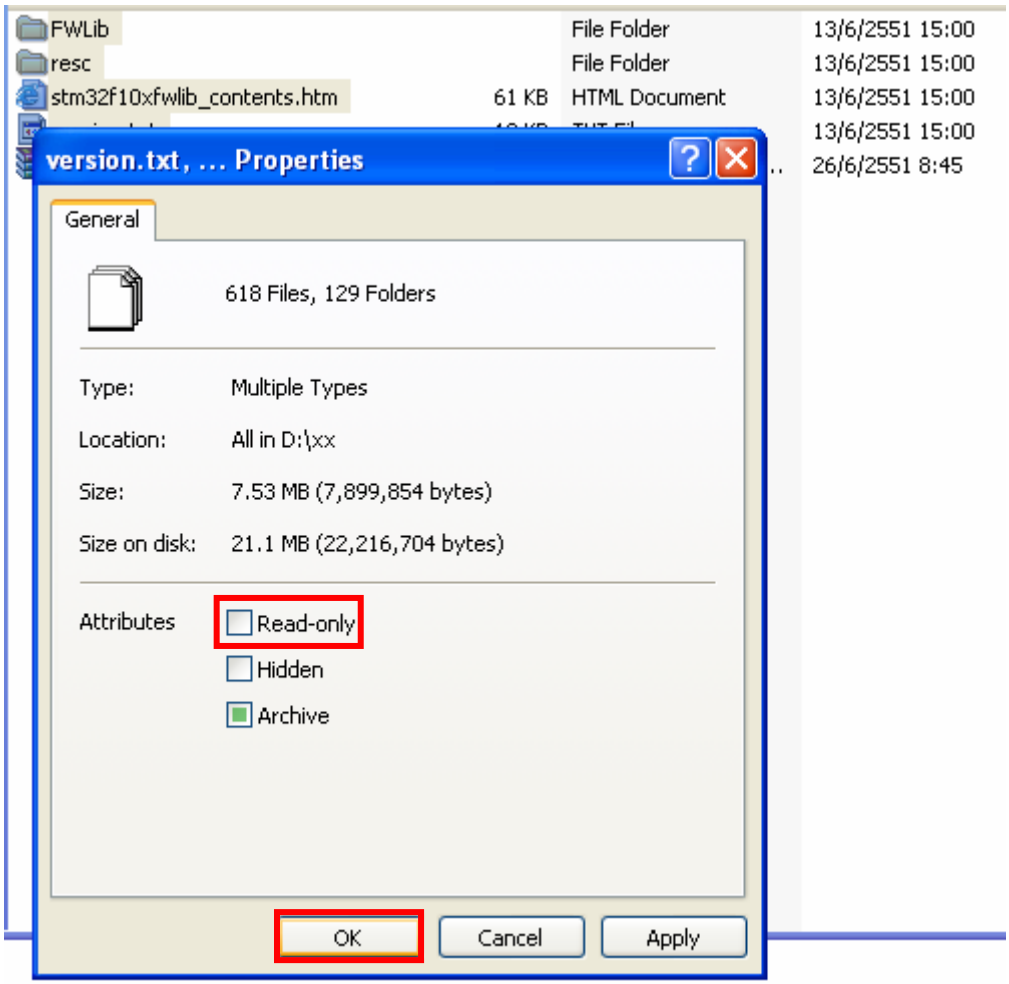
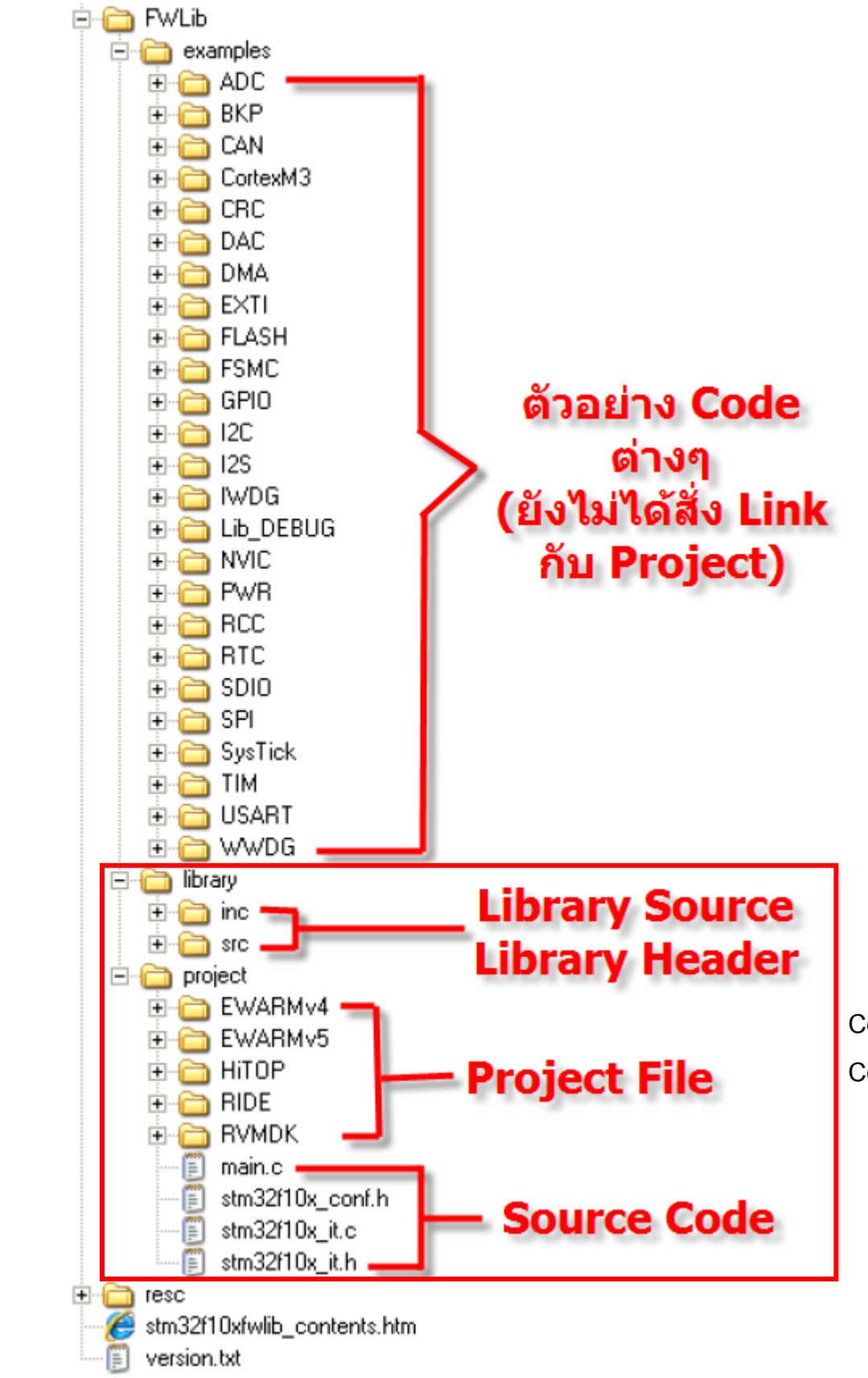
ตัวอย่างการพัฒนาโปรแกรมของบอร์ด ET-STM32F103 ด้วย ST Library

