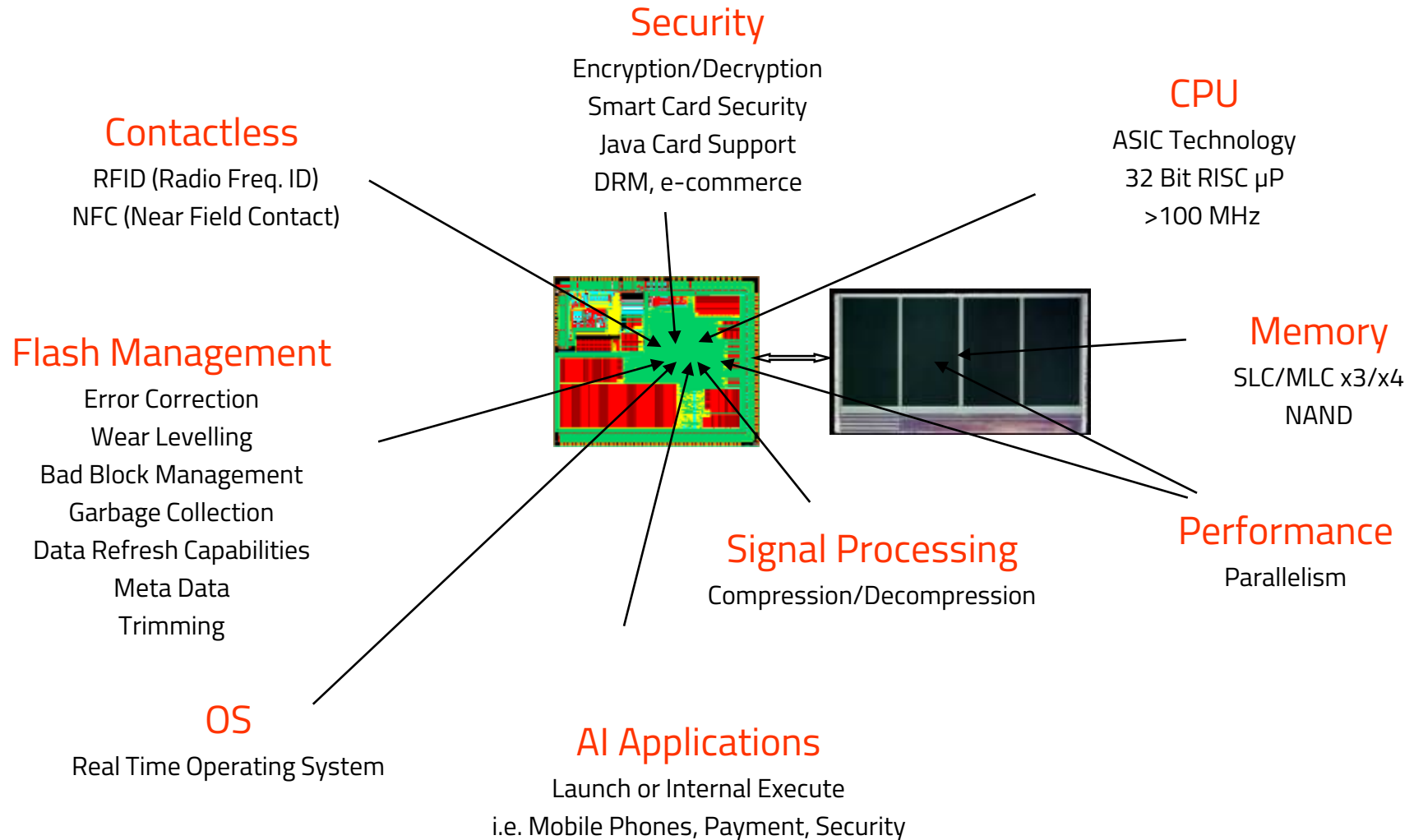


# 임베디드 MCU 프로그래밍 실습

## AURIX TC275 보드 프로젝트 생성

현대자동차 입문교육  
박대진 교수

# 하드웨어-소프트웨어 통합된 임베디드 시스템



# Debugging

## Compilation

## C language

# Assembling

## Assembly code (Machine code)

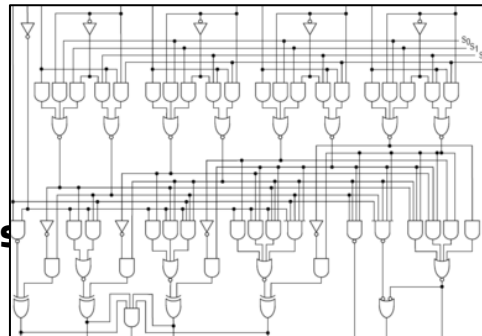
### Binary code (App)



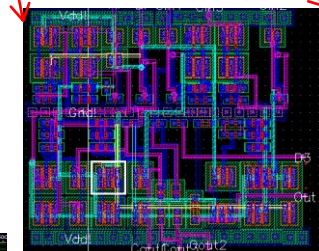
S/W

## H/W

## Synthesis



## Place & Route (Layout)



## Simulation



## 임베디드 시스템 프로그래밍 → 0/1을 온칩에 임베딩

- Software is deeply injected into the hardware silicon
  - Your dirty software code ... is directly translated into the machine code.
  - So that performance degradation starts from inefficiency of my code.

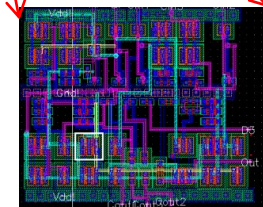
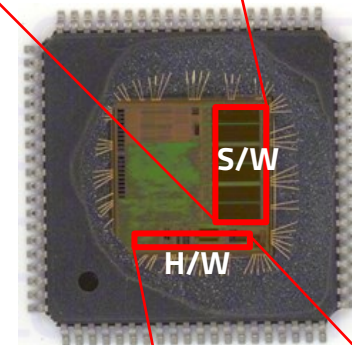
## Compilation/Assembling/Linking/Downloaded

```
int main() {
    int s, y;
    s = 1;
    y = s + 2;
    return y;
}
```

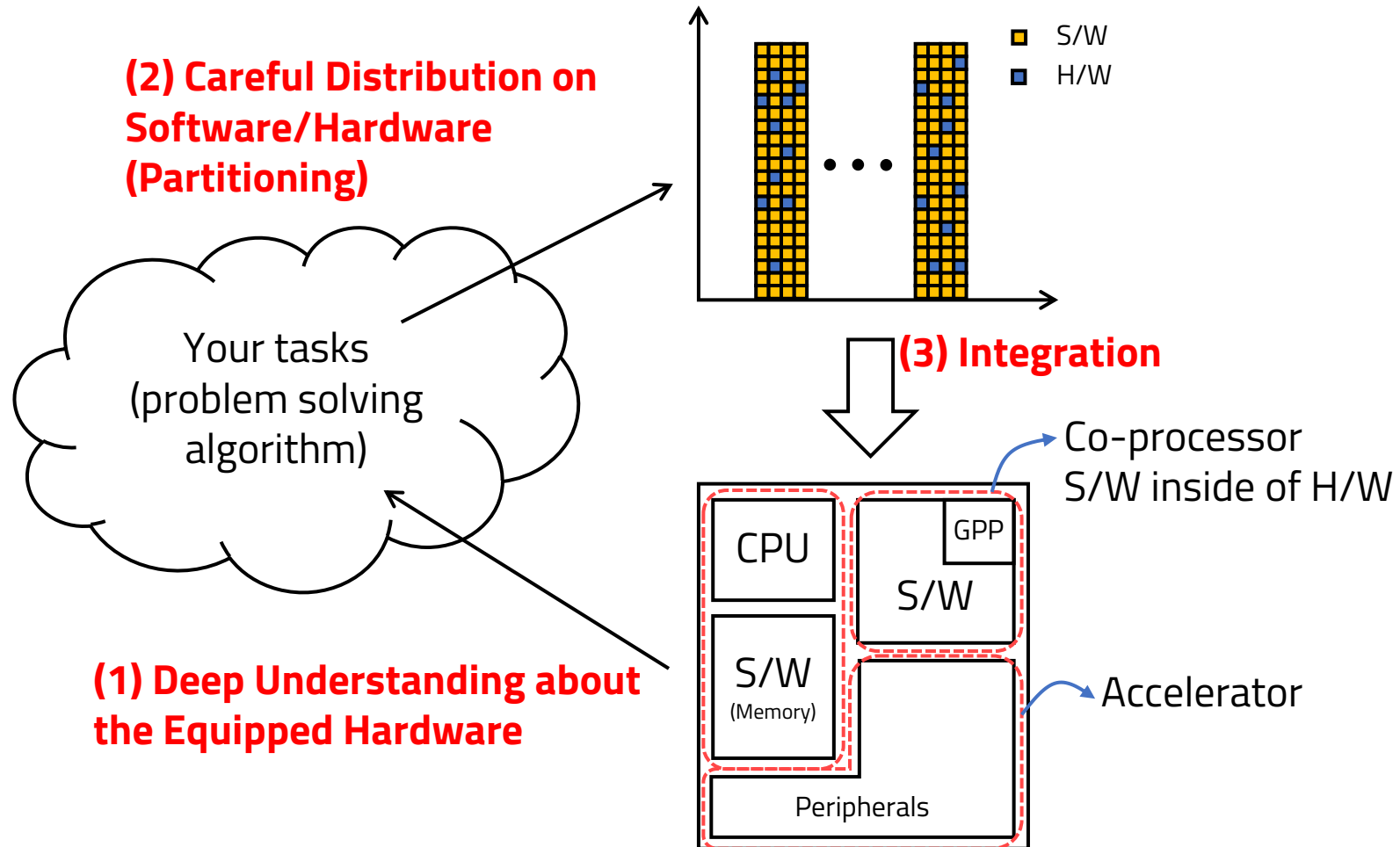
```

004013e0 <main>:
int main() {
    4013e0: 55                push    %ebp
    4013e1: 89 e5            mov     %esp,%ebp
    4013e3: 83 e4 f0        and     $0xfffffffff0,%esp
    4013e6: 83 ec 10        sub     $0x10,%esp
    4013e9: e8 a2 04 00 00  call   401890 <main>
        int s, y;
        s = 1;
    4013ee: c7 44 24 0c 01 00 00 movl    $0x1,0xc(%esp)
    4013f5: 00
        y = s + 2;
    4013f6: 8b 44 24 0c      mov     0xc(%esp),%eax
    4013fa: 83 c0 02        add     $0x2,%eax
    4013fd: 89 44 24 08      mov     %eax,0x8(%esp)
        return y;
    401401: 8b 44 24 08      mov     0x8(%esp),%eax
}

