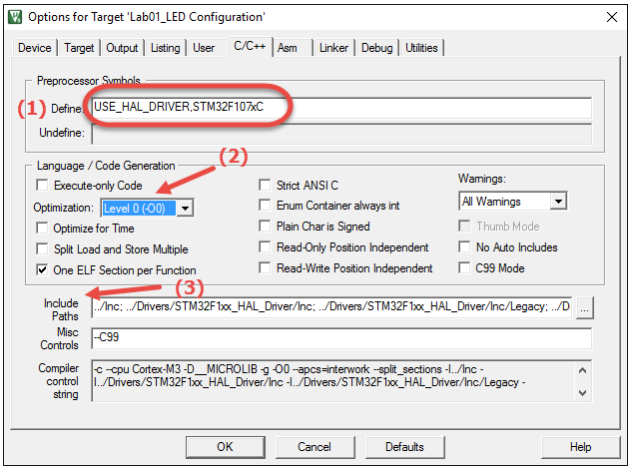
2.3 การตั้งค่าให้กับคอลไพเลอร์

ที่ Tab C/C++ บนหน้าต่างในรูปที่ 2.2 จะเป็นส่วนที่ใช้ตั้งค่าต่างๆ ให้กับคอลไพเลอร์ ดังรูปที่ 2.3 โดย

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

ตารางที่ 2.3 รายละเอียดของ Directives ที่ใช้

ไดเรคทีฟ	รายละเอียด
USE_HAL_DRIVER,	เพื่อให้คอมไพเลอร์รู้ว่ามีการใช้ไลบรารี่ STM32CubeF1
STM32F107xC	เพื่อให้คอมไพเลอร์รู้ว่าไมโครคอนโทรลเลอร์ที่ใช้อยู่ในตระกูล STM32F107xx



รูปที่ 2.3 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
- หมายเลข 3 ใช้ระบุพาธ (Path) ของไลบรารี่ต่างๆ ซึ่งสามารถกำหนดเพิ่มเติมได้เมื่อใช้ไลบรารี่เพิ่มเติม

2.4 การเพิ่มโค้ดในฟังก์ชัน main

ให้เพิ่มโค้ดลงในฟังก์ชัน main ใน while loop ดังรูปที่ 2.4 :ซึ่งจะทำให้ LED0 กระพริบ (ติด 300 ms ดับ 300 ms)

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


/* USER CODE BEGIN 3 */

  HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);
  /* Insert delay 300 ms */
  HAL_Delay(300);

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

รูปที่ 2.4 ส่วนของโปรแกรมในฟังก์ชัน main

2.5 การ Build Target

ขั้นตอนสุดท้ายคือการ Build Target เพื่อสร้างไฟล์สำหรับโปรแกรมลงบอร์ด โดยไปที่เมนู Project -> Build Target หรือกด F7 จากนั้นตรวจสอบหน้าต่าง Build Output ด้านล่างว่า Build สำเร็จหรือไม่ หากไม่สำเร็จให้ศึกษาข้อความแจ้งเตือนแล้วกลับไปแก้ไขโปรแกรมให้ถูกต้อง แล้วตรวจสอบว่ามีไฟล์ .Hex อยู่ในโฟลเดอร์ "`MDK-ARM\Lab01_LED Configuration`" หรือไม่