สำหรับแนวทางการพัฒนาโปรแกรมสำหรับใช้งานร่วมกับบอร์ด ET-STM32F103 นั้น สามารถทำได้มากมายหลายวิธี ขึ้นอยู่กับความถนัดของผู้พัฒนา แต่เพื่อความสะดวกและความรวดเร็วในการพัฒนา ขอแนะนำให้เลือกใช้การพัฒนาโปรแกรม ร่วมกับ Firmware Library ของ “STmicroelectronics” ที่จัดทำไว้ให้ เพราะสามารถใช้งานร่วมกับ Complier ต่างๆได้หลายตัว และที่สำคัญ คือ มีตัวอย่าง และ ฟังก์ชั่น การเข้าถึง ทรัพยากรต่างๆของ MCU ตระกูล STM32 จัดเตรียมไว้ให้เรียกใช้งาน ทำให้สามารถลดจำนวนคำสั่งในการพัฒนาโปรแกรมไปได้มากโดย Firmware Library ของทาง “STmicroelectronics” สามารถ Download มาใช้ได้ฟรีซึ่งในปัจจุบัน (กรกฎาคม 2551) เป็น Version 2.1 โดยเมื่อ Download มาแล้ว จะพบว่าไฟล์ดังกล่าวจะถูกบีบอัดไว้ในรูปของ ZIP File ชื่อ “um0427.zip” เมื่อทำการแตกตัว unzip ออกมาจะพบว่า Library ทั้งหมดจะถูกบรรจุไว้ใน โฟลเดอร์ชื่อ “FWLib” ในอันดับแรกให้ผู้ใช้ทำการแก้ไข Properties ของไฟล์ทั้งหมด โดยใช้ Select All แล้วคลิกขวาที่เมาส์ เลือก Properties แล้วปิดตัวเลือกหน้า “Read-only” เพื่อให้สามารถแก้ไขได้ดังรูป