```

[illegible]

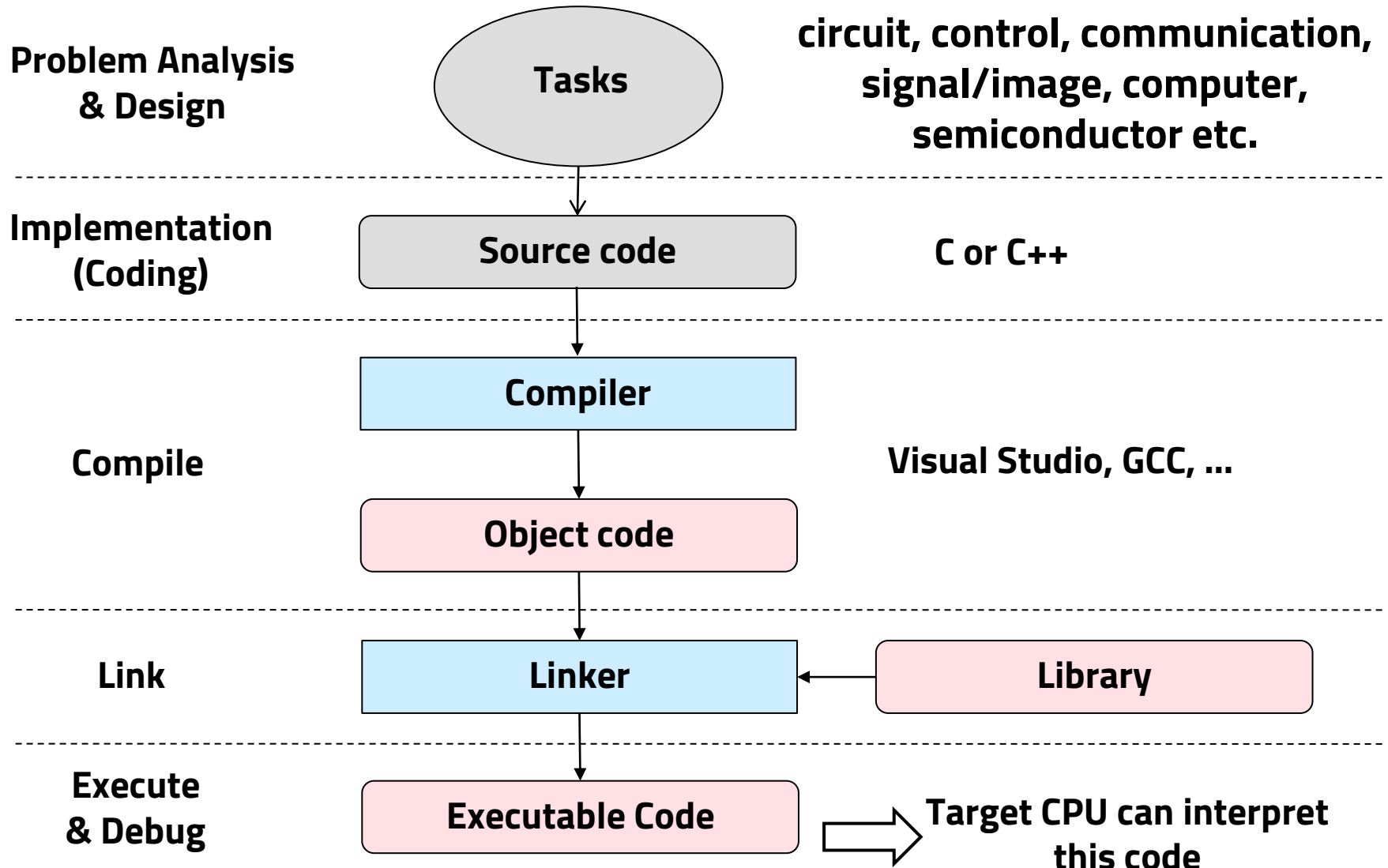
# 임베디드 S/W가 칩 내부의 H/W를 구동함



System On Chip (Latest Processor Architecture)

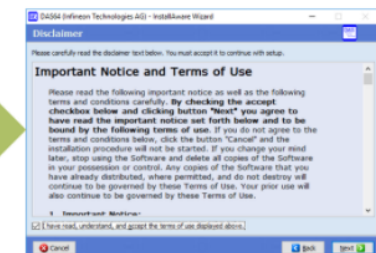
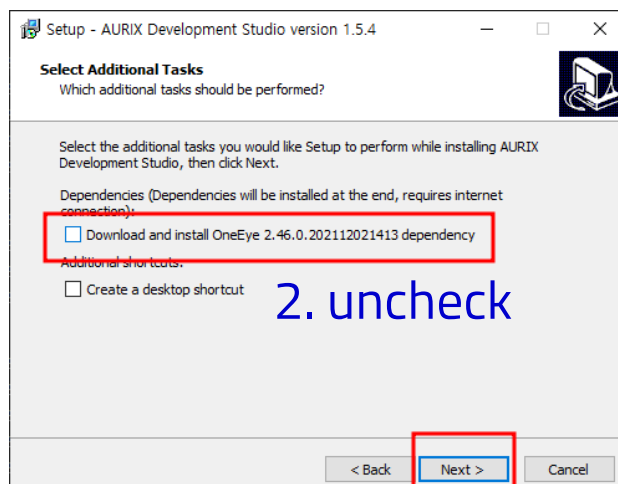
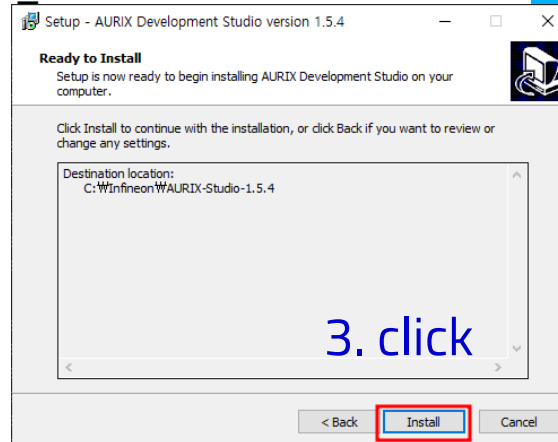
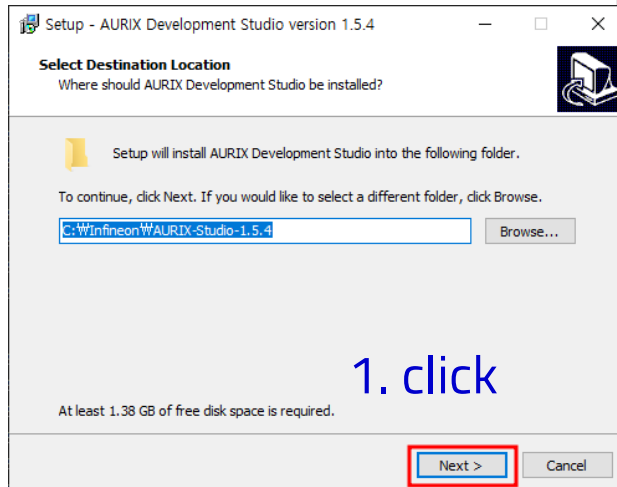
# Compilation

