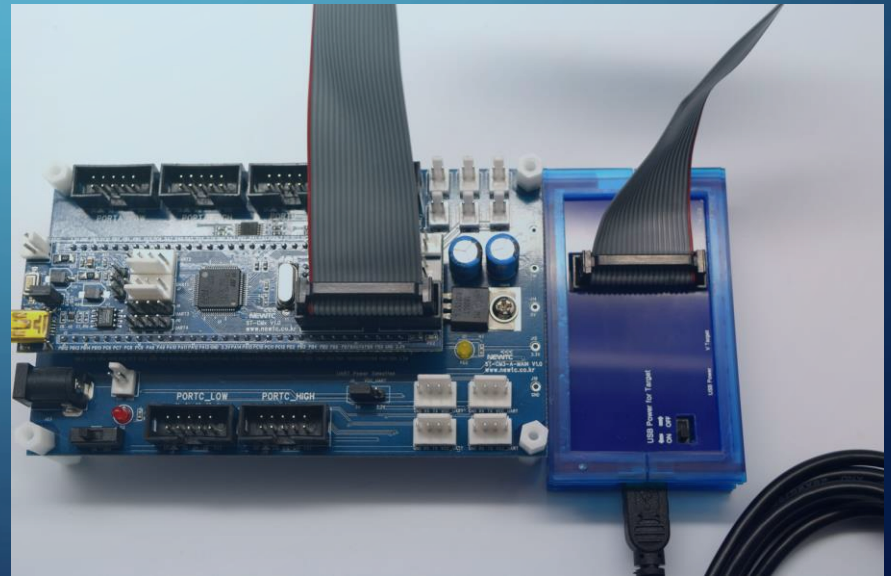


# STM32 CORTEX-M3 새 프로젝트 만들기

(주) 뉴티씨

대표이사 류대우





# 목차

1. 새 프로젝트 생성

2. 컴파일



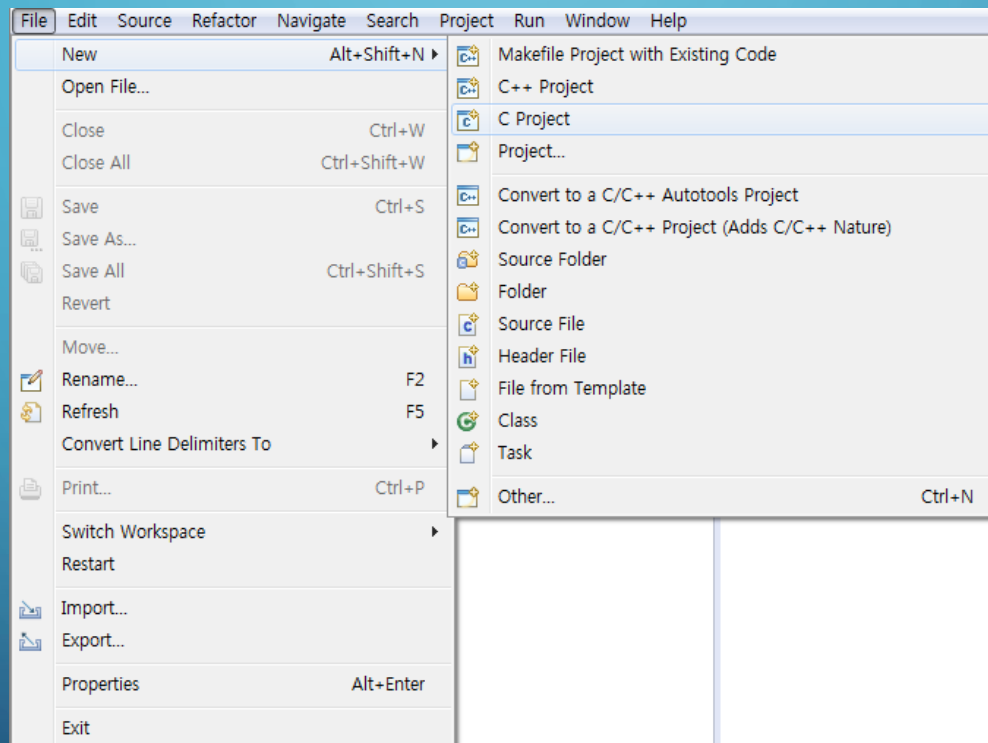
# 1. 새 프로젝트 생성

1) 새로운 프로젝트를 생성한다.