Code ส่วนที่ผู้ใช้ต้อง Copy ไปทำต้นแบบ Project Code

แสดง โครงสร้างของไฟล์ต่าง ๆใน FWLib ที่แตกตัวมาจากไฟล์ um0427.zip

เมื่อเข้าไปดูภายใน Folder ของ “FWLib” จะพบโฟลเดอร์ย่อย อีก 3 กลุ่มคือ

- Examples ซึ่งบรรจุ Code ตัวอย่างการใช้งานทรัพยากรต่างๆของ STM32 ไว้
- Library ซึ่งบรรจุ Source Code และ Header File ของ Library ทั้งหมดไว้
- Project ซึ่งบรรจุตัวอย่าง Project File สำหรับเลือกใช้งานกับ Complier ต่างๆ ให้ผู้ใช้ได้เลือกใช้ตามความถนัดของแต่ละคน โดยรองรับการใช้งานกับ Complier ถึง 5 รุ่น คือ
    - RVMDK เป็น Project File ที่สร้างไว้สำหรับใช้กับ “Keil Realview MDK-ARM”
    - RIDE เป็น Project File ที่สร้างไว้สำหรับใช้กับ “Raisonance Rkit-ARM for Ride7”
    - HiTOP เป็น Project File ที่สร้างไว้สำหรับใช้กับ “HiTOP”
    - EWARMv4 เป็น Project File ที่สร้างไว้สำหรับใช้กับ “IAR Embedded Workbench ARM V4”
    - EWARMv5 เป็น Project File ที่สร้างไว้สำหรับใช้กับ “IAR Embedded Workbench ARM V5”
    - stm32f10x_conf.h, stm32f10x_it.h, stm32f10x_it.c, main.c เป็น Source Code ต้นแบบสำหรับใช้พัฒนาเพื่อใช้งานตามความต้องการ โดย Source Code ส่วนนี้จะถูกเรียกใช้โดย Project File ของแต่ละ Complier โดยสามารถใช้กับ Complier ได้มากถึง 5 แบบ ตามความต้องการของผู้ใช้ว่าต้องการพัฒนาโปรแกรมด้วย Complier ตัวใด

โดยไม่ว่าผู้ใช้จะเลือกใช้ Complier ตัวใด เมื่อเรียกเปิด Project File ขึ้นมา มันจะทำการเชื่อมโยงไฟล์ของ Library File, Header File และ Source Code ทั้งหมดนี้เข้าด้วยกัน เป็นที่เรียบร้อยแล้ว โดยเงื่อนไขของ Project File ทั้งหมด จะกำหนดไว้เพื่อใช้งานร่วมกับชุดพัฒนาของ “STmicroelectronics” โดยสามารถเลือกได้ 2 รุ่น คือ รุ่น “STM3210B-EVAL” และ “STM3210E-EVAL” ซึ่งถ้าใช้งานกับชุดพัฒนาดังกล่าว ผู้ใช้ก็สามารถเลือก เงื่อนไขการแปลคำสั่งให้ตรงกับรุ่นของชุดพัฒนาได้ทันที แต่ในกรณีที่ใช้งานกับบอร์ด “ET-STM32F103” ของ อีทีที จะต้องทำการปรับแก้เงื่อนไขบางส่วนให้ตรงกับทรัพยากรณ์ของบอร์ด โดยแนะนำให้เลือกเงื่อนไขการแปลคำสั่งตามต้นแบบเดิมของชุดพัฒนา “STM3210B-EVAL” จาก “STmicroelectronice” แล้วเลือกกำหนดเบอร์ MCU ใหม่เป็นเบอร์ STM32F103RBT6 แทน

