# Creating a New Project in µVision IDE

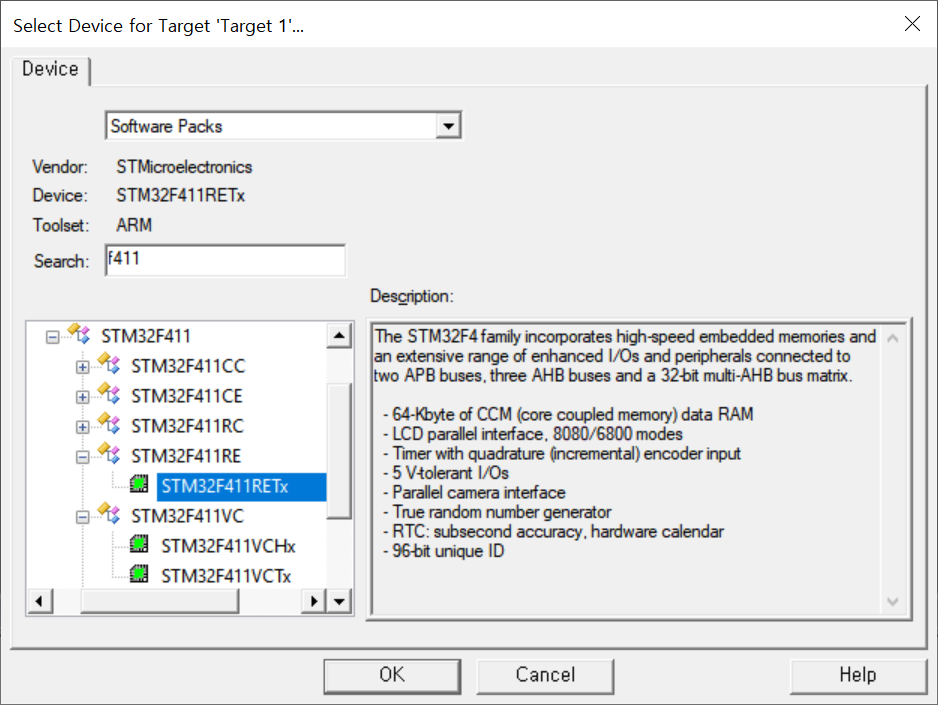
## Steps to create a new project in Keil uVision IDE

1. To create a new project, select Keil window menu **Project** ⟶ **New µVision Project**
2. Give the project a name and select its storage directory. For this tutorial, create a directory

“C:\Keil\_v5\Tutorial\Tutorial0\”

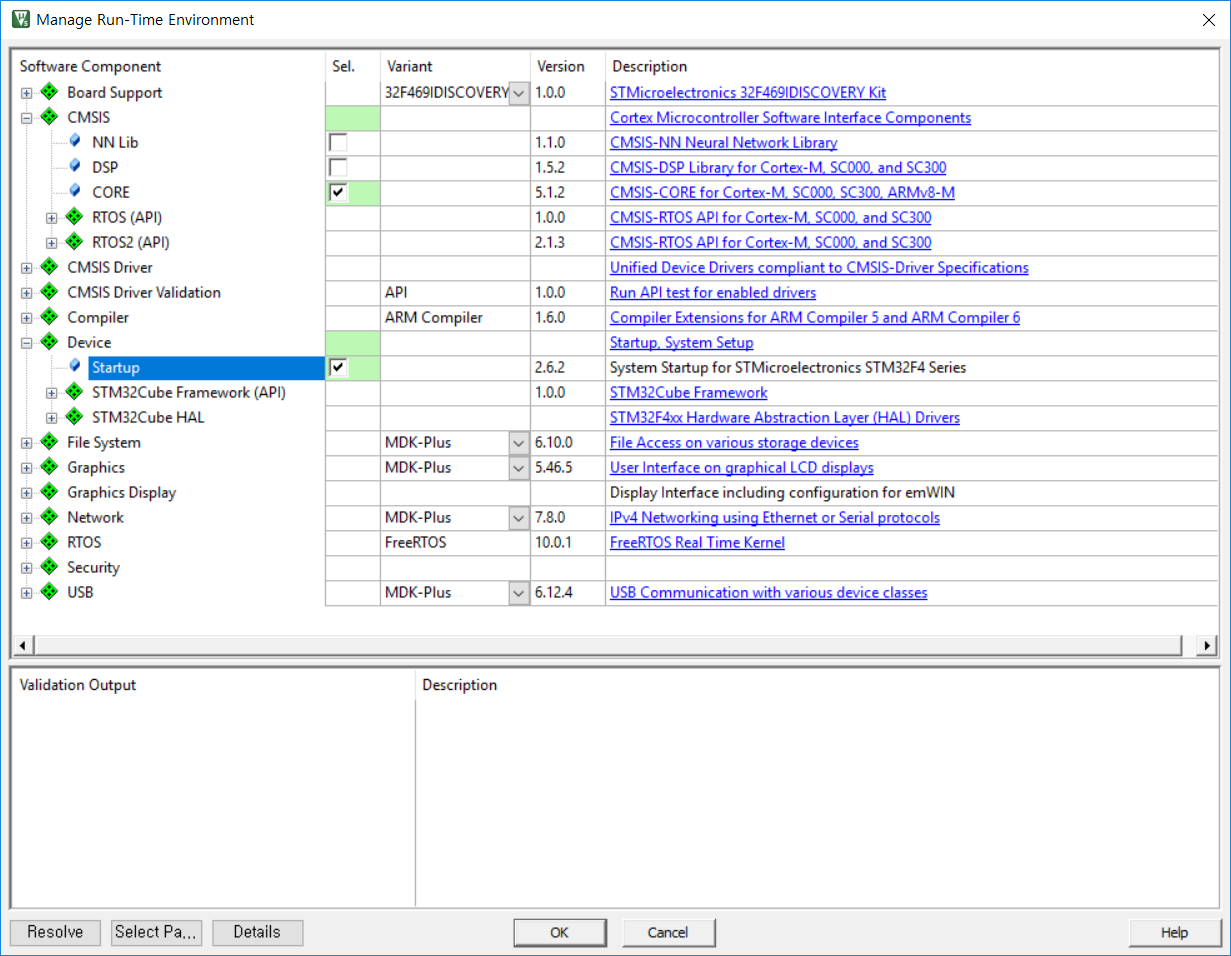
Name the project as “Tutorial0\_ProjectStart”. There should not be any Korean in the directory.