## Tasks → Executable Binary Code



# Installing Target-Specific Compiler

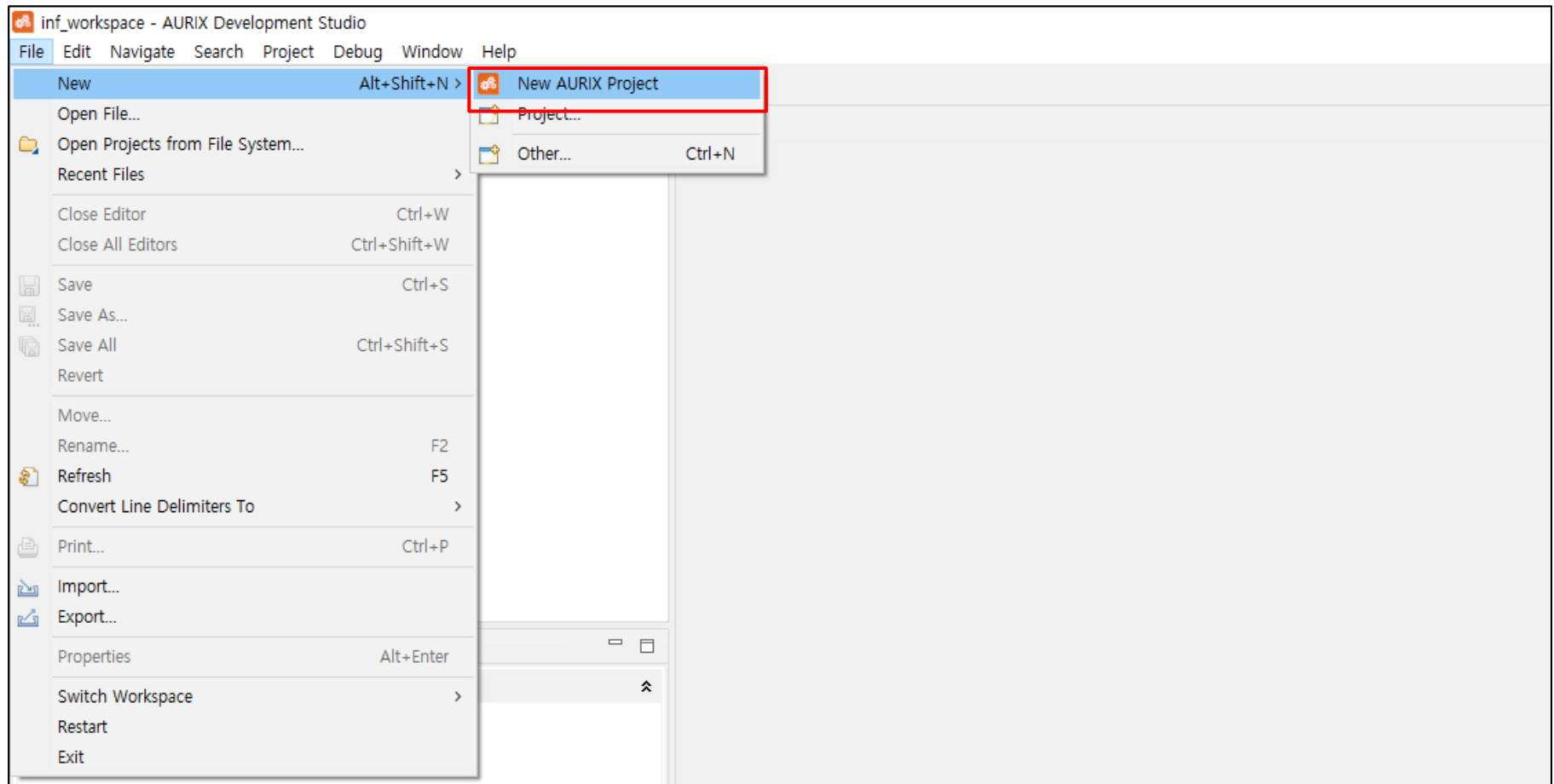
- AURIX™ Development Studio Install
  - AURIX-studio-setup\_1.5.4\_20211221-1518.exe



# New Project

## : Template Generation

1. AURIX Development Studio를 실행하고 왼쪽 상단의 '**File – New –New AURIX Project**' 를 클릭한다.





# New Project

## : Project Name

2. 생성할 Project의 이름을 입력하고, '**Next**'를 클릭한다.

New AURIX Development Studio Project

**New AURIX Project**

Specify the name and the location of the new project

Project name:

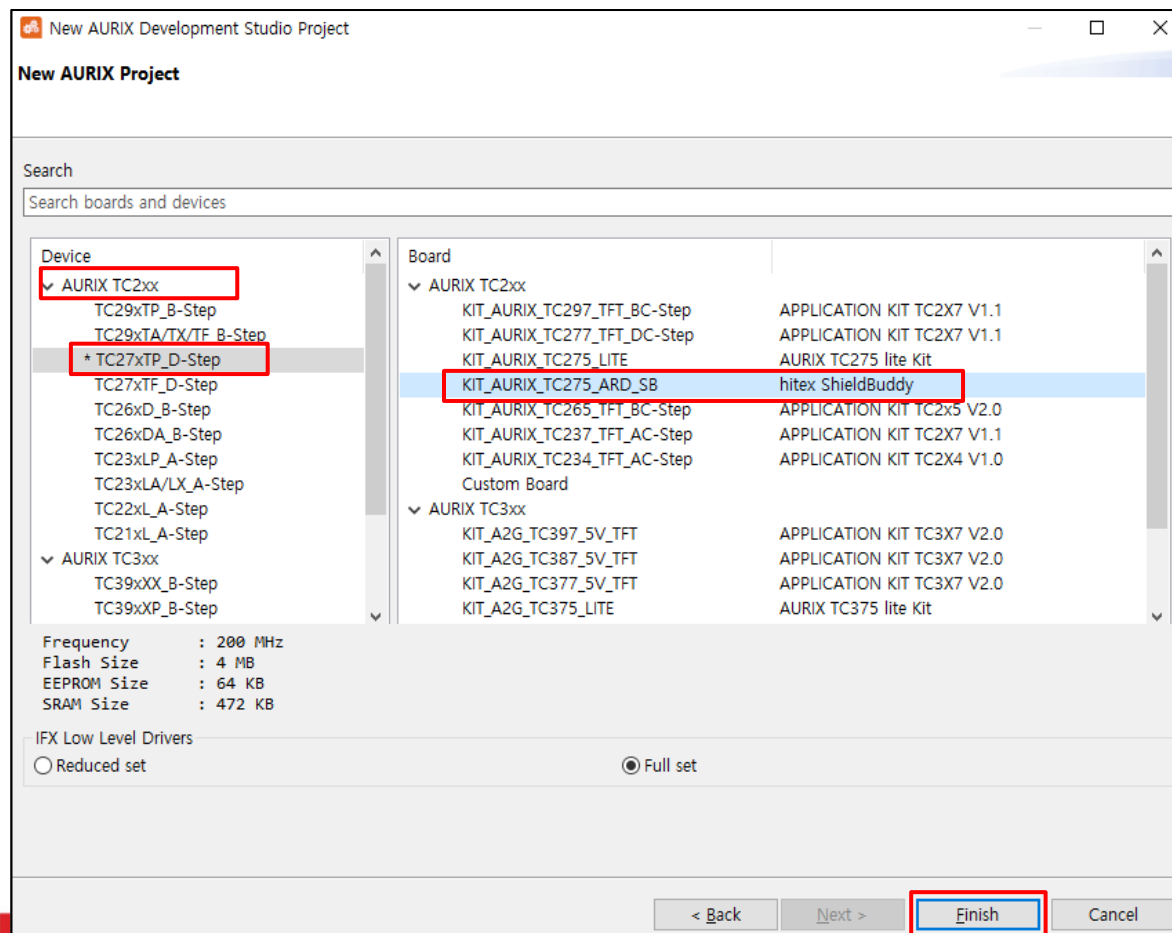
☒ Use default location

Location:

# New Project

## : Target CPU, Target Board Selection

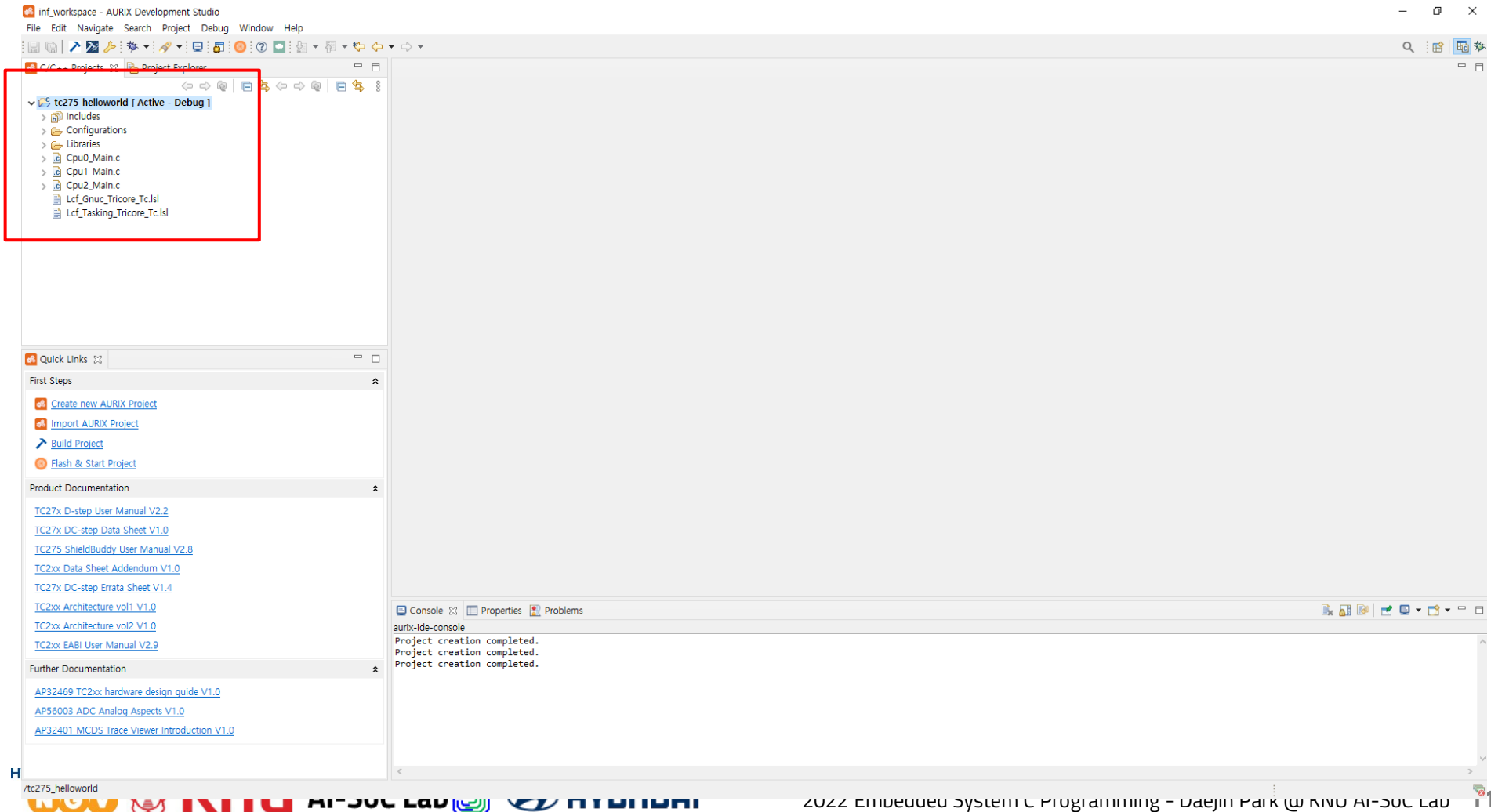
3. Device에서 '**AURIX TC2xx – TC27XTP\_D-Step – KIT\_AURIX\_TC275\_ARD\_SB**' 를 선택하고, 다른 설정은 그대로 유지한 채 '**Finish**'를 클릭한다.



# New Project

## : Code Explorer

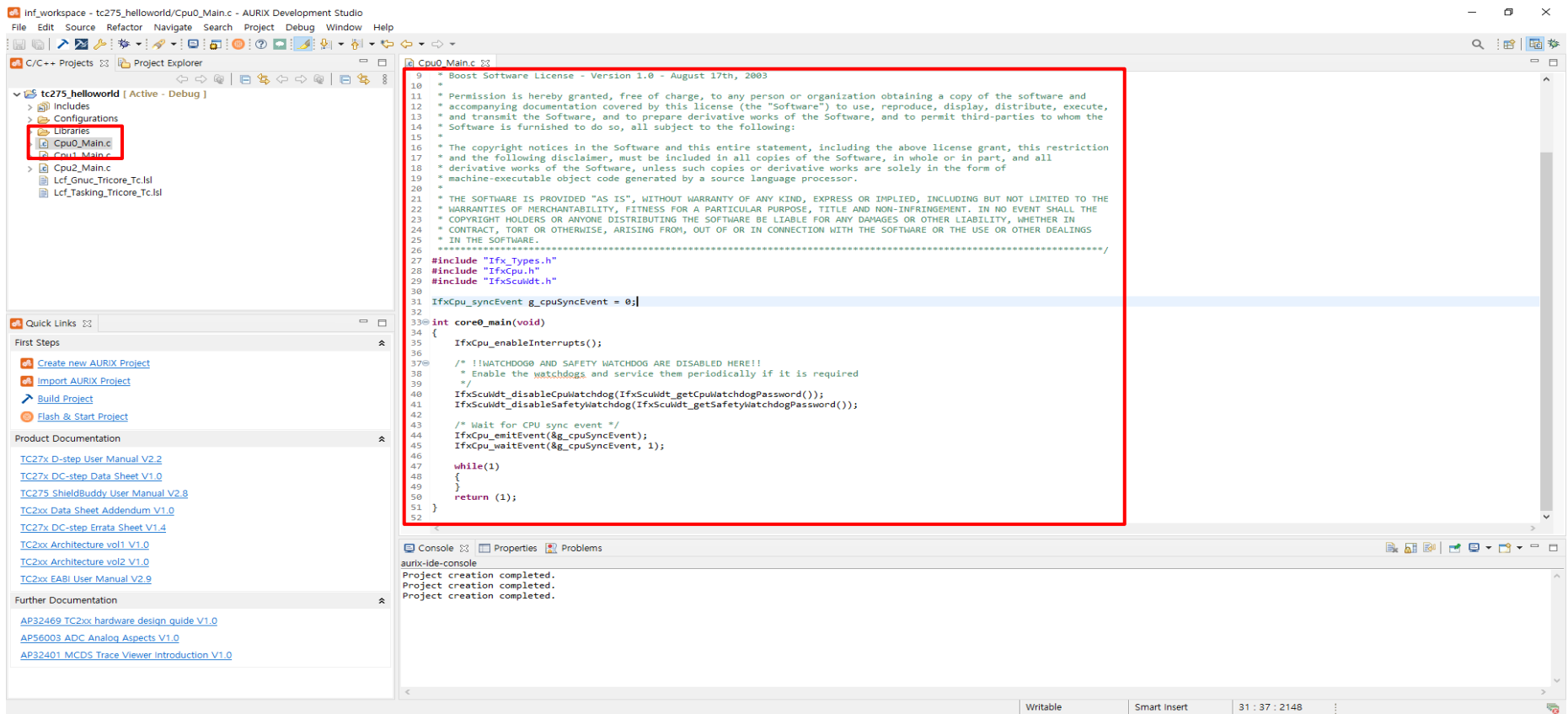
4. 왼쪽의 Project Explorer 창에서 프로젝트가 생성된 것을 확인한다.