เพื่อป้องกันการสับสน สำหรับผู้เริ่มต้นใช้งานที่ยังอาจไม่คุ้นเคยกับการใช้งาน Complier ซึ่งอาจเป็นความลำบากในการแก้ไข Project File ทาง อีทีที จึงได้นำ Source Code ต้นแบบของ ST มาทำการปรับแก้เงื่อนไขต่างๆ เพื่อให้สอดคล้องกับทรัพยากรของบอร์ด “ET-STM32F103” จัดเตรียมไว้อำนวยความสะดวกกับผู้ใช้ด้วย โดยไฟล์ดังกล่าวจะบรรจุไว้ในแผ่น CD–ROM โดยไฟล์ที่ได้ทำการปรับแก้เพื่อใช้งานกับบอร์ด “ET-STM32F103” จะถูกบรรจุไว้ภายใต้โฟลเดอร์ “..\ET-STM32F103\ETT EXAMPLES\STM32F10xFWLib_V2.1_For_ET-STM32F103” เวลาจะใช้งานผู้ใช้ก็เพียงแต่ Copy โฟลเดอร์ ชื่อ “library” และ “project” ไปเก็บไว้ยัง Folder ที่จะใช้พัฒนาโปรแกรม จากนั้นก็เลือกเปิด Project File ของ Complier ที่ต้องการขึ้นมาก็เป็นอันพร้อมใช้งานแล้ว โดยสามารถใช้งานกับ Complier ได้ 3 รุ่นคือ

- RVMDK เป็น Project File ที่สร้างไว้สำหรับใช้กับ “Keil Realview MDK-ARM”
- RIDE เป็น Project File ที่สร้างไว้สำหรับใช้กับ “Raisonance Rkit-ARM for Ride7”
- EWARMv5 เป็น Project File ที่สร้างไว้สำหรับใช้กับ “IAR Embedded Workbench ARM V5”

ส่วน Source Code จาก Project ที่ถูกสั่งเปิดขึ้นมาจะเป็น Code ต้นแบบ ที่ทำการสั่งเชื่อมโยง ไฟล์ Library File และ Header File รวมทั้ง Source Code ต้นแบบเตรียมไว้ให้เรียบร้อยแล้ว แต่เงื่อนไขการทำงานต่างๆยังไม่ได้ทำไว้ให้ ผู้ใช้สามารถเพิ่มเติมคำสั่งต่างๆในไฟล์ชื่อ ”main.c” ได้ทันที โดยในไฟล์ “main.c” จะมีลักษณะดังนี้

```
#include "stm32f10x_lib.h"
int main(void)
{
#ifdef DEBUG
  debug();
#endif

  /* Infinite loop */
  while (1)
  {
  }
}


#ifdef  DEBUG
void assert_failed(u8* file, u32 line)
{
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

  /* Infinite loop */
  while (1)
  {
  }
}
#endif
```

แนวทางการแก้ไข Code ต้นแบบ

ส่วนของ Code ต้นแบบ จะมีอยู่ทั้งหมด 4 ไฟล์คือ stm32f10x_conf.h, stm32f10x_it.h, stm32f10x_it.c และ main.c โดยแต่ละไฟล์มีหน้าที่ดังนี้

- **stm32f10x_conf.h** ใช้สำหรับเลือกกำหนดการเรียกใช้ Library เพื่อเข้าถึงทรัพยากร ต่างๆของ STM32 โดยในไฟล์ที่อยู่ใน Source Code ที่ใช้เป็นต้นแบบของ Project นี้ จะสั่งเปิดใช้งาน Library ทั้งหมดเลย ซึ่งข้อดีก็คือ ผู้ใช้สามารถเรียกใช้ฟังก์ชั่นต่างๆได้ทันที แต่ข้อเสียคือ ถ้าใน Code ที่กำลังพัฒนาอยู่ ไม่มีความจำเป็นต้องใช้งานทรัพยากรทั้งหมดแต่อาจใช้เพียงบางส่วน การที่สั่งเรียกใช้ Library ไว้ทั้งหมด จะทำให้ Code มีขนาดโตขึ้นเกินความจำเป็น และยังเสียเวลาในการแปลคำสั่งนานตามไปด้วย โดยผู้ใช้สามารถสั่งปิดการเรียกใช้ Library ในส่วนที่ไม่ต้องการใช้งานได้จากไฟล์นี้ โดยแนะนำให้ใส่เครื่องหมาย // นำหน้าคำสั่งเรียกใช้ Library ที่ไม่ต้องการใช้ออกไป เช่น เมื่อต้องการใช้งาน GPIOB เพื่อทดสอบ LED อย่างเดียว ก็สามารถสั่งปิด Library สำหรับใช้อ้างถึงพอร์ตอื่นๆออกไปดังตัวอย่าง