2) 보드의 설정에 맞게 구성한다.

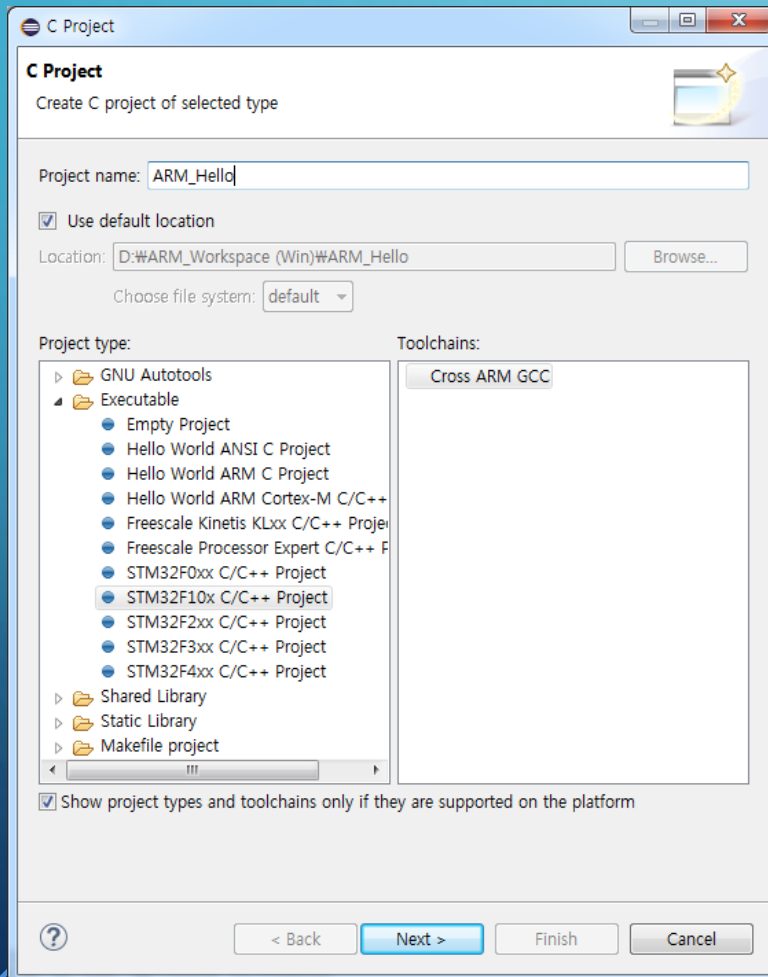
3) 예제 코드를 작성한다.