# Edit Project

## : Source Code Editing

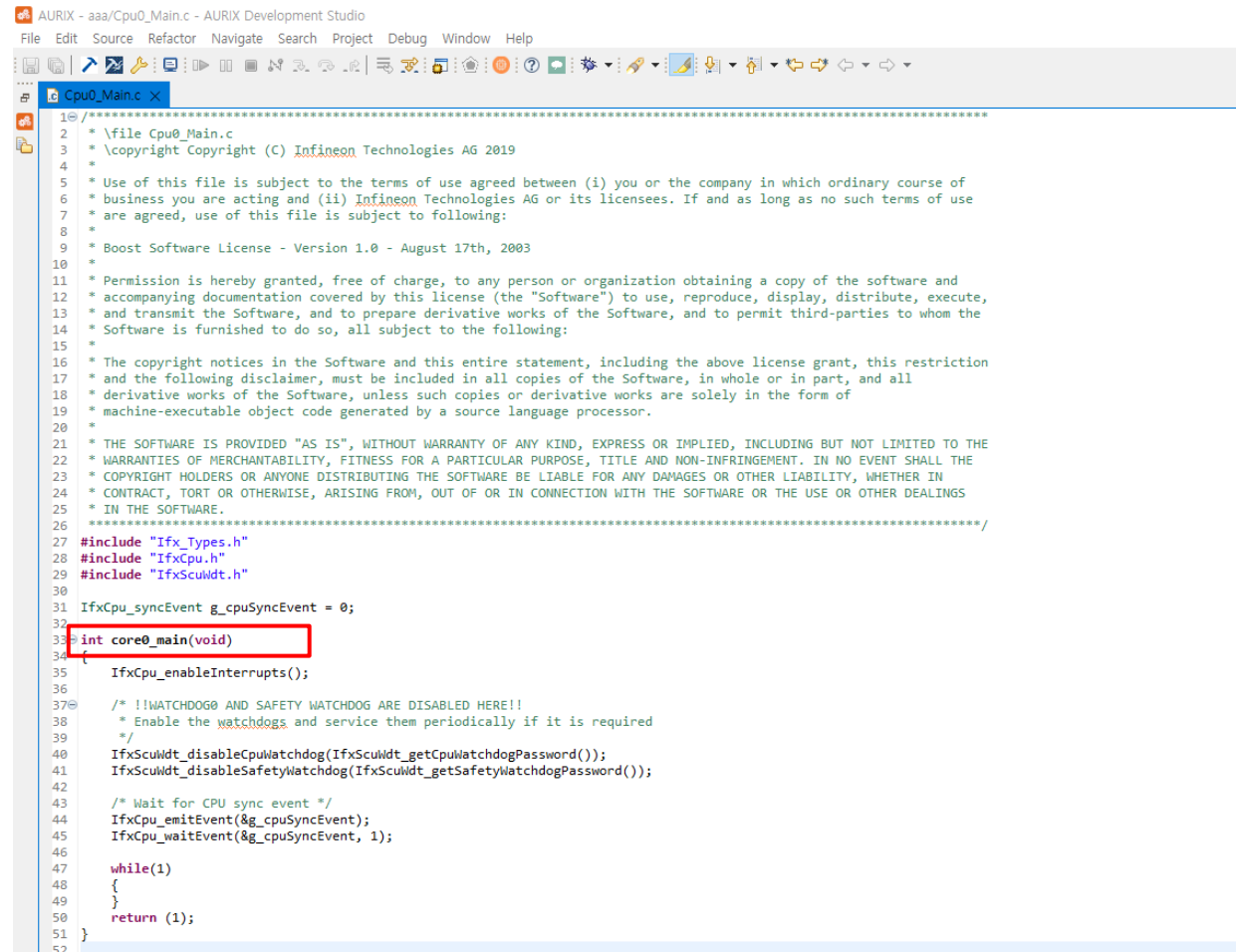
5. 왼쪽의 Project Explorer 창에서 **Project name**으로 생성된 파일인 '**Cpu0\_Main.c**' 파일을 더블 클릭하여 활성화한다.



# Edit Project

## : main function

6. 'Cpu0\_Main.c' 파일은 **core0\_main** 함수를 포함하고 있으며 이를 수정하여 보드의 동작을 설계한다.

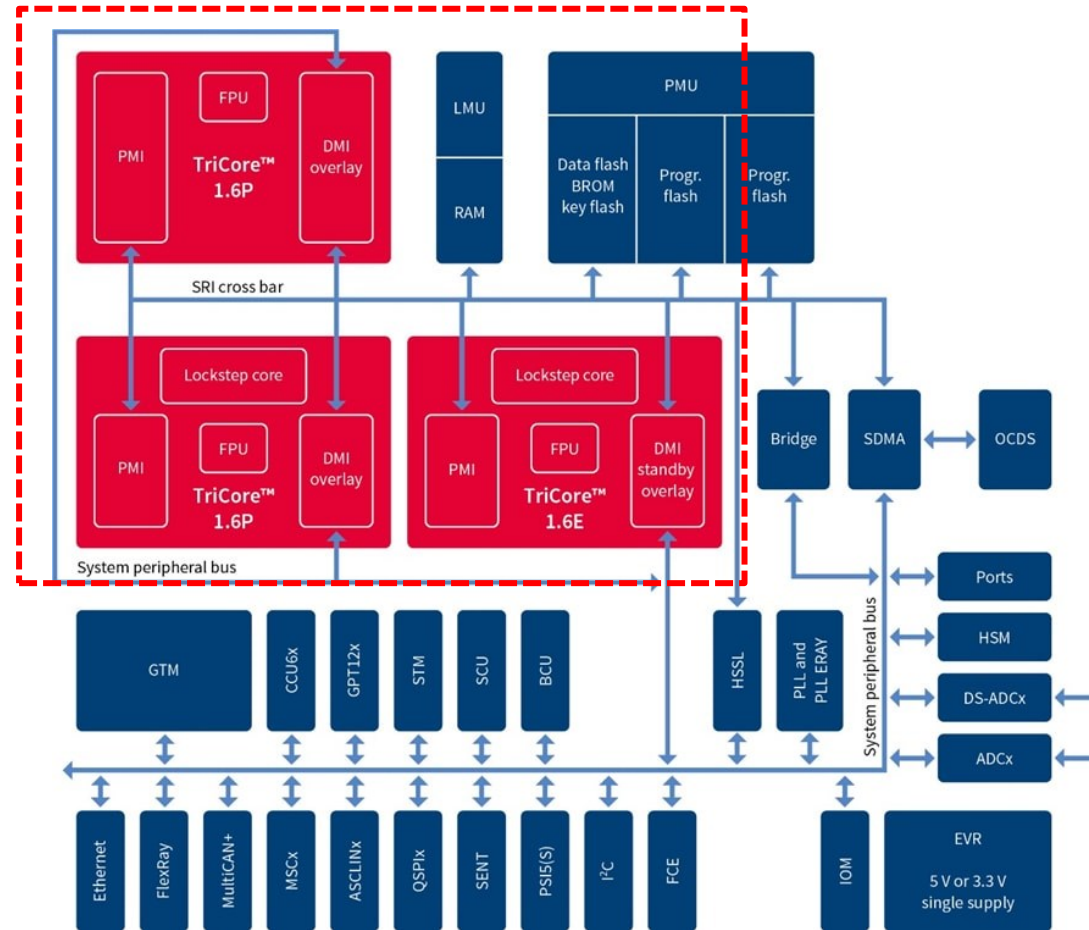


```
1 //*****
2 * \file Cpu0_Main.c
3 * \copyright Copyright (C) Infineon Technologies AG 2019
4 *
5 * Use of this file is subject to the terms of use agreed between (i) you or the company in which ordinary course of
6 * business you are acting and (ii) Infineon Technologies AG or its licensees. If and as long as no such terms of use
7 * are agreed, use of this file is subject to the following:
8 *
9 * Boost Software License - Version 1.0 - August 17th, 2003
10 *
11 * Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and
12 * accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute,
13 * and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the
14 * Software is furnished to do so, all subject to the following:
15 *
16 * The copyright notices in the Software and this entire statement, including the above license grant, this restriction
17 * and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all
18 * derivative works of the Software, unless such copies or derivative works are solely in the form of
19 * machine-executable object code generated by a source language processor.
20 *
21 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
22 * WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE
23 * COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN
24 * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
25 * IN THE SOFTWARE.
26 *
27 *****/
28 #include "IfxTypes.h"
29 #include "IfxCpu.h"
30 #include "IfxScuMdt.h"
31
32 IfxCpu_syncEvent g_cpuSyncEvent = 0;
33
34 int core0_main(void)
35 {
36     IfxCpu_enableInterrupts();
37
38     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
39      * Enable the watchdogs and service them periodically if it is required
40      */
41     IfxScuMdt_disableCpuWatchdog(IfxScuMdt_getCpuWatchdogPassword());
42     IfxScuMdt_disableSafetyWatchdog(IfxScuMdt_getSafetyWatchdogPassword());
43
44     /* Wait for CPU sync event */
45     IfxCpu_emitEvent(&g_cpuSyncEvent);
46     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
47
48     while(1)
49     {
50     }
51
52     return (1);
53 }
```