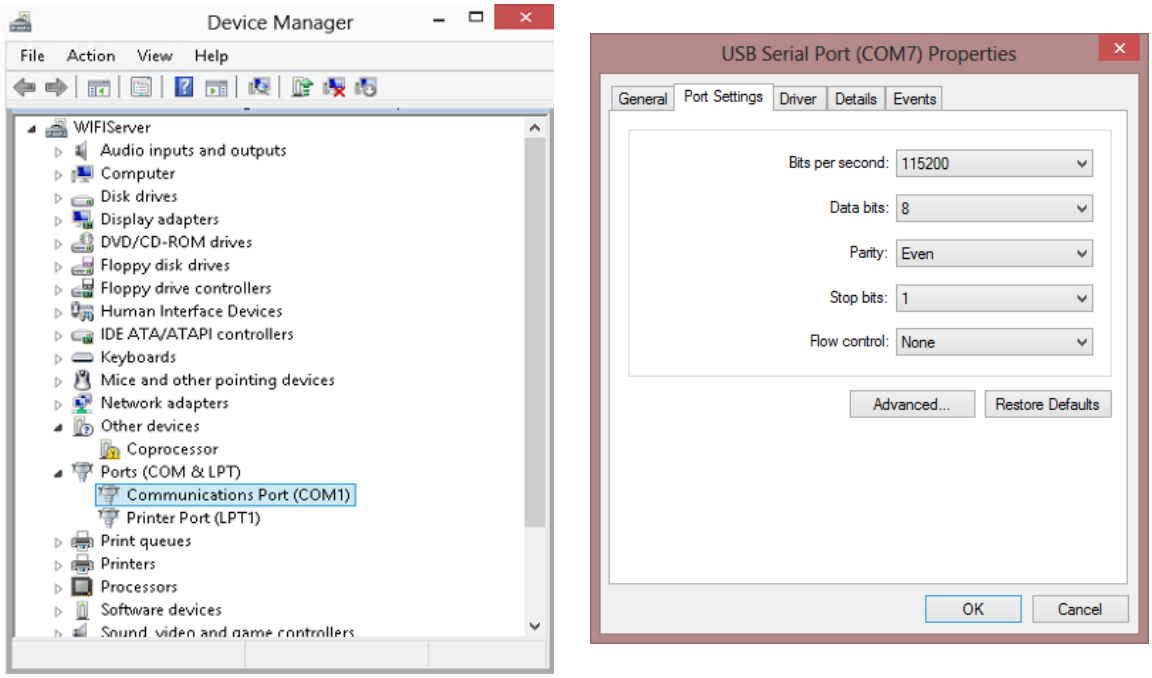
3. การตั้งค่า Serial Port

การโปรแกรมไมโครคอนโทรลเลอร์ต้องดาวน์โหลดไฟล์นามสกุล .hex ที่ได้จากการเขียนโปรแกรมลงบอร์ดผ่านทาง serial port โดยใช้โปรแกรม Flash Loader Demonstrator ดังนั้นจึงต้องมีการตรวจสอบและตั้งค่า serial port ของเครื่องคอมพิวเตอร์ ดังนี้

3.1 ติดตั้ง Driver สายแปลง USB to RS232 ได้จากโฟลเดอร์ "`GDrive\USB2Serial`"

3.2 หากสายแปลง USB to RS232 เสียบอยู่ที่เครื่องเมื่อไปที่ control panel -> device manager จะพบ Communications Port ดังรูปที่ 3.1 (1)

3.3 เลือก Ports -> Communication Port (COM1) โดยคลิกขวาเลือก Properties เลือก Tab Port Settings แล้วตั้งค่าดังรูปที่ 3.1 (2) ในบางเครื่องอาจจะแสดงหมายเลขพอร์ตเป็น COM2, COM3 หรือ COM**x** ก็ให้ใช้ตามนั้นโดยตลอด



(1) (2)

รูปที่ 3.1 (1) Device Manager, (2) การตั้งค่า serial port

4. การดาวน์โหลดไฟล์ HEX ลงบอร์ด MCU

การดาวน์โหลดไฟล์ .hex ให้กับหน่วยความจำ Flash ของ MCU ในบอร์ดนั้น จะใช้โปรแกรมชื่อ Flash Loader Demonstrator ของ “ST Microelectronics” ซึ่งจะติดต่อกับ MCU ผ่าน Serial Port ของเครื่องคอมพิวเตอร์ โดยติดตั้งจากโฟลเดอร์ “`GDrive\Flash Demo`” หรือดาวน์โหลดจาก http://www.st.com