# 1-(1)-A. 새 프로젝트 생성



File -> New -> C Project 클릭

# 1-(2)-A. 새 프로젝트 생성

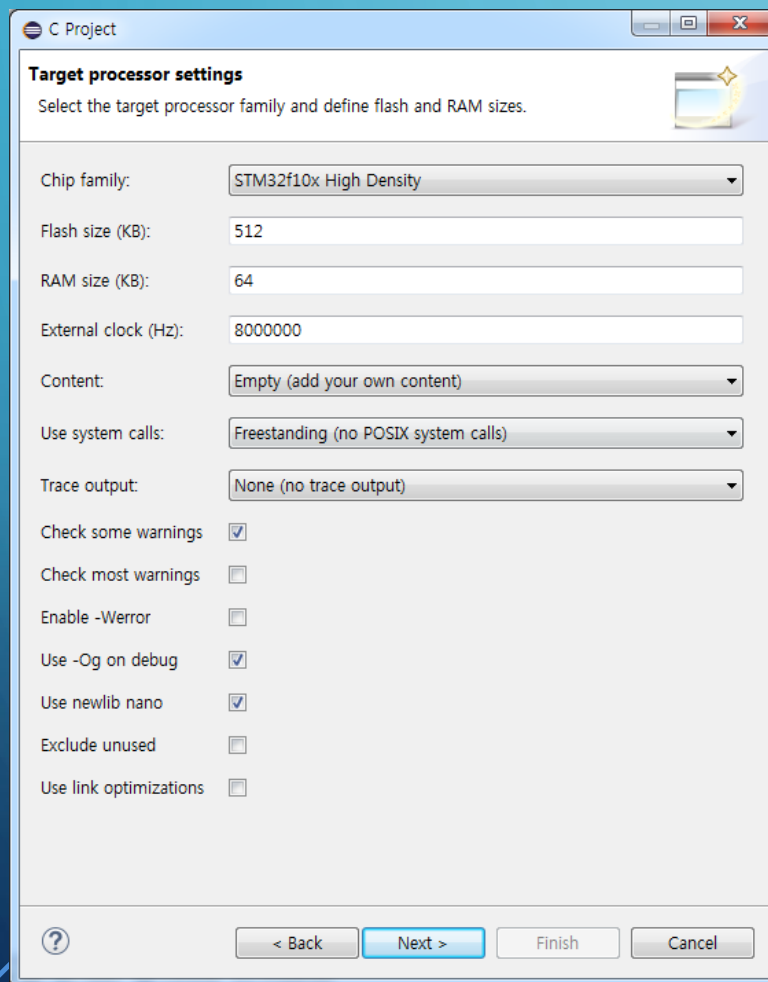


Project Type :  
Executable/STM32F10x C/C++ Project

Project Name :  
프로젝트 명 (영문 및 숫자만 사용.)  
(첫 글자는 영문)