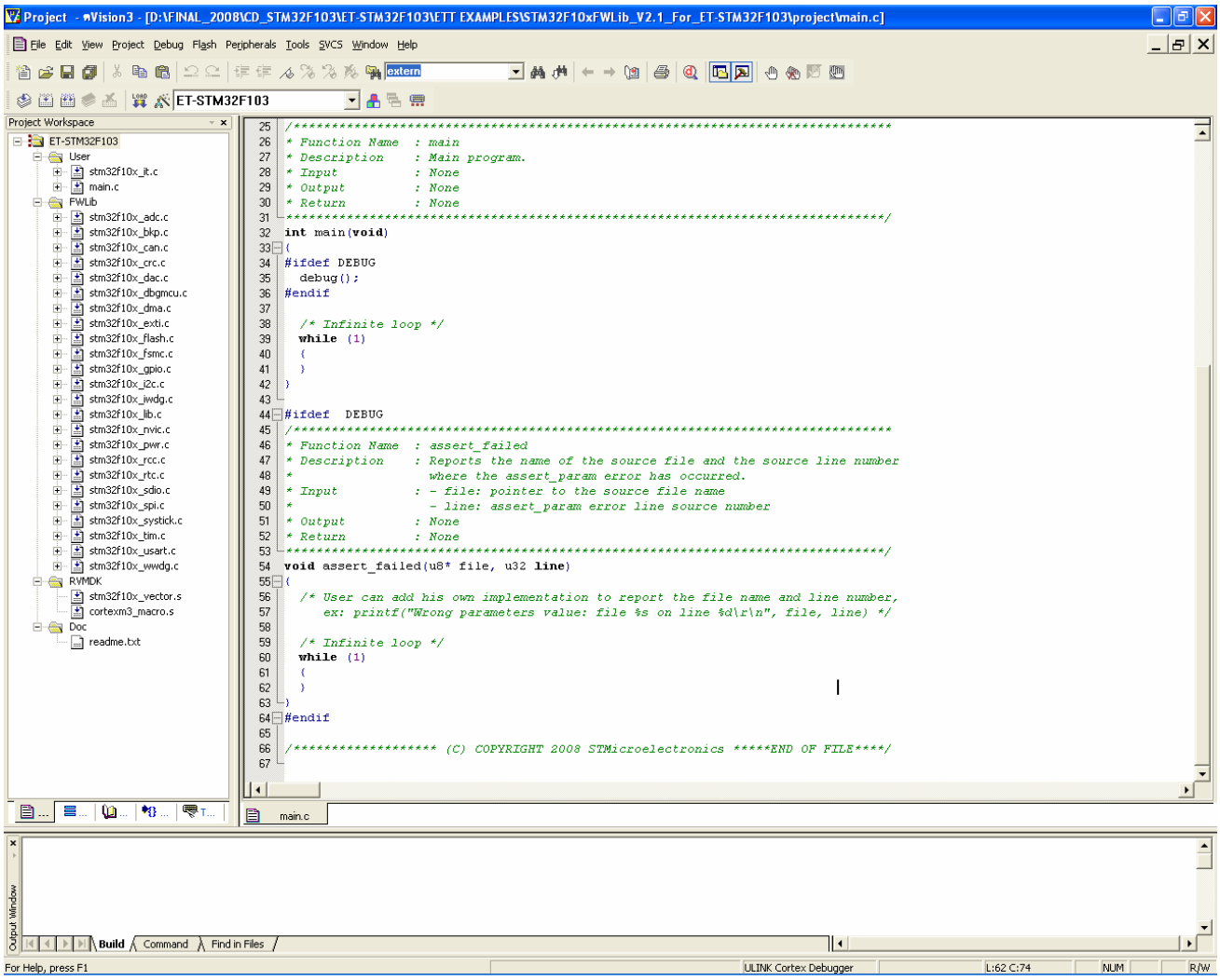
#define _GPIO	**ใช้ Library ของ GPIO**
//#define _GPIOA	ไม่ใช้ Library ของ GPIOA
#define _GPIOB	**ใช้ Library ของ GPIOB**
//#define _GPIOC	ไม่ใช้ Library ของ GPIOC
//#define _GPIOD	ไม่ใช้ Library ของ GPIOD
//#define _GPIOE	ไม่ใช้ Library ของ GPIOE
//#define _GPIOF	ไม่ใช้ Library ของ GPIOF
//#define _GPIOG	ไม่ใช้ Library ของ GPIOG
//#define _AFIO	ไม่ใช้ Library ของ Alternate Function ของ GPIO

- **stm32f10x_it.c** ใช้สำหรับเขียน Function Code สำหรับรองรับการ Interrupt ต่างๆ โดยในต้นแบบได้ประกาศสร้างชื่อของฟังก์ชั่นต่างๆไว้แล้วใน Header ชื่อ **stm32f10x_it.h** ส่วนในไฟล์นี้จะเป็นส่วนของตัวฟังก์ชั่น ที่สร้างเป็นโครงไว้ให้ ผู้ใช้สามารถเขียน Code เพิ่มเติมได้ตามต้องการ
- **main.c** เป็นส่วนของ Code ที่จะเขียนสั่งงาน ซึ่งผู้ใช้สามารถเพิ่มเติมแก้ไขได้ตามความต้องการ