# Edit Project

## : main functions for Tri Core

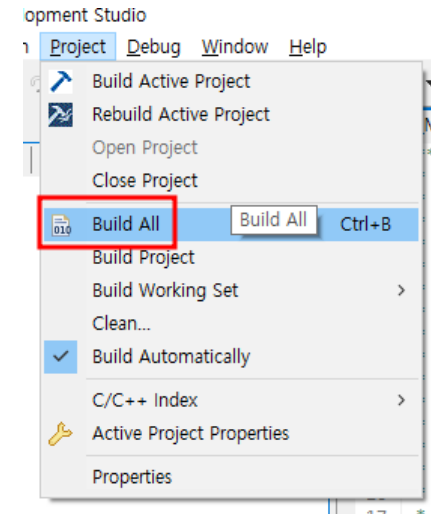
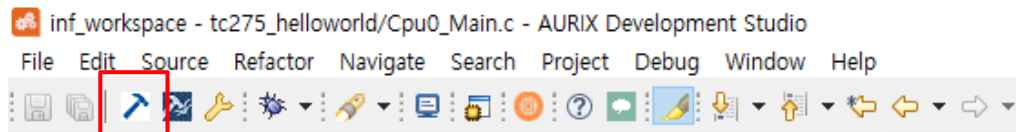
- AURIX TriCore TC275 MCU는 3개의 CPU를 갖고 있지만, 본 실습에서는 1개의 CPU만을 사용. (CPU0)
- Cpu1\_Main.c 와 Cpu2\_Main.c 파일들은 사용하지 않음.



**AURIX TC27xT 32-bit TriCore Microcontrollers**

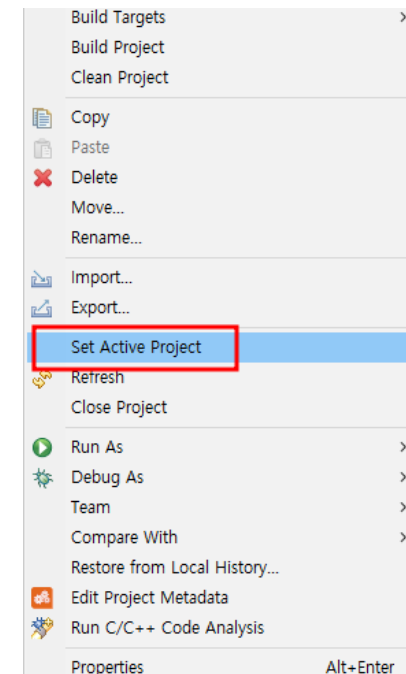
# Build

7. 상단의 메뉴에서 '**Build**' 버튼 또는 '**Project – Build All**' 을 클릭하여 프로젝트를 Build한다.



8. (Build/Debug는 Active Project에 대해 수행  
Build를 수행할 Project를 Active Project로 미리 설정해야  
한다.

'Project Explorer' 에서 대상 프로젝트를 우클릭 한 뒤, 'Set  
Active Project'를 클릭하여 Active Project로 설정할 수 있다.)

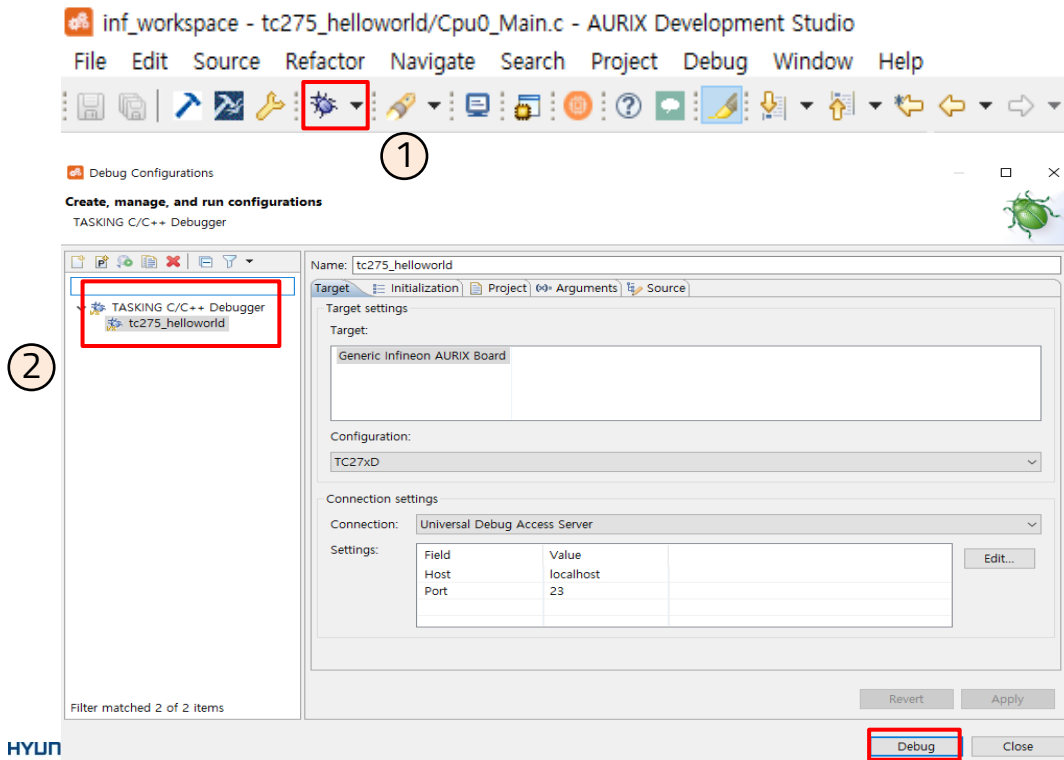


# Debug

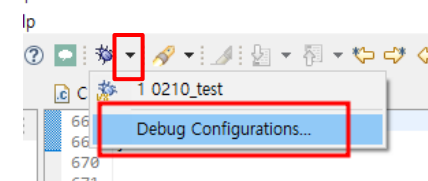
## : Configure Target Project for Debugging

9. 상단의 메뉴에서 '**Debug**' 버튼(벌레 모양)을 클릭하여 Debug를 실행한다.

- ✓ 'Debug' 버튼을 처음으로 클릭하면 'Debug Configurations' 창이 활성화된다. 왼쪽 패널에서 '**TASKING C/C++ Debugger – Project 이름**' 을 확인하고 '**Debug**' 버튼을 클릭한다.  
(이후부터는 버튼 클릭 시 Debug가 바로 실행)



다시 'Debug Configurations' 창을  
활성화하려면 'Debug' 버튼 오른쪽  
화살표 클릭  
→ 'Debug Configurations...' 버튼  
클릭

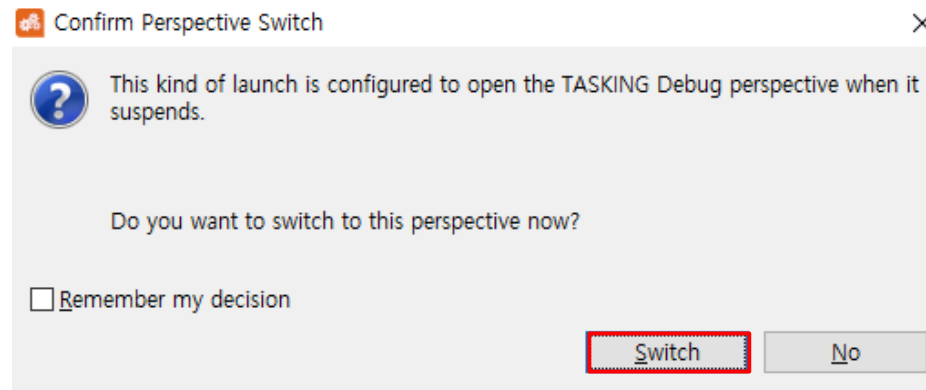




# Debug

10. 상단의 메뉴에서 '**Debug**' 버튼을 클릭하여 Debug를 실행한다.

