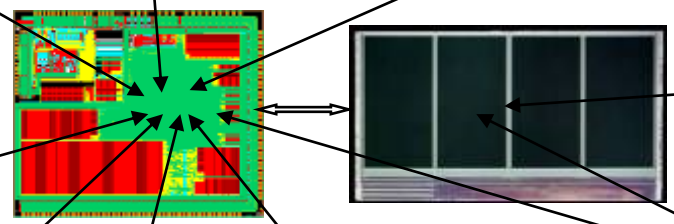
임베디드 MCU 프로그래밍 실습

AURIX TC275 보드 프로젝트 생성

현대자동차 입문교육
박대진 교수

하드웨어-소프트웨어 통합된 임베디드 시스템



Security
Encryption/Decryption
Smart Card Security
Java Card Support
DRM, e-commerce

Contactless
RFID (Radio Freq. ID)
NFC (Near Field Contact)

CPU
ASIC Technology
32 Bit RISC μP
>100 MHz

Flash Management
Error Correction
Wear Levelling
Bad Block Management
Garbage Collection
Data Refresh Capabilities
Meta Data
Trimming

Memory
SLC/MLC x3/x4
NAND

Signal Processing
Compression/Decompression

Performance
Parallelism

OS
Real Time Operating System

AI Applications
Launch or Internal Execute
i.e. Mobile Phones, Payment, Security

S/W H/W 구현 과정

Debugging

```
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        if (iN == (iLength - 3) ) {
```

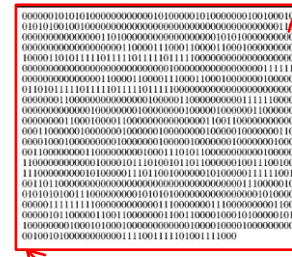
C language

Compilation

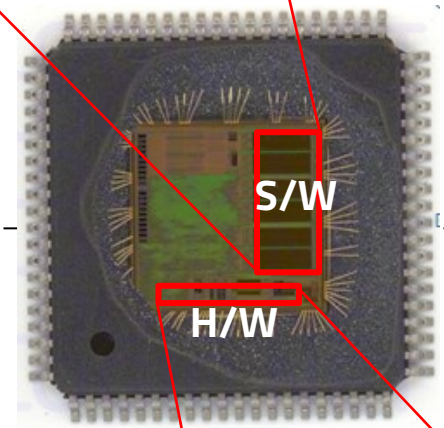
```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax, al
retn
```

Assembly code
(Machine code)

Assembling



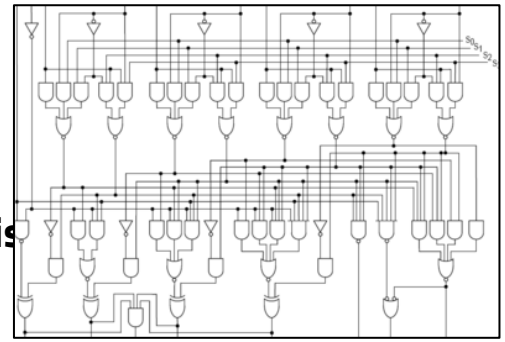
Binary code
(App)



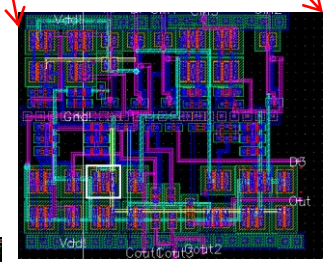

```
module alu (func, a, b, c);
 input [1:0] func;
 input [3:0] a, b;
 output [3:0] c;
 reg [3:0] c;        // so it can be assigned in alwa

 task my_and;
   input[3:0] a, b;
   output [3:0] andout;
   integer i;
   begin
     for (i = 3; i >= 0; i = i - 1)
       andout[i] = a[i] & b[i];
   end
 endtask

 always @ (func or a or b) begin
   case (func)
     2'b00: my_and (a, b, c);
     2'b01: c = a | b;
     2'b10: c = a - b;
     default: c = a + b;
   endcase
 end
endmodule
```

H/W Design

Synthesis



Place &
Route
(Layout)



Simulation

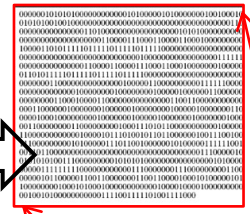
임베디드 시스템 프로그래밍 → 0/1을 온칩에 임베딩

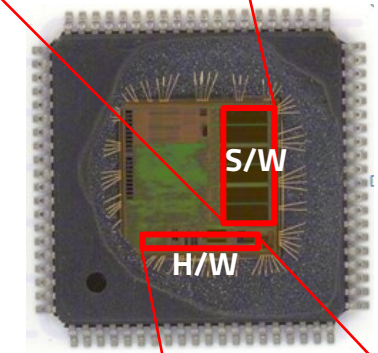
- Software is deeply injected into the hardware silicon
  - Your dirty software code … is directly translated into the machine code.
  - So that performance degradation starts from inefficiency of my code.