4.1 ต่อสายสัญญาณ RS232 ระหว่างพอร์ตสื่อสารอนุกรม RS232 ของ PC และบอร์ด UART2

4.2 จ่ายไฟเลี้ยงวงจรให้กับบอร์ด ซึ่งจะสังเกตเห็น LED PWR ติดสว่างให้เห็น

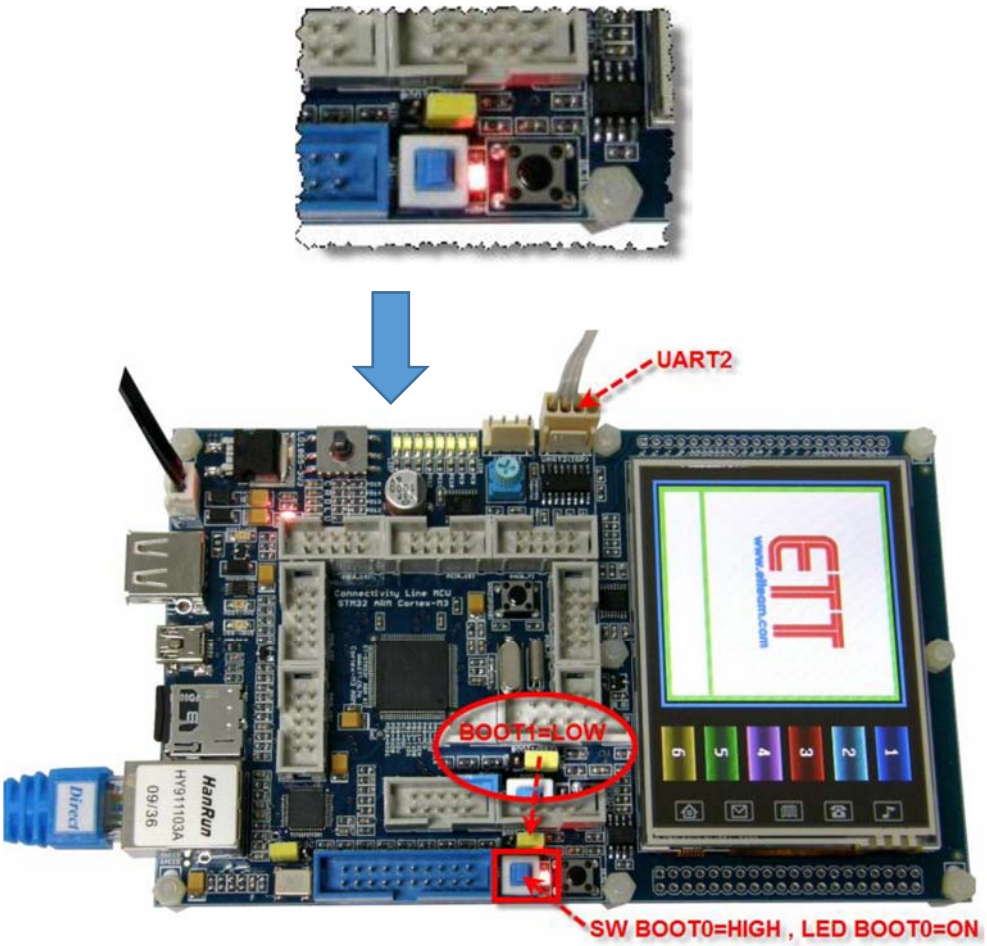
**** ระวัง!!! การต่อไฟเลี้ยงวงจรสลับขั้วจะทำให้บอร์ดเสียหาย ****