1. On ‘Select Device for Target’ window: Search 🡪 STM32F411RE. Select ‘**STM32F411RETx**’. We will be using an evaluation board of NUCLEO STM32F411RE for this course.

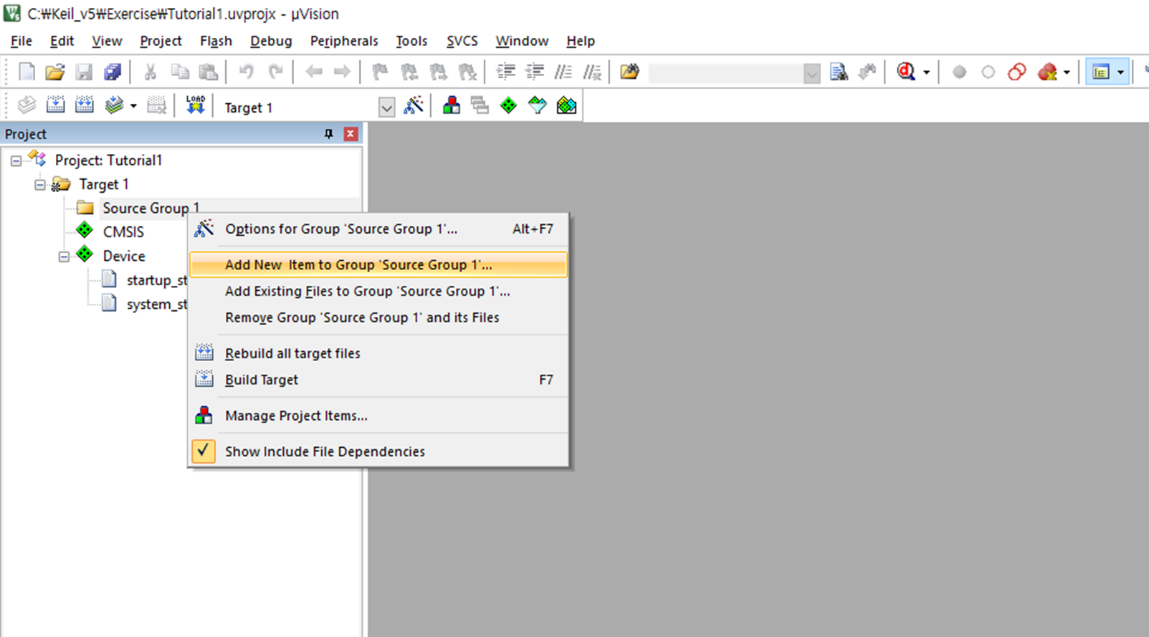


1. On the **Manage Run-Time Environment** window,

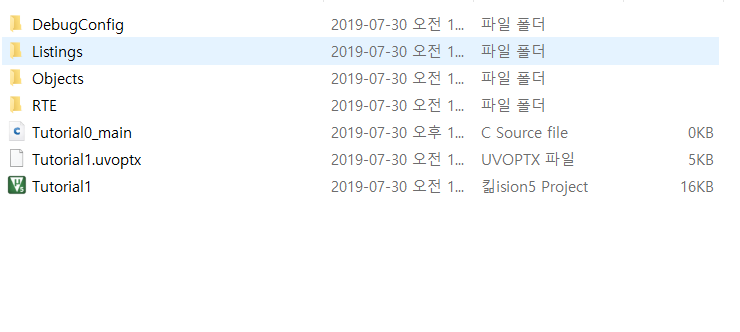
* Select ‘**Software Component’**🡪 **CMSIS🡪 Core &**
* Select ‘**Device 🡪Startup.** It will create the system start-up files



1. You will see a blank project created. Check if the following startup codes are included under **Device** folder