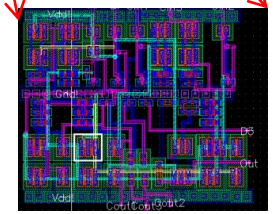
Compilation/Assembling/Linking/Downloaded

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

}
```

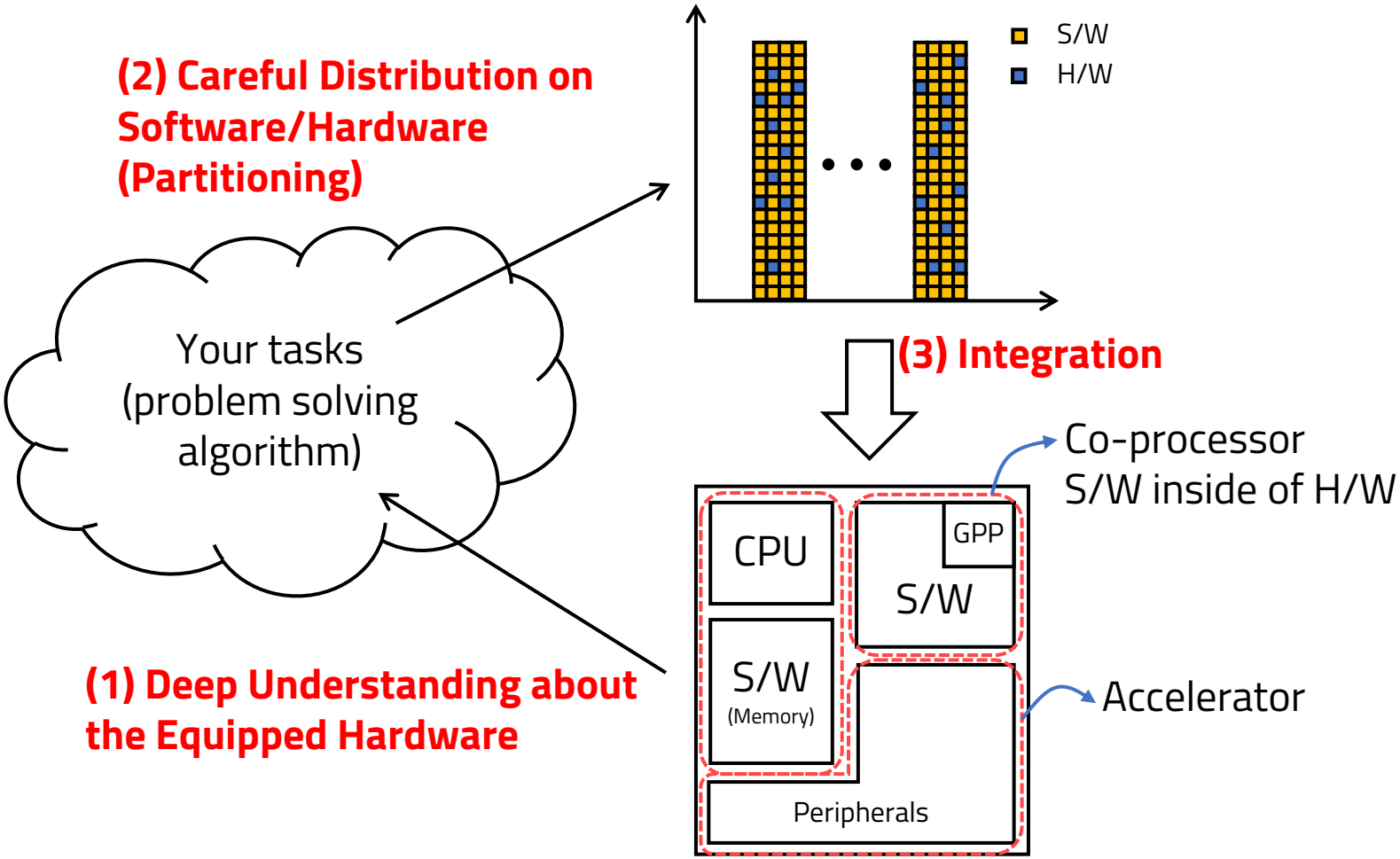
```
004013e0 <_main>:

int main() {
  4013e0:   55                      push   %ebp
  4013e1:   89 e5                   mov    %esp,%ebp
  4013e3:   83 e4 f0                and    $0xfffffff0,%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