*** สังเกตที่ adapter จะมีรูปบอกว่า pin ไหนเป็น + หรือ – ****

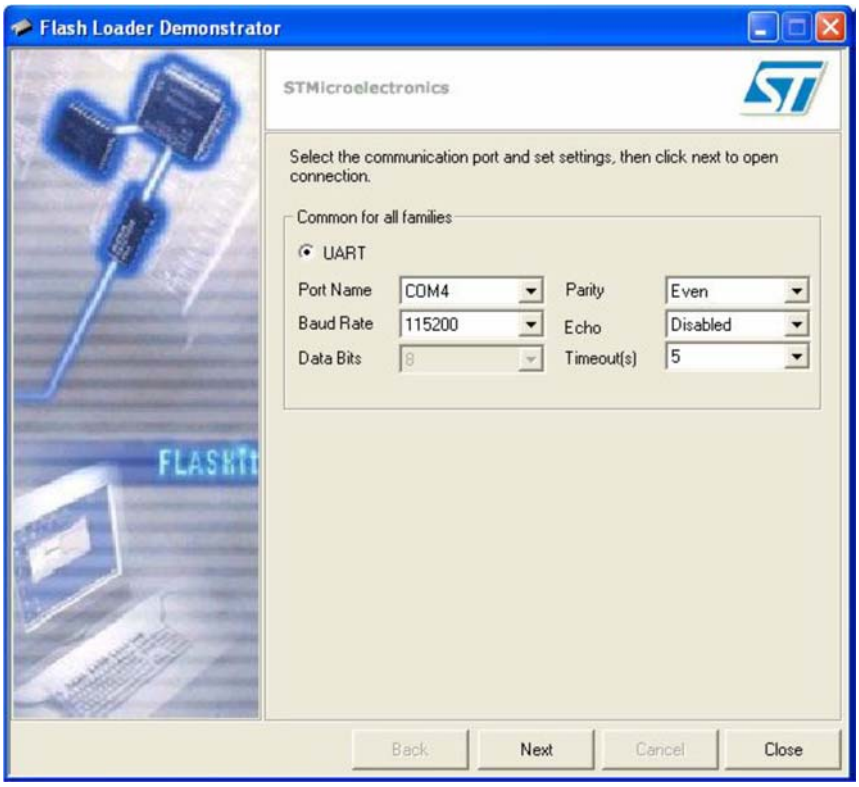
**** และที่ socket บนบอร์ดจะบอก pin 5V เอาไว้ ****

4.3 เลือกกำหนด Jumper BOOT1 ไว้ทางด้าน Low (ปกติเป็น low อยู่แล้วไม่ต้องทำอะไร)

4.4 กดเลือกสวิตช์ BOOT0 เป็น High โดยจะเห็น LED BOOT0 ติดสว่างให้เห็น ดังรูป



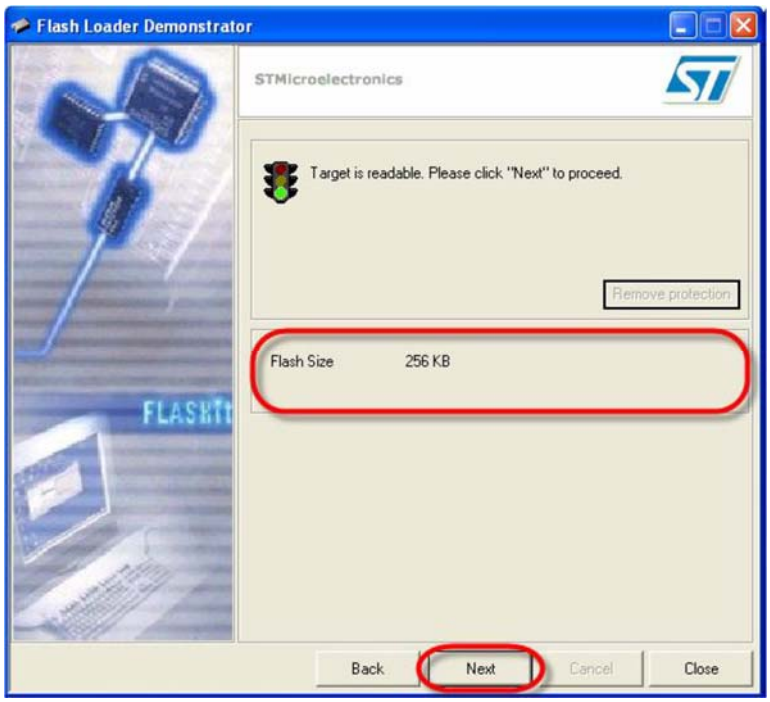
4.5 สั่ง Run โปรแกรม Flash Loader ซึ่งถ้าเป็น Version 2.10 จะได้ผลดังรูป