모두 설정 후 Next 클릭

# 1-(2)-B. 새 프로젝트 생성



Chip family :  
STM32f10x High Density

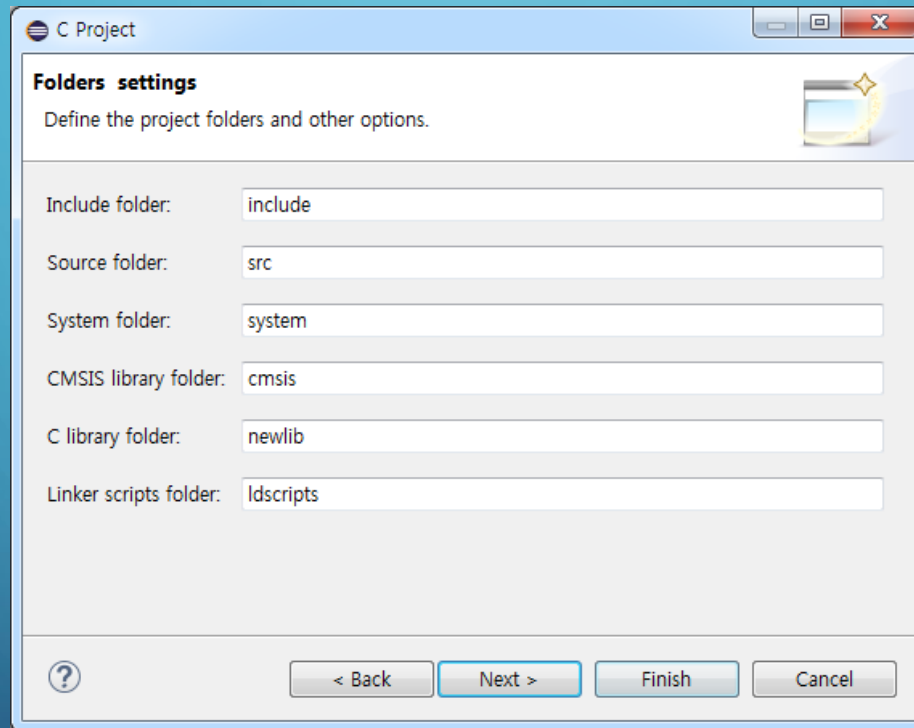
Flash size, RAM size :  
512, 64

External clock (Hz):  
8000000

나머지는 사진처럼 설정

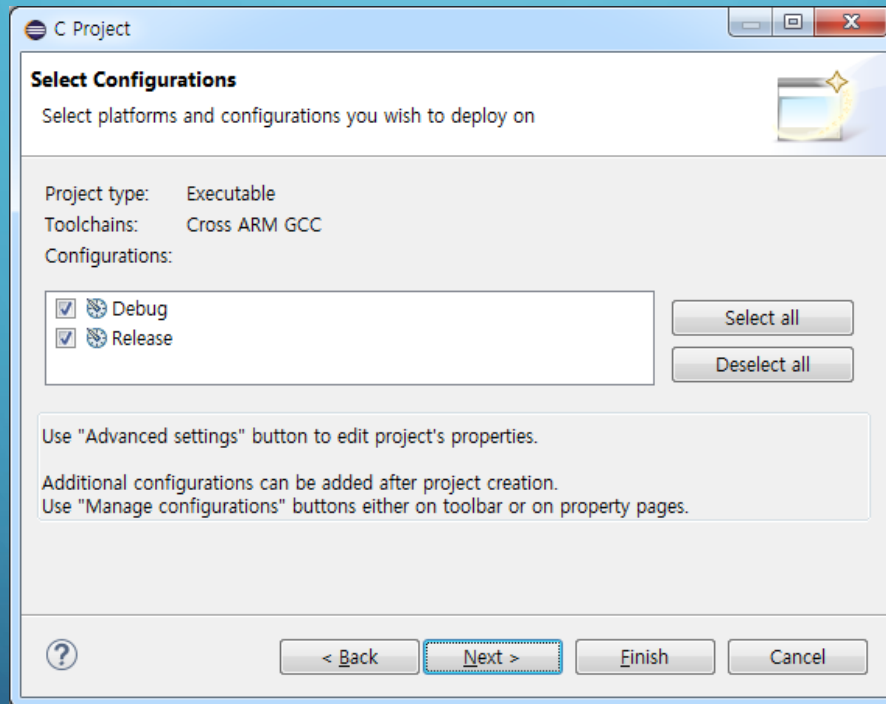
설정 후 Next 버튼 클릭

# 1-(2)-C. 새 프로젝트 생성



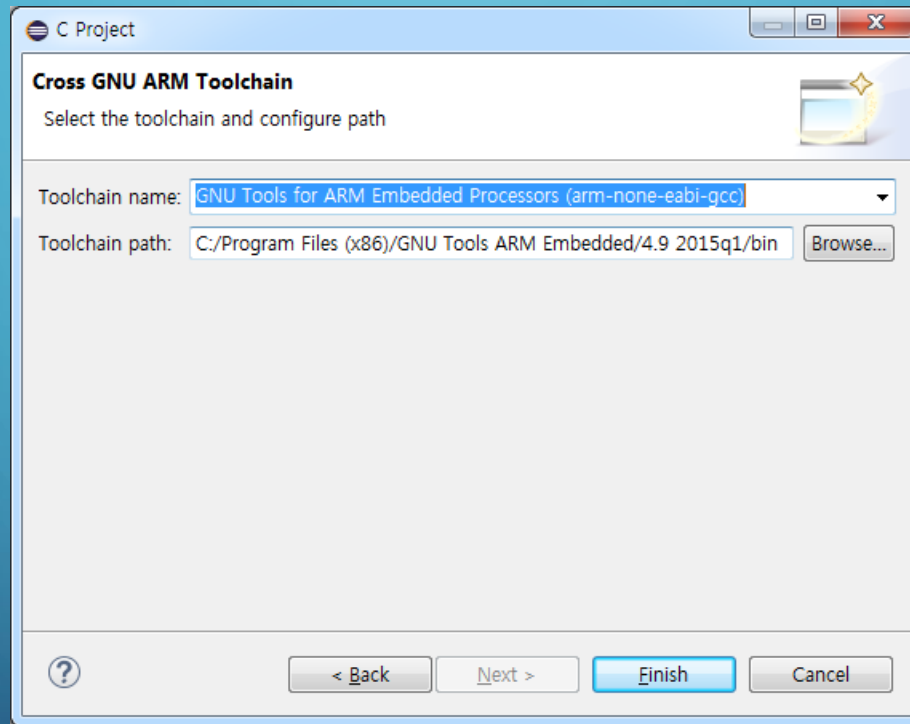
Next 버튼 클릭

# 1-(2)-D. 새 프로젝트 생성



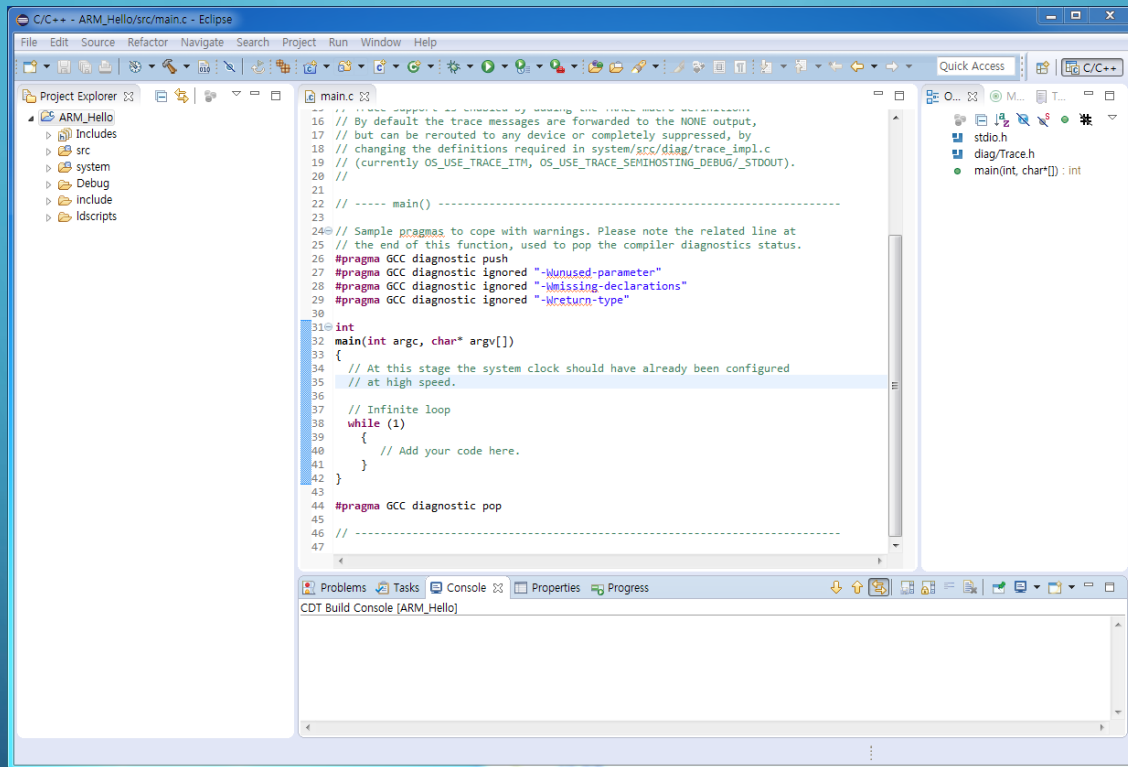
Next 버튼 클릭

# 1-(2)-E. 새 프로젝트 생성



Finish 버튼 클릭

# 1-(2)-F. 새 프로젝트 생성



초기 개발환경 확인

## 1-(3). 새 프로젝트 생성 - 예제 코드 작성

```
#include "stm32f1x_conf.h"

void delay_ms(int ms)
{
    for(int i=0;i<ms/2;i++)
    {
        for(int j=0;j<12000;j++);
    }
}

int main(int argc, char* argv[])
{
    GPIO_InitTypeDef PC;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
```

```
PC.GPIO_Mode=GPIO_Mode_Out_PP;
PC.GPIO_Pin=GPIO_Pin_All;
PC.GPIO_Speed=GPIO_Speed_50MHz;
GPIO_Init(GPIOC,&PC);
```

```
while (1)
{
    delay_ms(1000);

    GPIO_Write(GPIOC,0xFFFF);

    delay_ms(1000);

    GPIO_Write(GPIOC,0x0000);

}
```

```
}
```