}
```







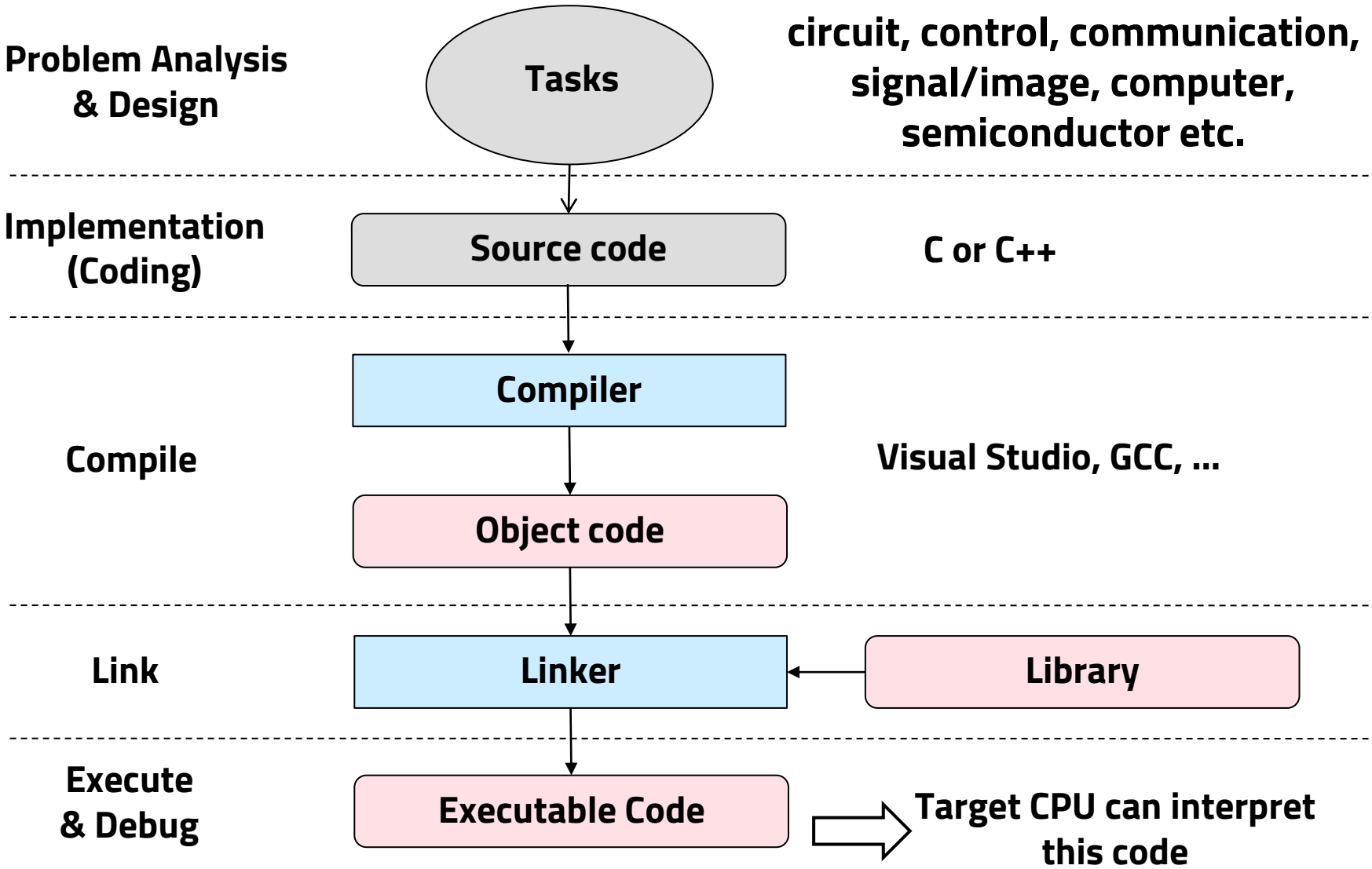
임베디드 S/W가 칩 내부의 H/W를 구동함



System On Chip (Latest Processor Architecture)

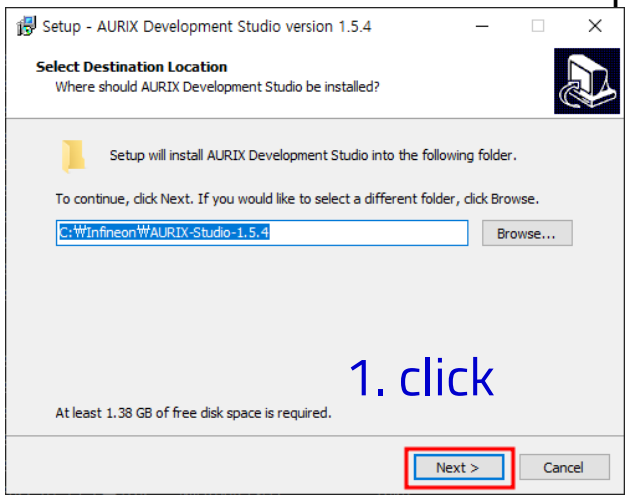
Compilation

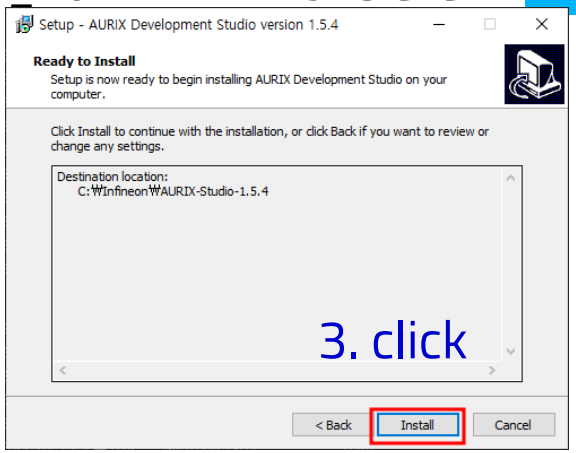
Tasks → Executable Binary Code

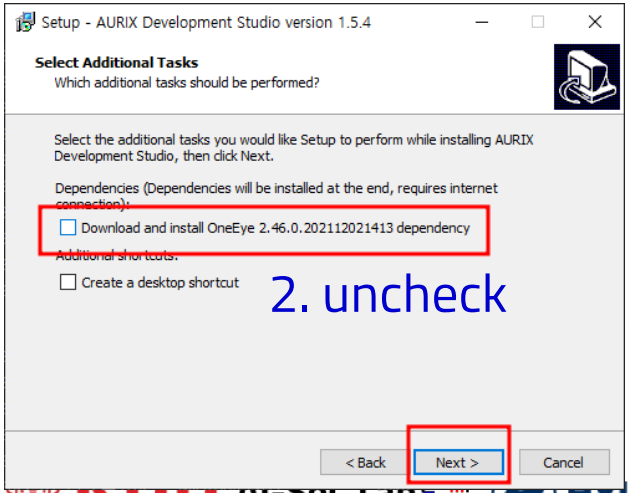


Installing Target-Specific Compiler

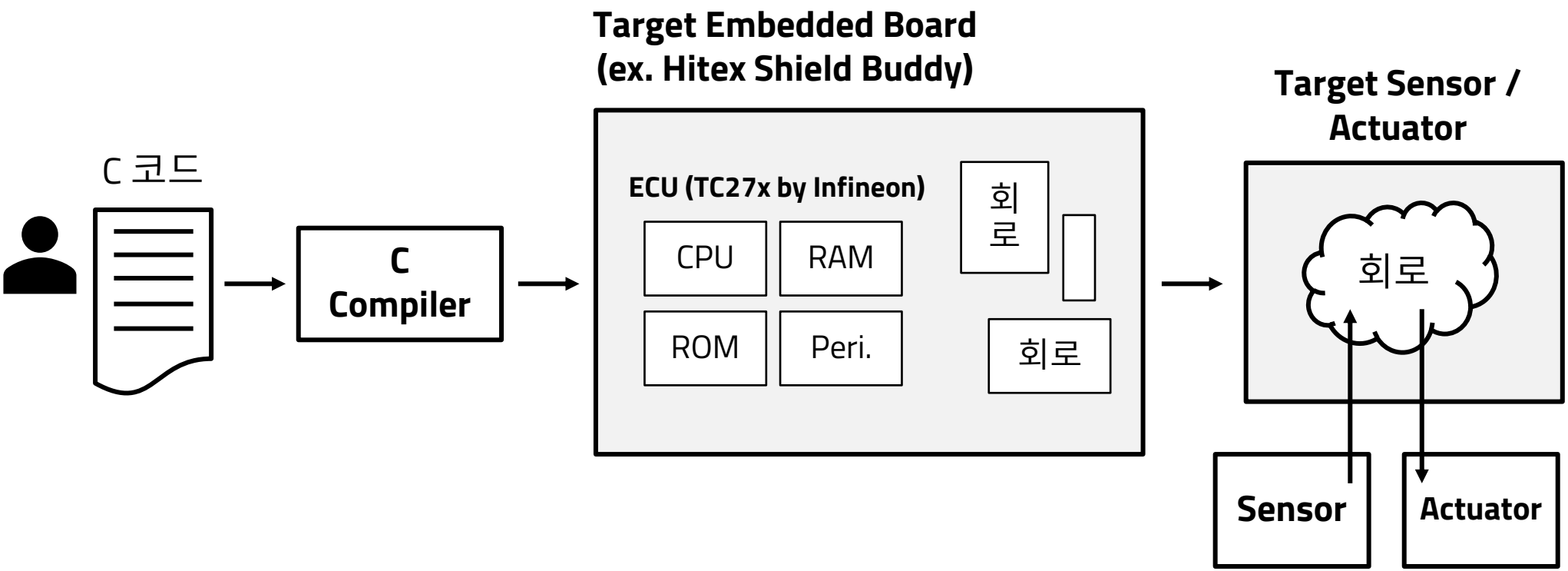