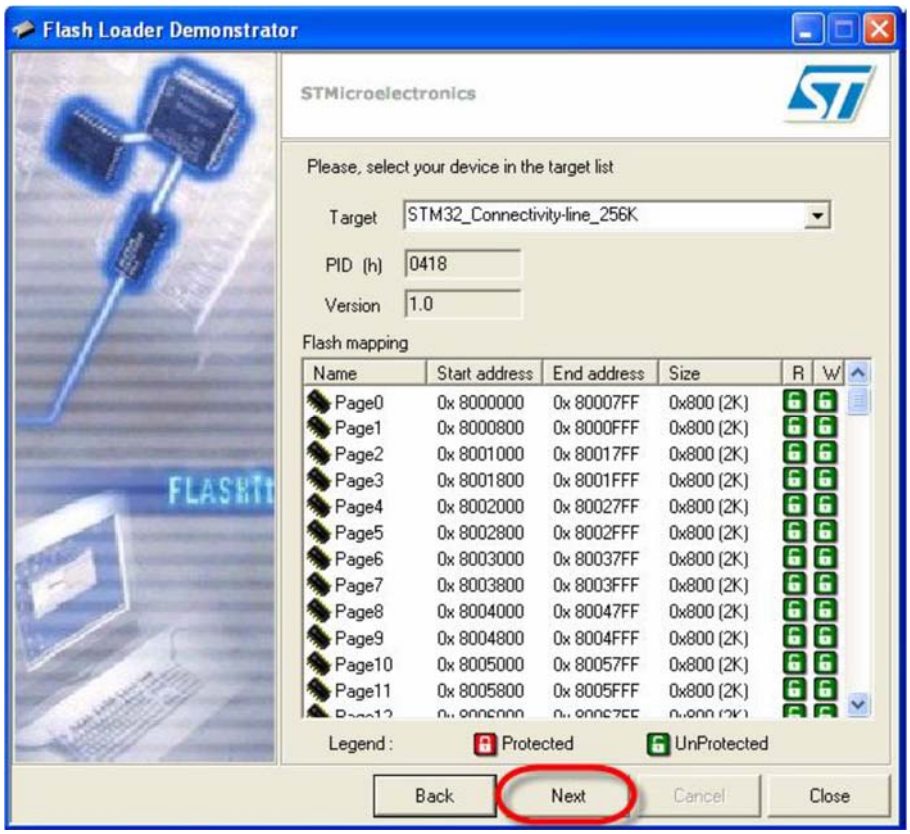


4.6 เริ่มต้นกำหนดค่าตัวเลือกต่างๆให้กับโปรแกรมตามต้องการ ซึ่งในกรณีนี้ใช้กับ STM32F107VCT6 ของบอร์ด ET-STM32F ARM KIT ของ อีทีที ให้เลือกกำหนดค่าต่างๆให้โปรแกรมดังนี้