- ✓ 'Confirm Perspective Switch' 창이 뜨면 Switch를 눌러 디버그 창으로 전환.
- ✓ (소스 코드 외에 편리한 디버깅을 위한 여러 가지 window 를 포함하는 layout 으로 전환 여부)



# Debug

## : Control Panel, Debugging Window Layout

### 11. 기본 Debug 화면 구성

The screenshot displays the AURIX Development Studio interface with the following components:

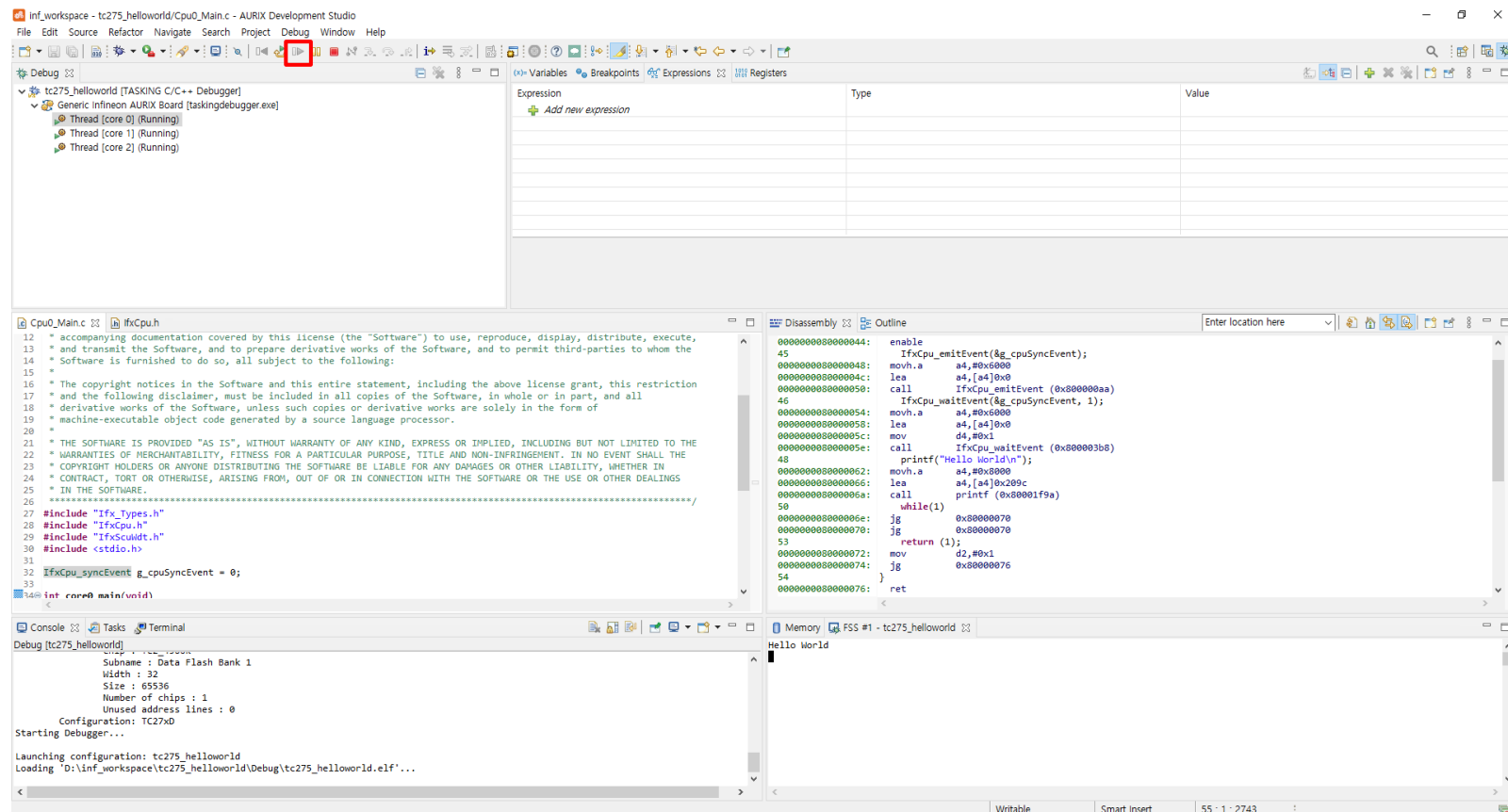
- Debug Control Panel:** Located at the top, it contains buttons for **Restart / Resume / Terminate / Step Into / Step Over** 등 디버깅 제어 버튼들 (Debugging control buttons).
- Source Code:** The left pane shows the source code for `Cpu0_Main.c` and `IfxCpu.h`.
- Disassembly:** The right pane shows the assembly code for the `enable` function, including instructions like `enable`, `IfxCpu_emitEvent(&g_cpuSyncEvent);`, `movh.a a4,#0x6000`, `lea a4,[a4]0x0`, `call IfxCpu_emitEvent (0x800000aa)`, `IfxCpu_waitEvent(&g_cpuSyncEvent, 1);`, `movh.a a4,#0x6000`, `lea a4,[a4]0x0`, `mov d4,#0x1`, `call IfxCpu_waitEvent (0x800000b8)`, `printf("Hello World\n");`, `movh.a a4,#0x8000`, `lea a4,[a4]0x209c`, `call printf (0x80001f9a)`, `while(1)`, `lg 0x80000070`, `lg 0x80000070`, `return (1);`, `mov d2,#0x1`, `lg 0x80000076`, and `ret`.
- Console:** The bottom-left pane shows the output of the debugger, including the message "Starting Debugger..." and the loading of the configuration file `tc275_helloworld`.
- Memory:** The bottom-right pane shows the memory dump for the `FSS #1 - tc275_helloworld` process.

# Debug

## : Start Debugging

### 12. Debug 결과 확인

- ✓ 상단의 '**Resume**' 버튼을 클릭하여 디버깅을 위한 보드 동작 실행.
- ✓ 현재 'Cpu0\_Main.c' 파일에 코드를 작성하지 않았기 때문에 아무 출력이 없음.