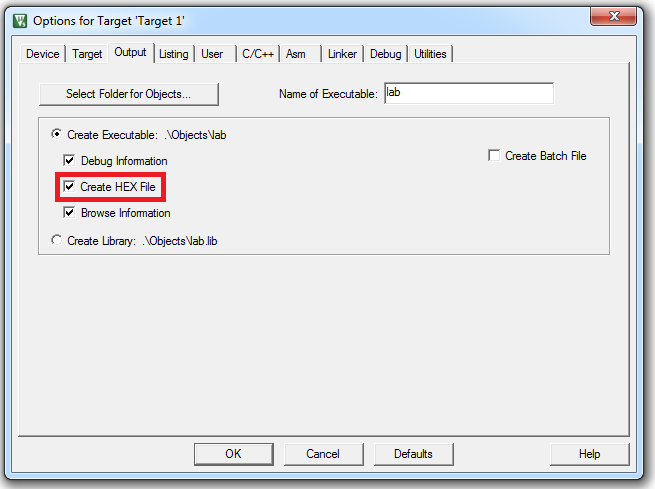
 ‘startup\_stm32f411xe.s’, ‘system\_stm32f411xx.c’

1. To program source codes, add ‘New items’ or add ‘Existing Files’ to **Source Group 1**. For this tutorial, createnew itemcalled **‘Tutorial0\_main.c’**

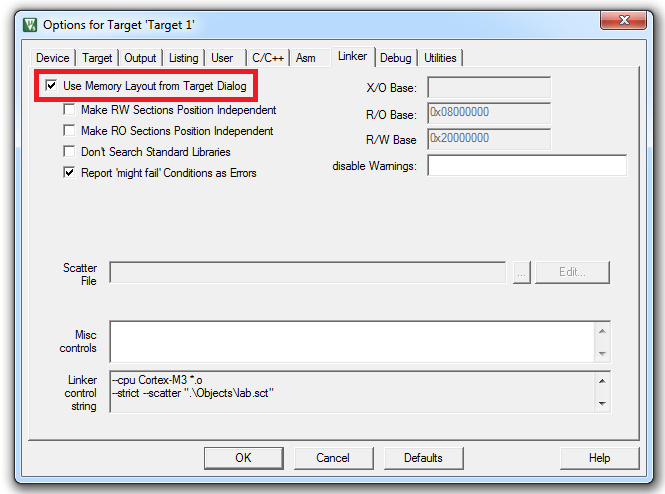


1. To set Project Properties, select **Project Menu** ⟶ ‘**Option for Target’**

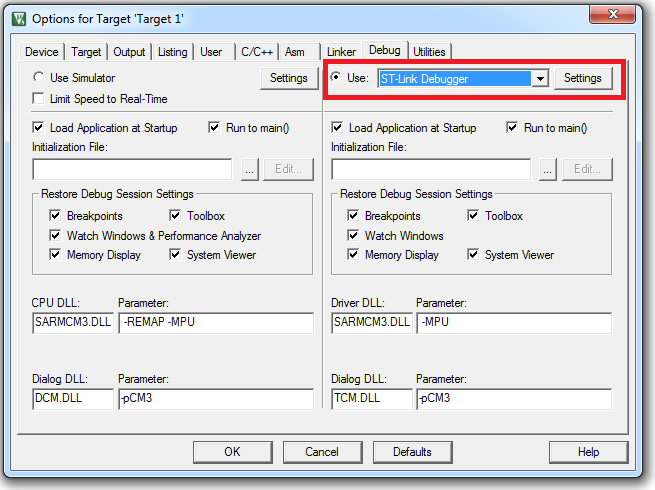
* Select ‘**Output’** tab and check “**Create HEX file**”
  + This will create a file which contains the machine instruction codes