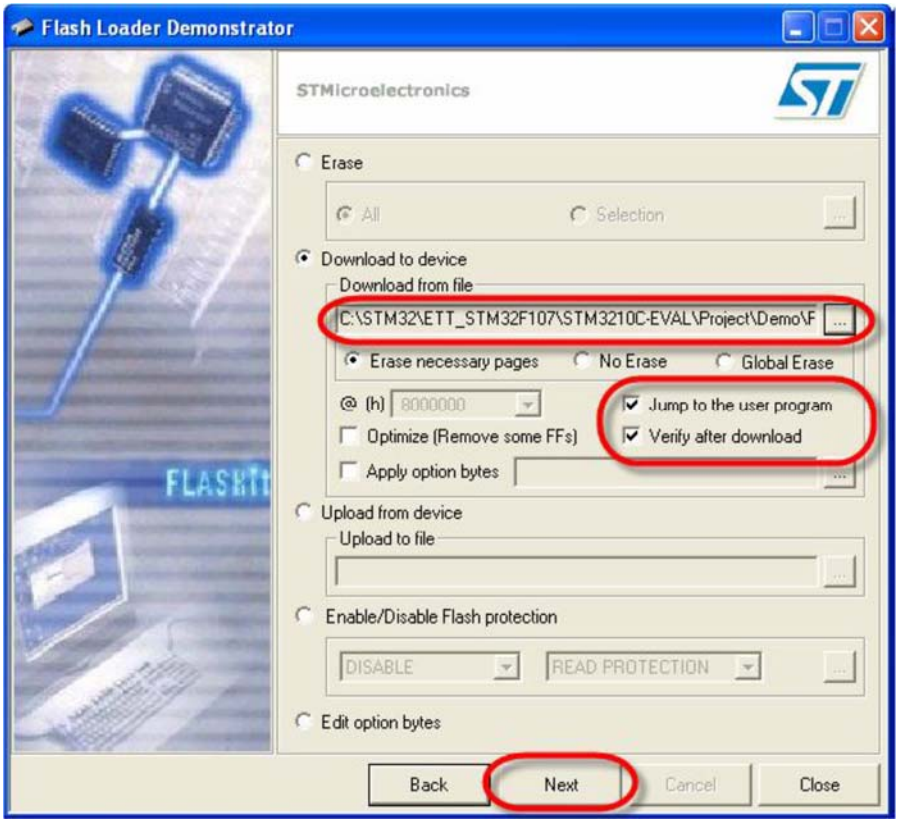
- เลือก COM Port ให้ตรงกับหมายเลข COM Port ที่ใช้งานจริง (ในตัวอย่างใช้ COM4)
- ตั้งค่า Baud Rate เป็น 115200
- เลือก Parity เป็น Even
- เลือก Echo เป็น Disable
- กำหนดค่า Timeout เป็น 5 วินาที
- ให้กดสวิตช์ Reset ที่บอร์ด “ET-STM32F ARM KIT” เพื่อทำการ Reset ให้ MCU ทำงานใน Boot Loader โดยให้ตรวจสอบเงื่อนไขตามขั้นตอนดังต่อไปนี้
    - เลือก Jumper BOOT1 = Low
    - เลือก Switch BOOT0 ไว้ทางด้าน High(LED BOOT0 ติดสว่าง)
    - ต่อสายสัญญาณ RS232 เข้ากับ UART2(ISP) ของบอร์ดให้เรียบร้อย
    - กดสวิตช์ Reset

4.7 ถ้าทุกอย่างถูกต้องจะได้ผลดังรูป จากนั้นให้เลือก Next เพื่อไปยังขั้นตอนต่อไป ถ้าเกิดข้อผิดพลาดให้ลองตรวจสอบเงื่อนไขตามข้อ 4.6 และ กดสวิตช์ Reset ซ้ำใหม่