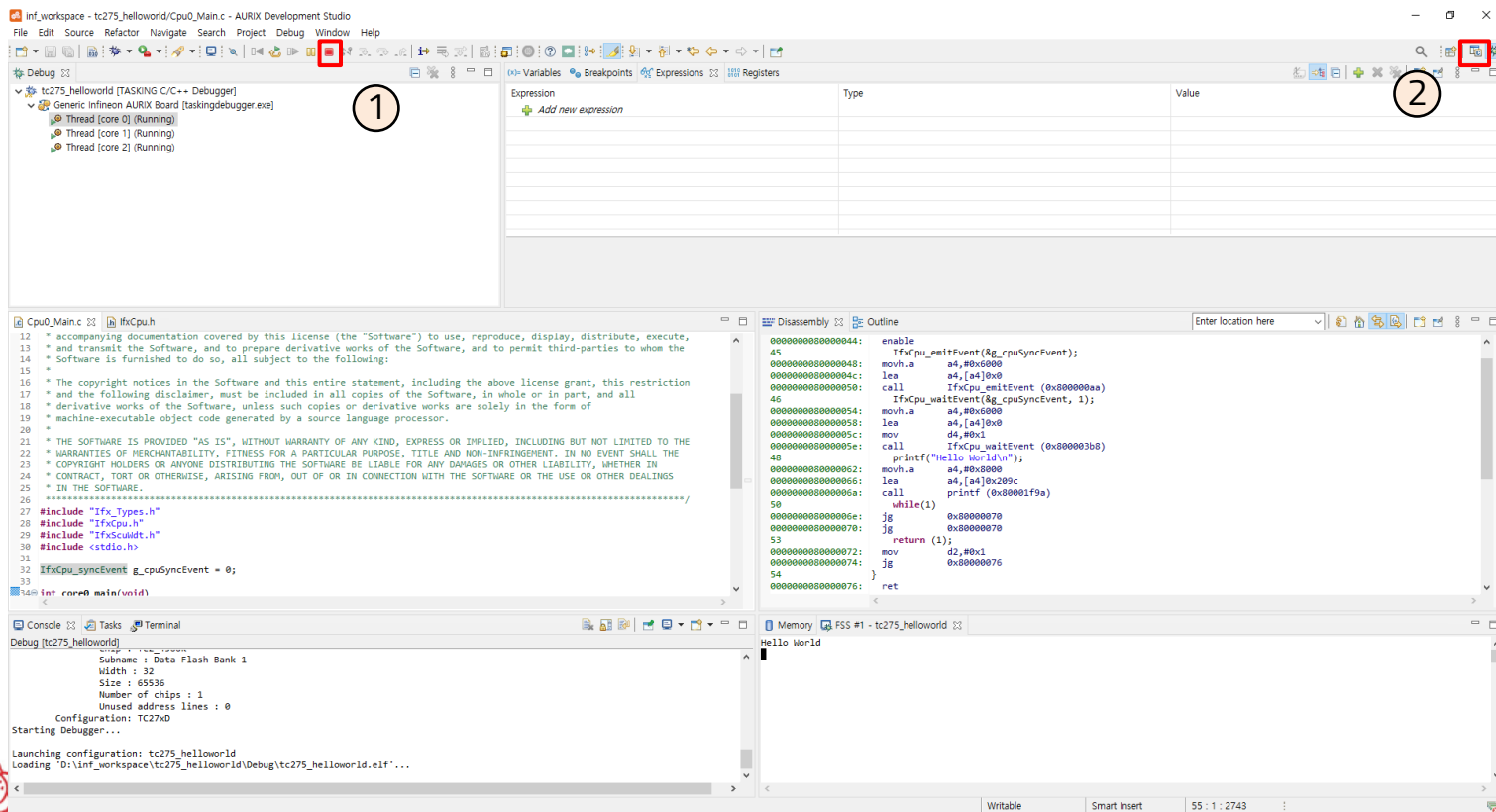
# Debug

## 13. Debug를 종료한다.

- ✓ 상단의 '**Terminate**' 버튼을 클릭하여 Debug를 종료한다.

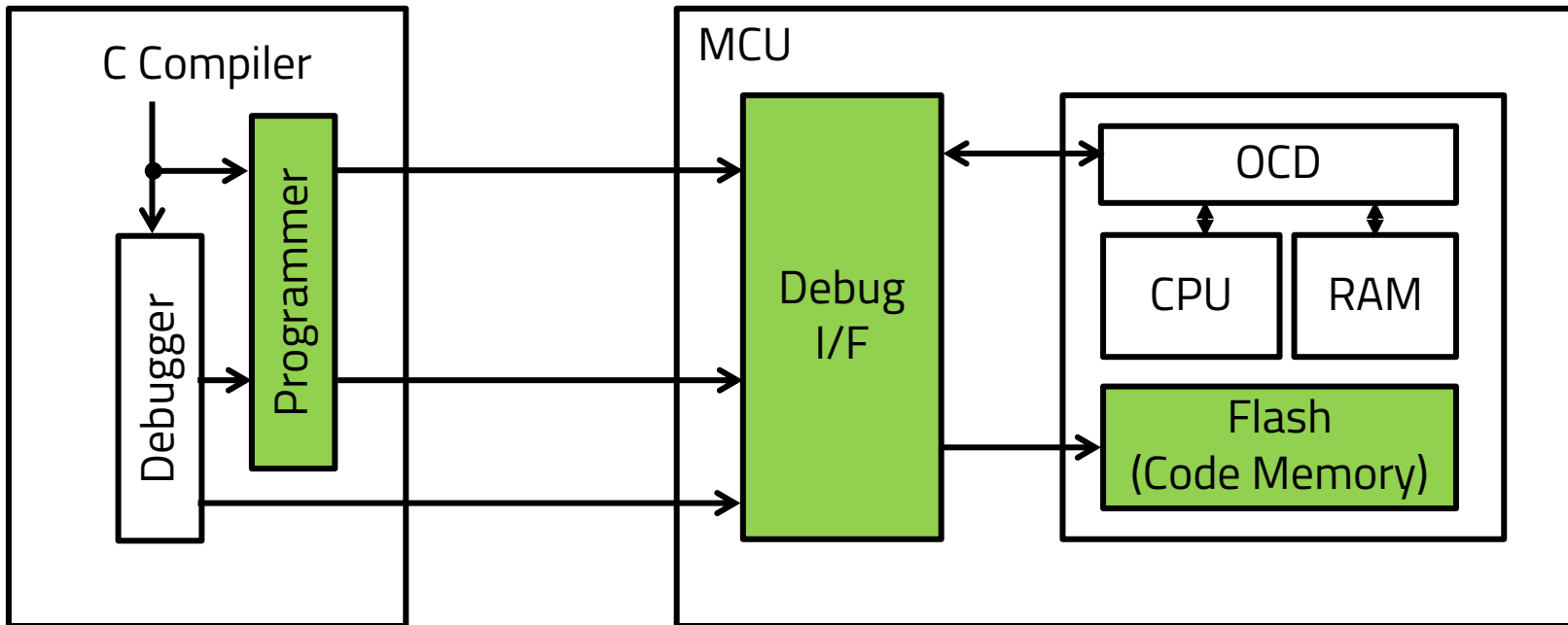
(Debug 종료 시, 반드시 'Terminate' 버튼을 클릭하여 정상적으로 종료한다.)

- ✓ Debug 종료 후, 소스코드 편집 layout으로 돌아가기 위해서는 우측 상단의 'C/C++' 클릭 (벌레 모양 버튼 왼쪽)



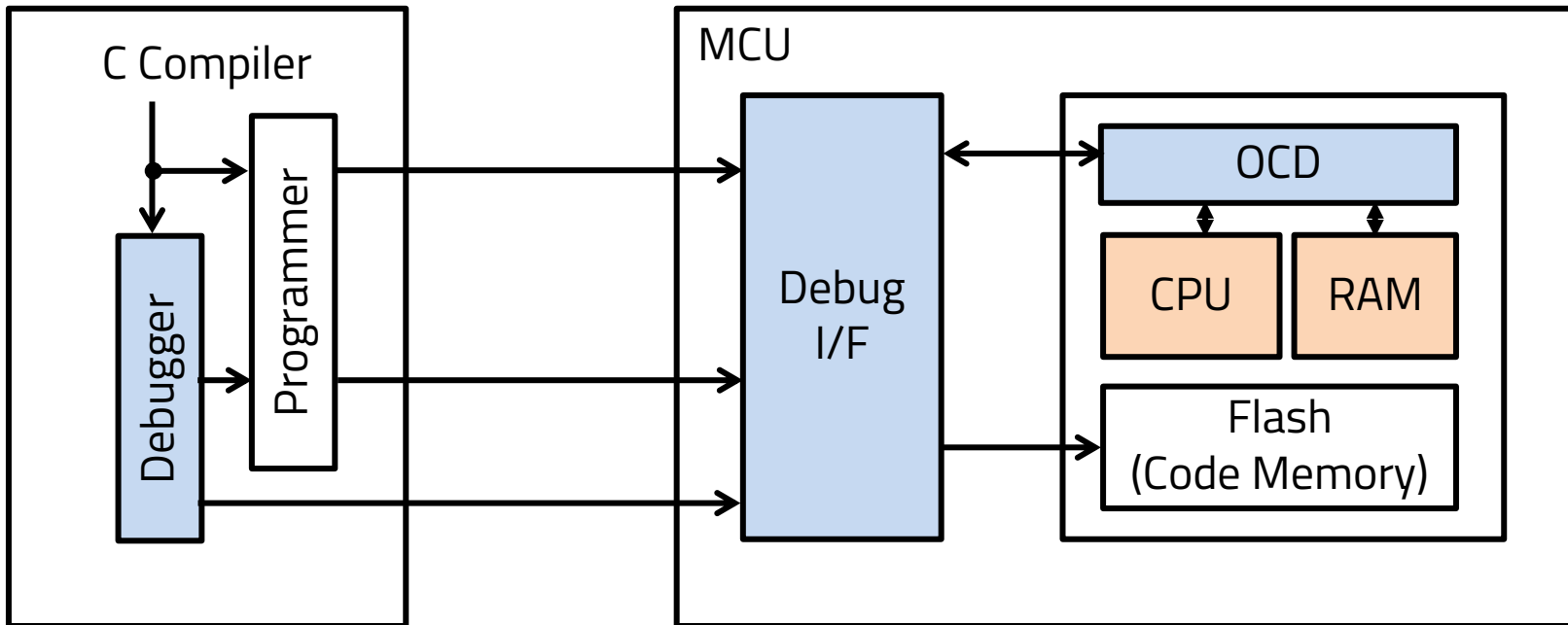
# Programmer(Downloader) vs Debug

## Download Stage



# Programmer(Downloader) vs Debug

## Debugging Stage



감사합니다. 휴식~~

## FAQ

- 처음 시작할때 Resume
- 디버깅 끝나고 다시 에디터로 돌아갈때 Terminate