- AURIX™ Development Studio Install
  - AURIX-studio-setup_1.5.4_20211221-1518.exe







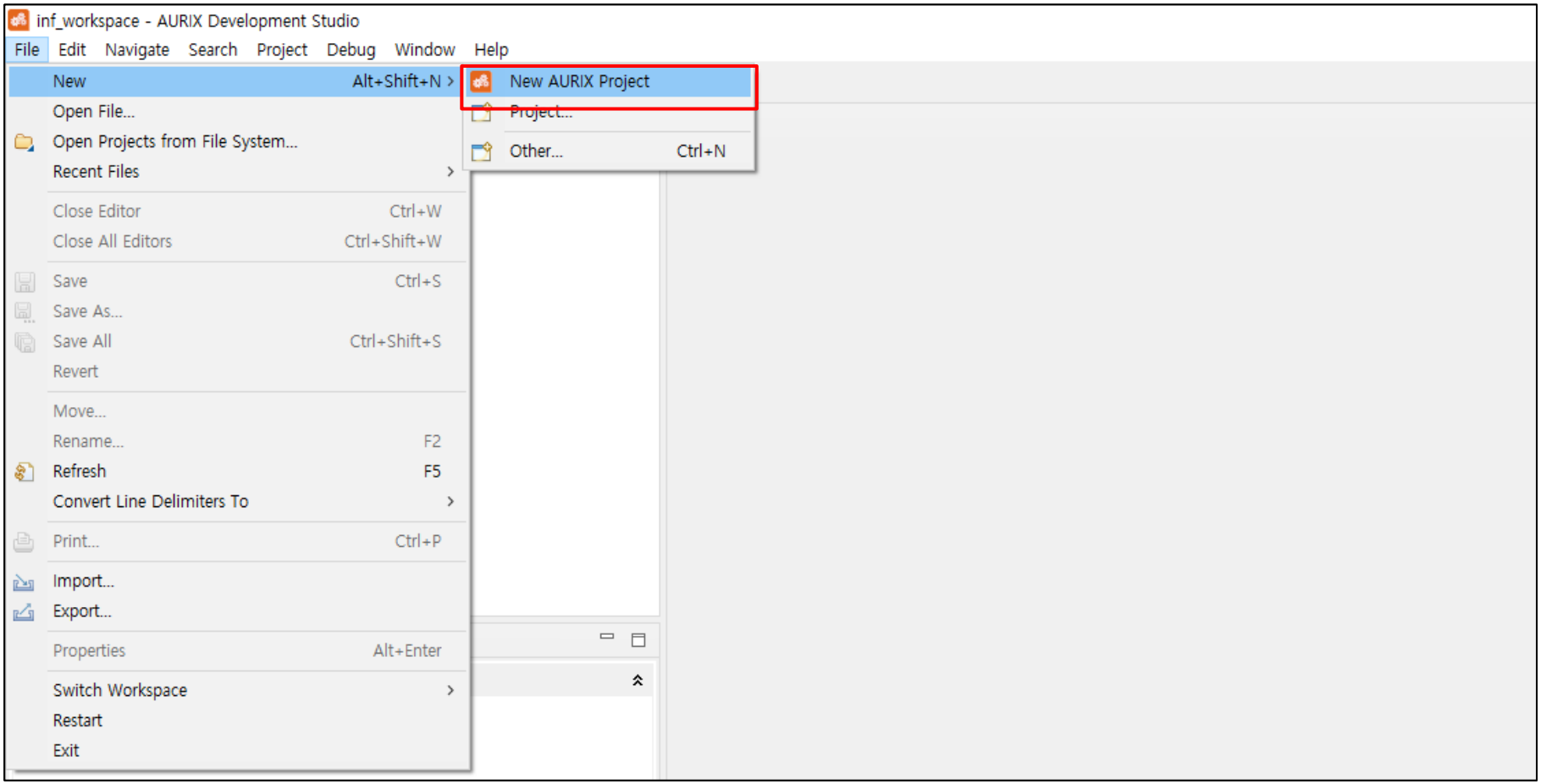
Embedded System Programming 대상



New Project

: Template Generation

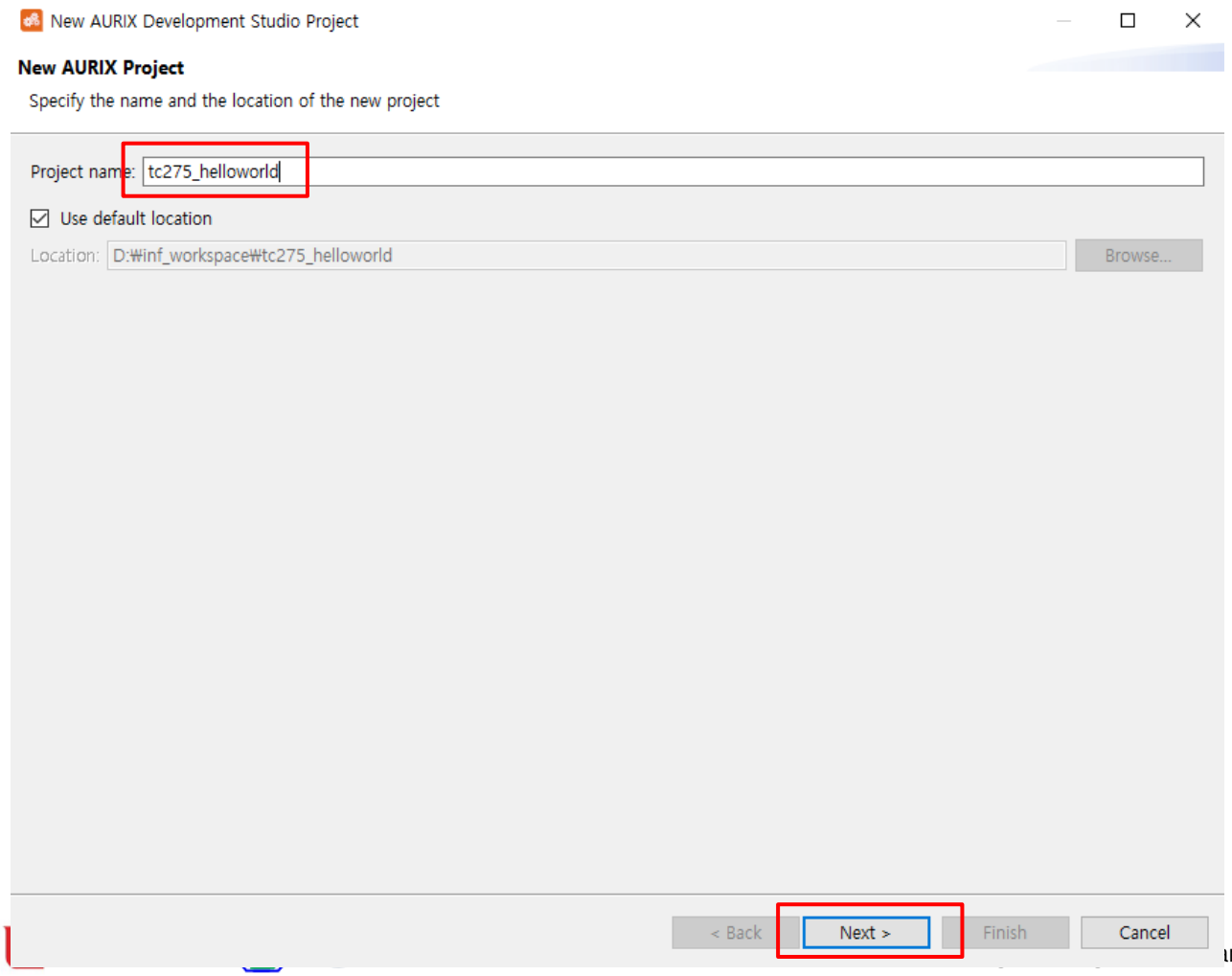
1. AURIX Development Studio를 실행하고 왼쪽 상단의 **'File – New –New AURIX Project'** 를 클릭한다.



New Project

: Project Name