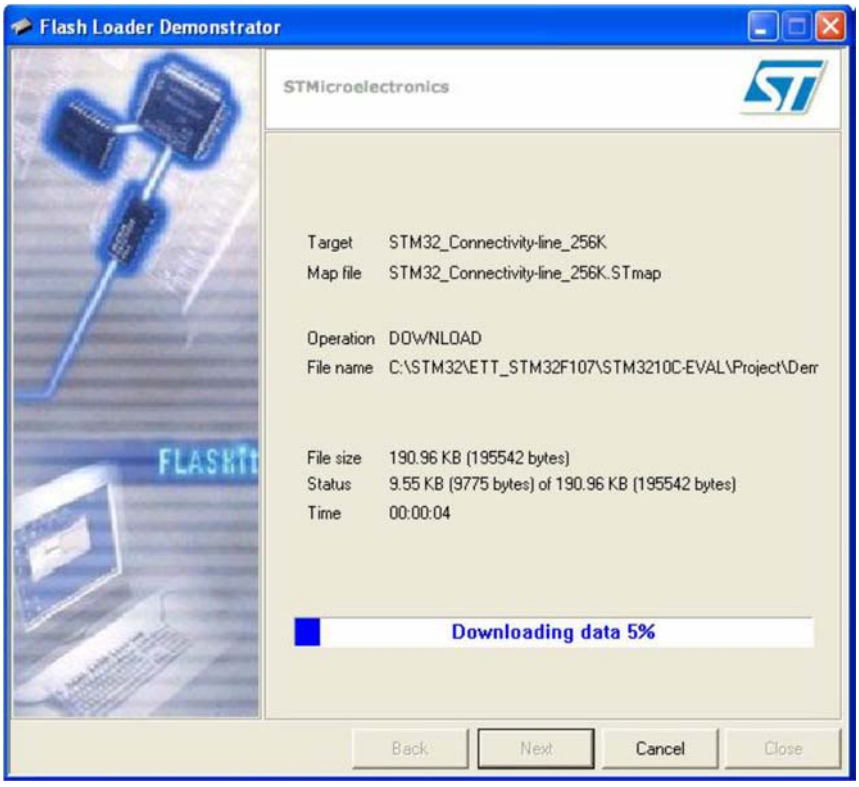


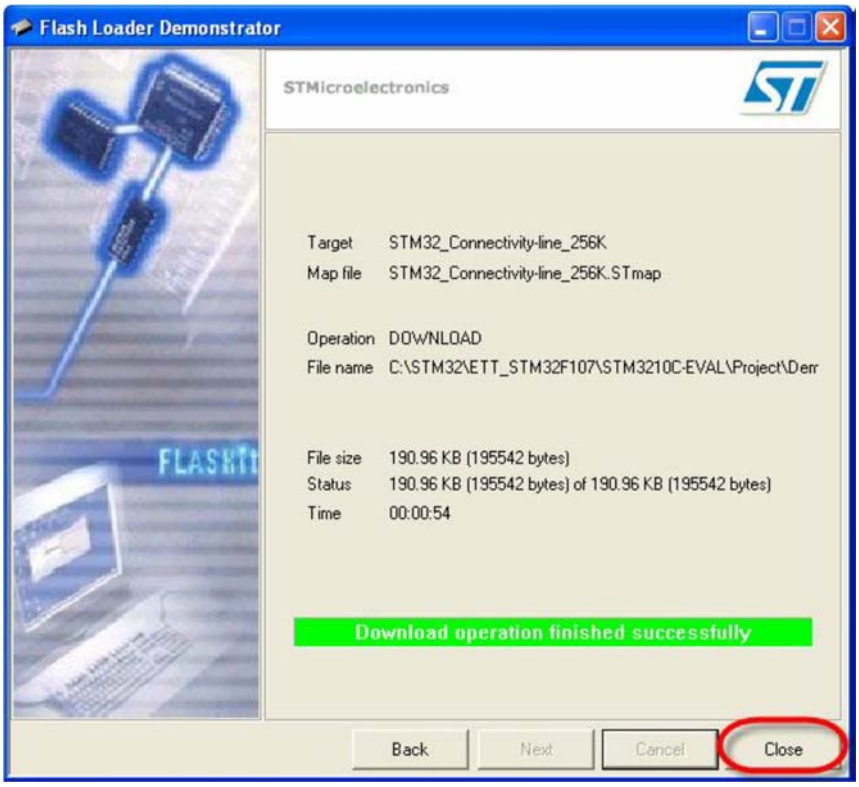


4.8 เลือกกำหนด Hex File ที่ต้องการ Download ให้กับบอร์ด แล้วเลือก Next ดังรูป โดยให้เลือกไฟล์ .hex ที่ได้สร้างขึ้นในข้อ 2.5



4.9 เมื่อโปรแกรม Flash Loader เริ่มต้นทำการ Download ข้อมูลให้กับ MCU ทันที โดยสังเกตการทำงานที่ Status bar โดยในขั้นตอนนี้ให้รอจนกว่าการทำงานของโปรแกรมจะเสร็จสมบูรณ์





4.10 เมื่อทำงานของโปรแกรมเสร็จเรียบร้อยแล้ว ให้กดสวิตช์ BOOT0 กลับมาอยู่ในตำแหน่ง Low โดยจะสังเกตเห็น LED BOOT0 ดับ แล้วจึงกดสวิตช์ Reset ที่บอร์ด ทำให้ MCU เริ่มต้นทำงานตามโปรแกรมที่ Download ลงไป