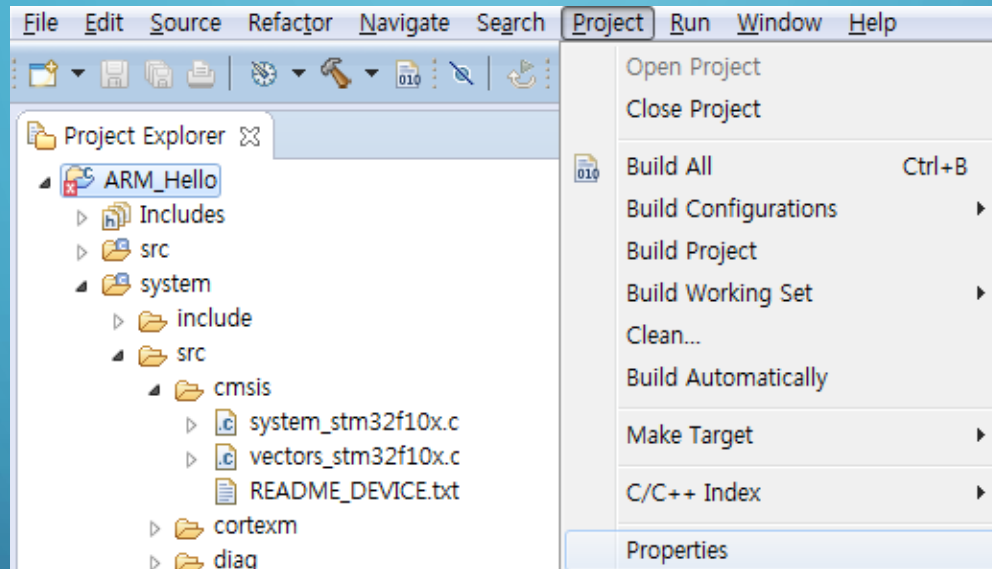
## 2. 컴파일

1) 컴파일러를 설정한다.

2) 컴파일한다.