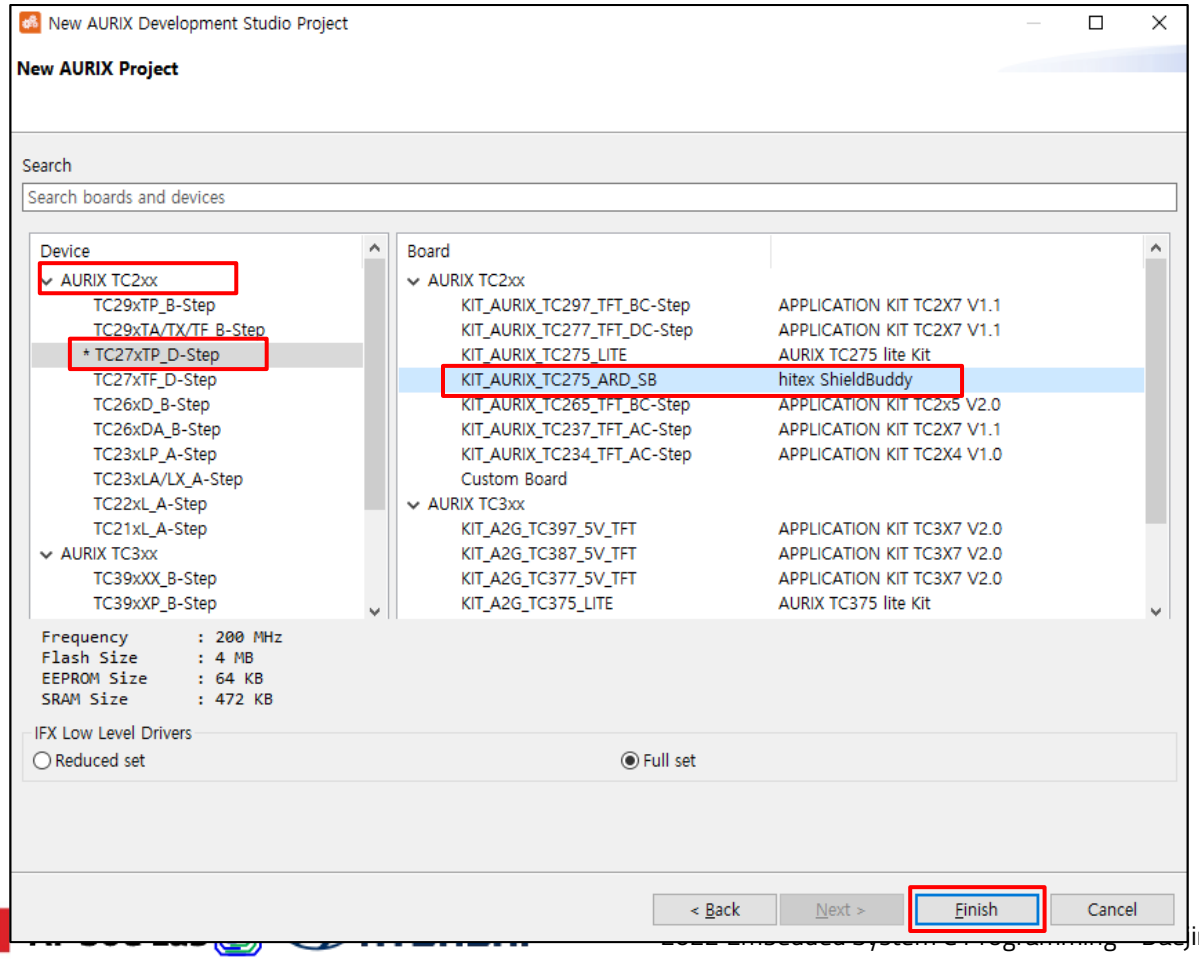
2. 생성할 Project의 이름을 입력하고, **'Next'**를 클릭한다.



New Project

: Target CPU, Target Board Selection

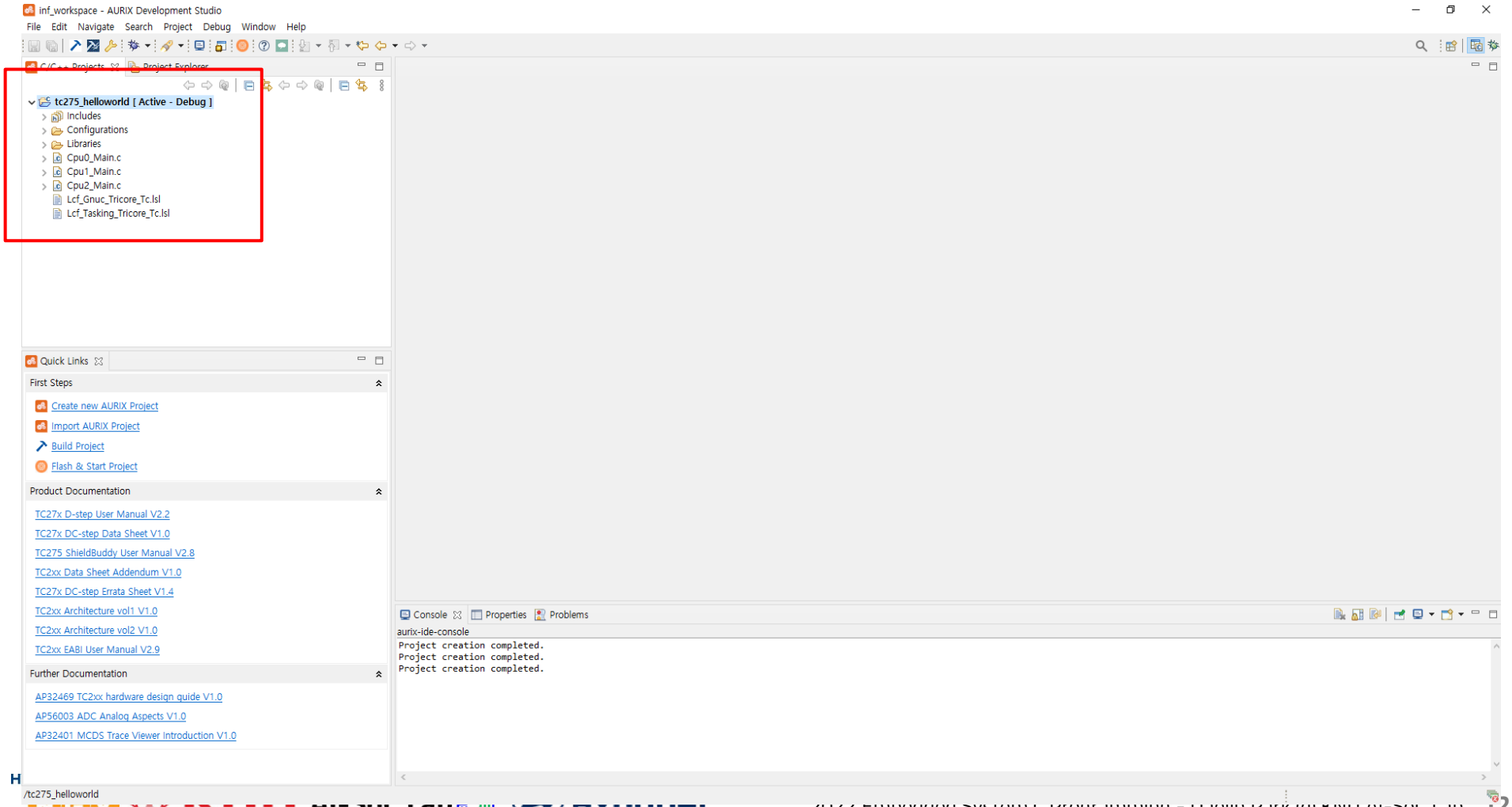
3. Device에서 **'AURIX TC2xx – TC27XTP_D-Step – KIT_AURIX_TC275_ARD_SB'** 를 선택하고, 다른 설정은 그대로 유지한 채 **'Finish'**를 클릭한다.



New Project

: Code Explorer

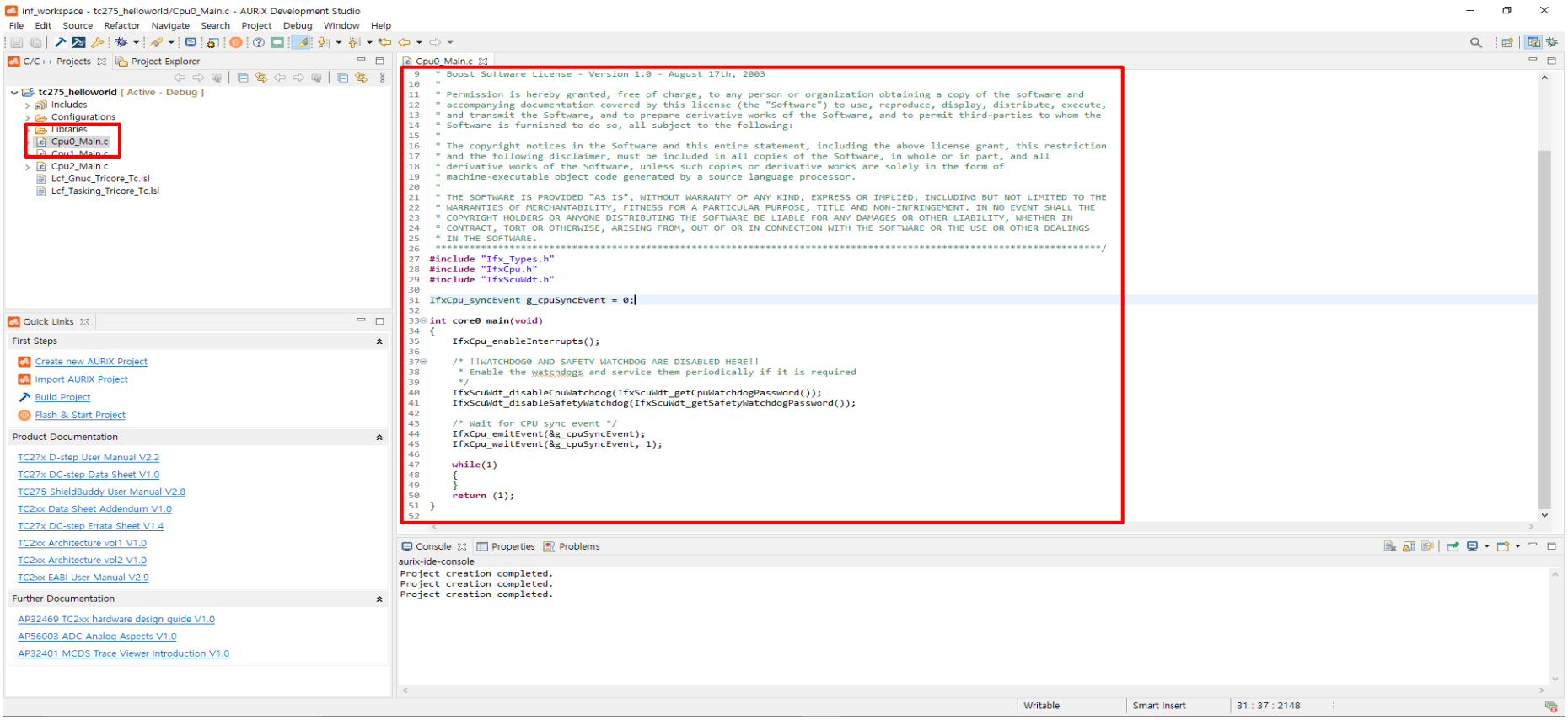