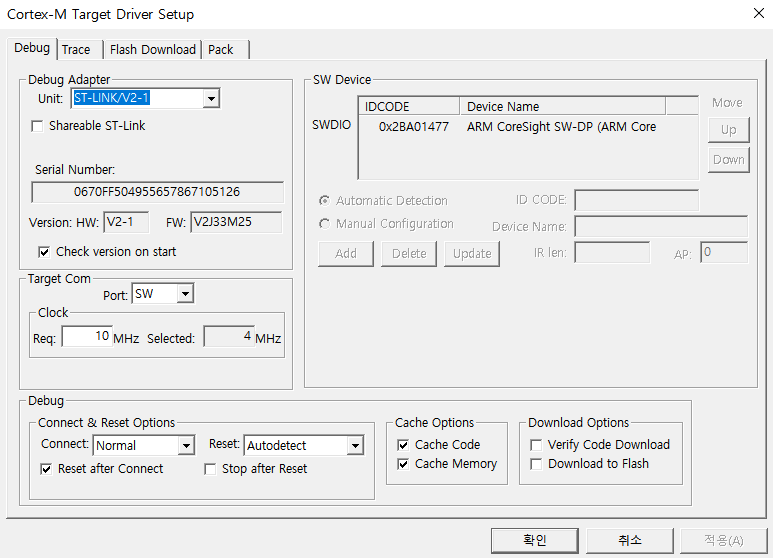
* Select ‘**Linker’** tab, and check “**Use Memory Layout from Target Dialog**”
  + This will automatically create a linker scatter file from the memory information supplied in the dialogs [Target](http://www.keil.com/support/man/docs/uv4/uv4_dg_adstarg.htm) and [Properties](http://www.keil.com/support/man/docs/uv4/uv4_dg_property.htm).



* Select **‘Debug’** tab, and check **“ST-Link Debugger”**
  + It will use ST-Link debugger embedded on the target board to debug the program. You will need to connect the target board to your PC for debugging.



* Click the ‘**Settings**’ of ‘ST-Link Debugger’ and a new window ‘**Cortex-M Target Driver Setup**’ will be opened. Select “**SW**” (Serial Wire) as the **Target Com Port.**
  + ‘JTAG’ or ‘SerialWire’ is the type of ULINK adapter. We will use ‘SW’ type.

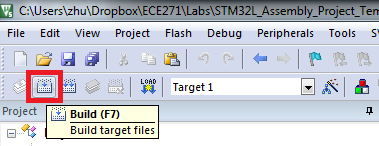


1. Copy and paste the following code in your ‘**Tutorial0\_main.c**’source file

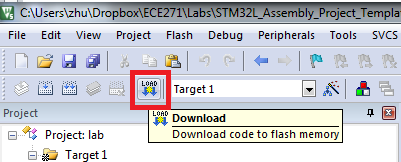
|  |
| --- |
| #include "stm32f4xx.h"  #define LED\_PIN 5  #define BUTTON\_PIN 13  int main(void) {  // RCC Control Register (HSI)  RCC->CR |= ((uint32\_t)RCC\_CR\_HSION);  // wait until HSI is ready  while ( (RCC->CR & (uint32\_t) RCC\_CR\_HSIRDY) == 0 ) {;}  // Select HSI as system clock source  // RCC Configuration Register  RCC->CFGR &= (uint32\_t)((uint32\_t) ~(RCC\_CFGR\_SW));  RCC->CFGR |= (uint32\_t)RCC\_CFGR\_SW\_HSI;  // Wait till HSI is used as system clock source  while ((RCC->CFGR & (uint32\_t)RCC\_CFGR\_SWS) != 0 ) {;}  // HSI is used as system clock  // RCC Peripheral Clock Enable Register  RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOAEN;  // GPIO Mode Register  GPIOA->MODER &= ~(3UL<<(2\*LED\_PIN));  GPIOA->MODER |= 1UL<<(2\*LED\_PIN);  // GPIO Output Speed Register  GPIOA->OSPEEDR &= ~(3UL<<(2\*LED\_PIN));  GPIOA->OSPEEDR |= 2UL<<(2\*LED\_PIN);  // GPIO Output Type Register  GPIOA->OTYPER &= ~(1UL<<LED\_PIN);  // GPIO Pull-Up/Pull-Down Register  GPIOA->PUPDR &= ~(3UL<<(2\*LED\_PIN));  // Dead loop & program hangs here  while(1){GPIOA->ODR = 1UL << LED\_PIN; }  } |

1. **Compile and run** your project.

* Build the target files (F7)



* If there is no compile error, then **Flash** or **Download code to flash memory** of the test board.



1. Check if the LED5 of testboard is turned on.

Comments:

* Instead of starting from a blank project, use example codes and tutorial codes provided by ARM and STM webpage.