โดยไม่ว่าจะสั่งเปิด Project File สำหรับใช้กับ Complier โปรแกรม Editor ของ Complier ก็จะมาสั่งเปิดไฟล์ Source Code ทั้ง 4 ไฟล์นี้ (stm32f10x_conf.h, stm32f10x_it.h, stm32f10x_it.c, main.c) เหมือนกันหมด การแก้ไขไฟล์ใดไฟล์หนึ่ง หรือทั้งหมด ไม่ว่าจะเป็นการแก้ไขจากภายนอก หรือ จาก Complier ตัวใดตัวหนึ่ง ก็จะส่งผลเปลี่ยนแปลงไปถึง Complier อื่นๆที่ถูกสั่งเปิดใช้งานในลำดับต่อไปด้วยเสมอ

ตัวอย่างการใช้กับ keil Realview MDK-ARM

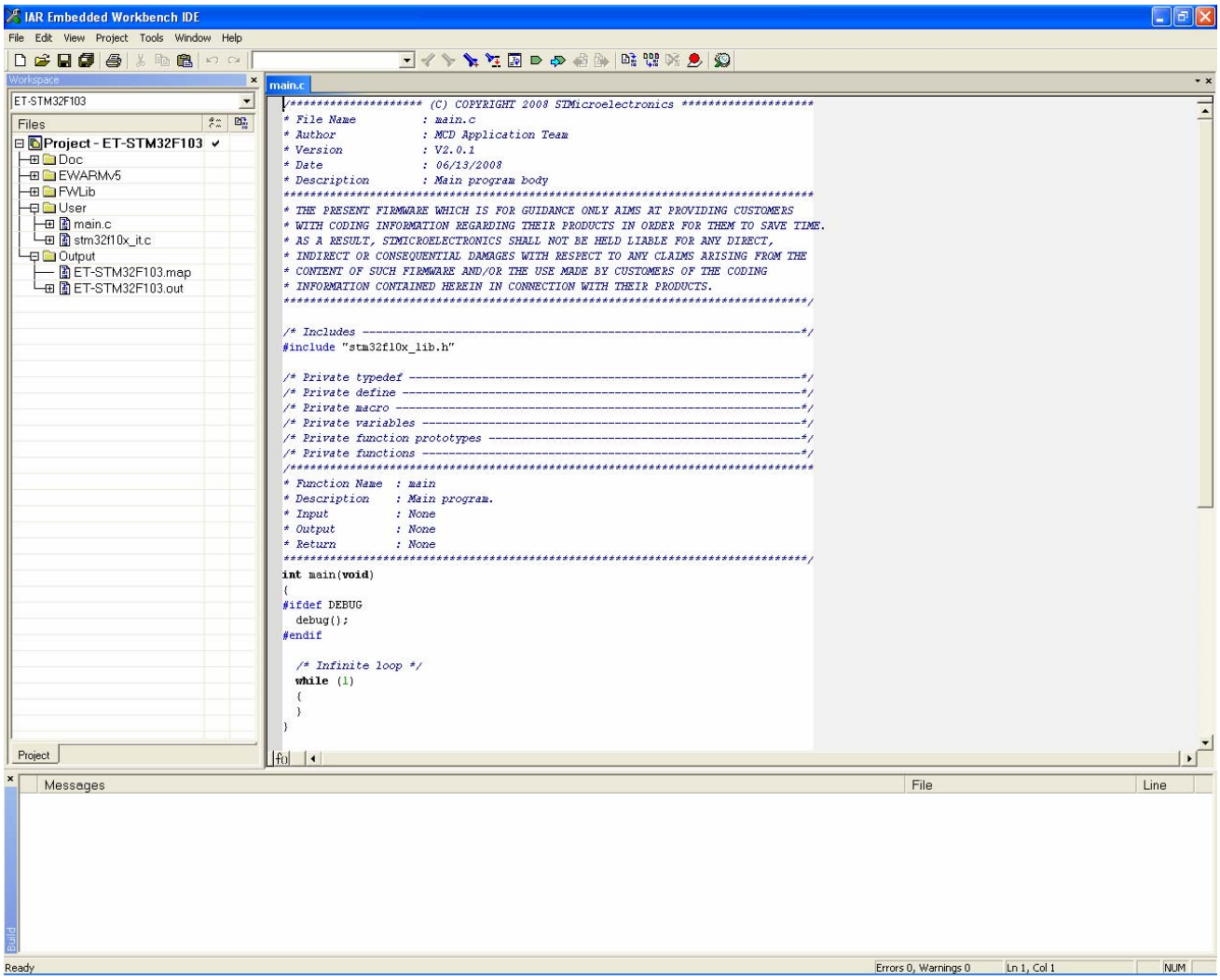
ในกรณีต้องการใช้ Project File ต้นแบบกับ Keil Realview MDK-ARM ก็สั่งเปิด Project File ของ keil ในโฟลเดอร์ "..\project\RVMDK\project.uv2" จะได้ผลดังรูป