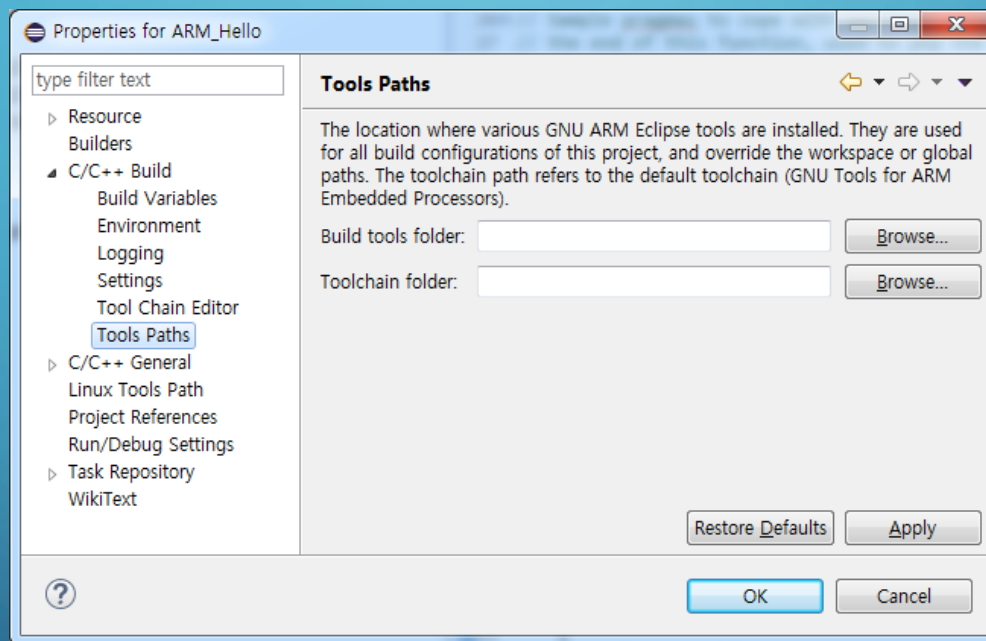


## 2-(1)-A. 컴파일



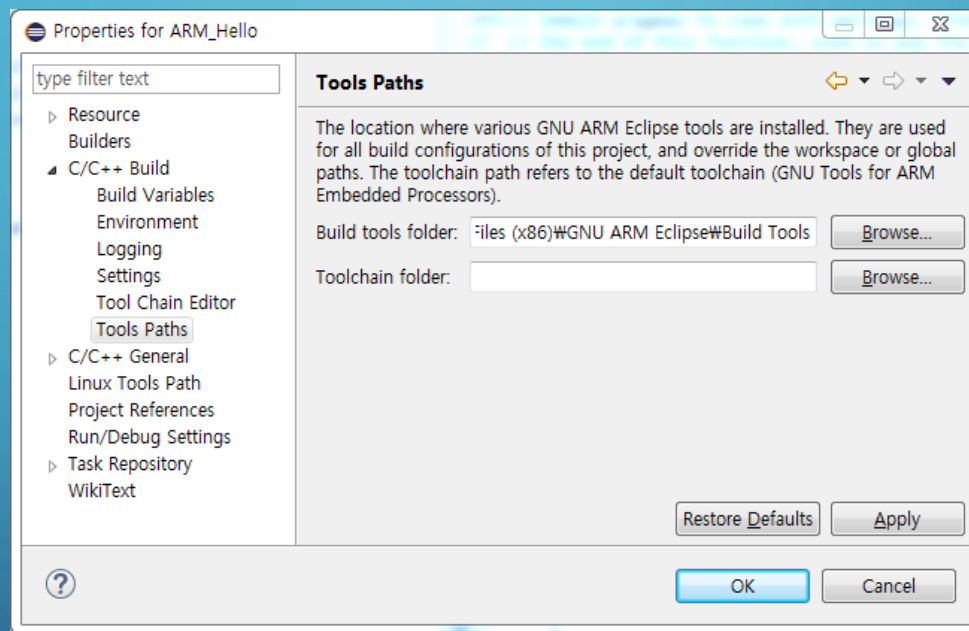
Project -> Properties 를 클릭하여 프로젝트 속성 진입  
(Project Explorer에서 프로젝트 명을 클릭)

## 2-(1)-B. 컴파일



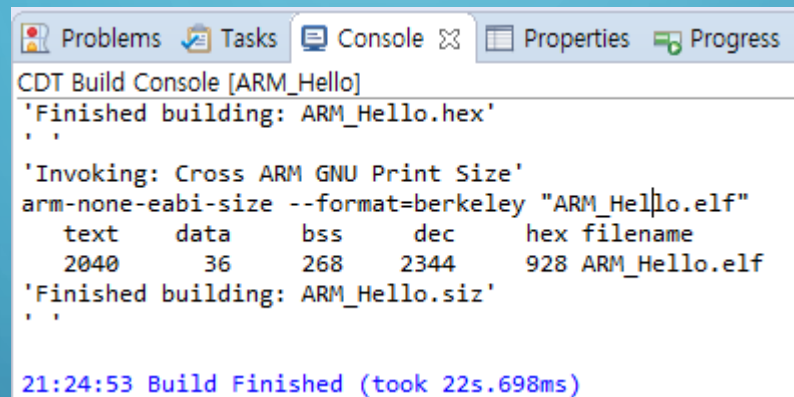
C/C++ Build -> Tool Paths 진입  
그 후 Build tools folder의 Browse 클릭

## 2-(1)-C. 컴파일



설정한 Build Tools 의 경로를 설정  
(32bit 기본값 : C:\Program Files\GNU ARM Eclipse\Build Tools)  
(64bit 기본값 : C:\Program Files (x86)\GNU ARM Eclipse\Build Tools)  
그 후 OK 버튼 클릭

## 2-(2). 컴파일



The screenshot shows the CDT Build Console for a project named 'ARM\_Hello'. The console output indicates that the build is finished for 'ARM\_Hello.hex'. It then shows the invocation of 'Cross ARM GNU Print Size' to generate a size report for 'ARM\_Hello.elf'. The report lists memory usage for text, data, bss, and dec sections in both decimal and hexadecimal, along with the filename. Finally, it shows the build is finished for 'ARM\_Hello.siz'.

```
CDT Build Console [ARM_Hello]
'Finished building: ARM_Hello.hex'
', '
'Invoking: Cross ARM GNU Print Size'
arm-none-eabi-size --format=berkeley "ARM_Hello.elf"
  text    data    bss     dec     hex filename
 2040     36    268    2344    928 ARM_Hello.elf
'Finished building: ARM_Hello.siz'
', '

21:24:53 Build Finished (took 22s.698ms)
```

Ctrl+B 키를 눌러 컴파일