- การสั่งแปล ใช้คำสั่ง Project->Rebuild all target files
- การ Load project image ใช้คำสั่ง Debug->Start/Stop Debug Session
- การสั่ง Run ใช้คำสั่ง Debug->Run (F5)

ตัวอย่างการใช้กับ IAR Embedded Workbench ARM V5

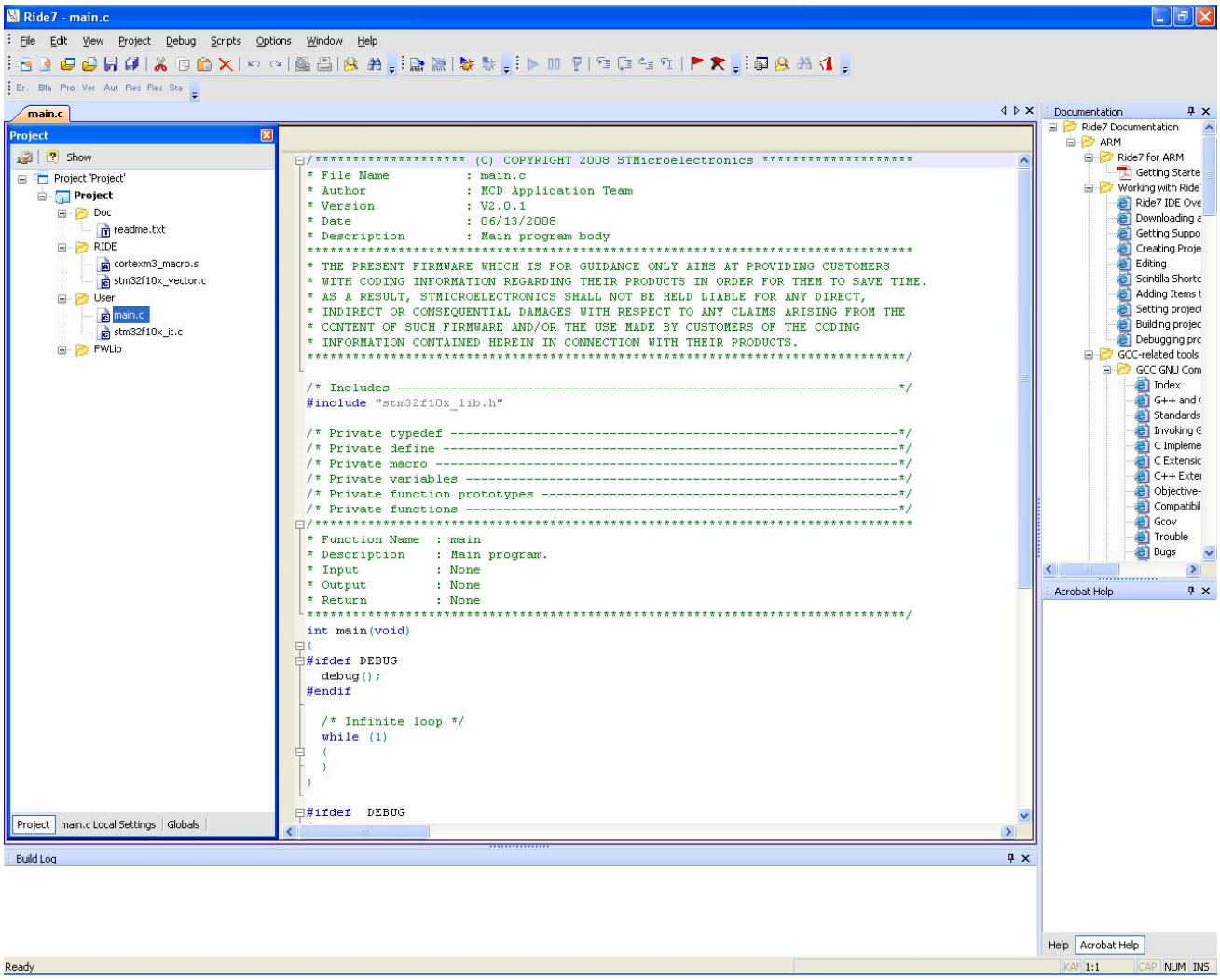
ในกรณีต้องการใช้ Project File ต้นแบบกับ IAR Embedded Workbench ARM V5 ก็สั่งเปิด Project File ของ IAR ในโฟลเดอร์ "..\project\EWARMv5\project.eww" จะได้ผลดังรูป



- การสั่งแปล ใช้คำสั่ง Project->Rebuild all
- การ Load project image ใช้คำสั่ง Project->Debug
- การสั่ง Run ใช้คำสั่ง Debug->Go(F5)

ตัวอย่างการใช้กับ Raisonance Rkit-ARM for Ride7

ในกรณีต้องการใช้ Project File ต้นแบบกับ Raisonance Rkit-ARM for Ride7 ก็สั่งเปิด Project File ของ IAR ในโฟลเดอร์ "..\project\RIDE\project.rprj" จะได้ผลดังรูป



- การสั่งแปล ใช้คำสั่ง Project->build project
- การ Load project image ใช้คำสั่ง Debug->start(ctrl+D)
- การสั่ง Run ใช้คำสั่ง Debug->Run(ctrl+F9)

ตัวอย่างโปรแกรมไฟกระพริบ 1 ดวงที่ PB8

1. ทำการ Copy ไฟล์ต้นแบบจาก CD ใน "..\..\STM32F10xFWLib_V2.1_For_ET-STM32F103\" ซึ่งประกอบด้วยไฟล์จำนวน 2 โฟลเดอร์ คือ "project" และ "library" มาไว้ยังโฟลเดอร์ที่ต้องการสร้าง Project
2. เข้าไปในโฟลเดอร์ "project" แล้วเลือกโฟลเดอร์ของ Project File สำหรับ Complier ที่ต้องการ
3. สั่งเปิด Project ไฟล์ของต้นแบบ Project ที่สร้างไว้แล้ว
4. ทำการเพิ่ม Code ให้กับไฟล์ main.c แล้ว Complier พร้อมกับ Download ทดสอบการทำงาน ดังนี้

```
#include "stm32f10x_lib.h"
GPIO_InitTypeDef GPIO_InitStructure;
ErrorStatus HSEStartUpStatus;

void RCC_Configuration(void);
void NVIC_Configuration(void);
void Delay(vu32 nCount);

int main(void)
{
#ifdef DEBUG
  debug();
#endif

  /* System Clocks Configuration */
  RCC_Configuration();

  /* NVIC Configuration */
  NVIC_Configuration();

  /* Configure PB8..15 = Output */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
  GPIO_InitStructure.GPIO_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_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  while (1)
  {
    /* Turn on LED-PB8 */
    GPIO_SetBits(GPIOB, GPIO_Pin_8);
    Delay(1000000);

    /* Turn off LED-PB8 */
    GPIO_ResetBits(GPIOB, GPIO_Pin_8);
    Delay(1000000);
  }
}
```

```
void RCC_Configuration(void)
{
  RCC_DeInit();
  RCC_HSEConfig(RCC_HSE_ON);
  HSEStartUpStatus = RCC_WaitForHSEStartUp();

  if(HSEStartUpStatus == SUCCESS)
  {
    FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
    FLASH_SetLatency(FLASH_Latency_2);
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
    RCC_PCLK2Config(RCC_HCLK_Div1);
    RCC_PCLK1Config(RCC_HCLK_Div2);
    RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
    RCC_PLLCmd(ENABLE);
    while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)
    {
    }
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
    while(RCC_GetSYSCLKSource() != 0x08)
    {
    }
  }
}
```

```
void NVIC_Configuration(void)
{
#ifdef  VECT_TAB_RAM
  NVIC_SetVectorTable(NVIC_VectTab_RAM, 0x0);
#else  /* VECT_TAB_FLASH  */
  NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x0);
#endif}
```

```
void Delay(vu32 nCount)
{
  for(; nCount != 0; nCount--);
}
```

```
#ifdef  DEBUG
void assert_failed(u8* file, u32 line)
{
  while (1)
  {
  }
}
#endif
```

ตัวอย่าง Code โปรแกรมไฟกระพริบ 1 ดวงที่ PB8