4. 왼쪽의 Project Explorer 창에서 프로젝트가 생성된 것을 확인한다.



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 포함**하고 있으며 이를 수정하여 보드의 동작을 설계한다.

AURIX - aaa/Cpu0_Main.c - AURIX Development Studio

File Edit Source Refactor Navigate Search Project Debug Window Help

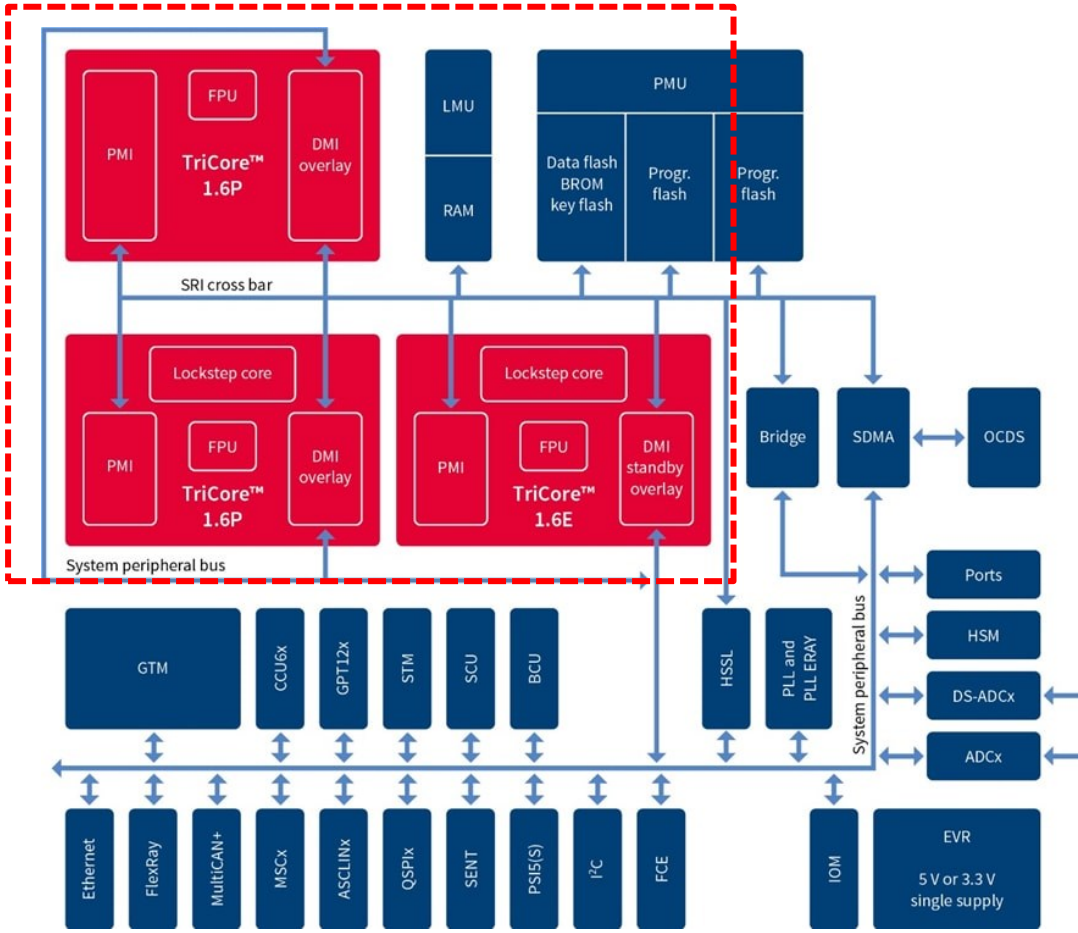
Cpu0_Main.c

```c
/*********************************************************************************************************
 * \file Cpu0_Main.c
 * \copyright Copyright (C) Infineon Technologies AG 2019
 *
 * Use of this file is subject to the terms of use agreed between (i) you or the company in which ordinary course of
 * business you are acting and (ii) Infineon Technologies AG or its licensees. If and as long as no such terms of use
 * are agreed, use of this file is subject to following:
 *
 * Boost Software License - Version 1.0 - August 17th, 2003
 *
 * Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and
 * accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute,
 * and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the
 * Software is furnished to do so, all subject to the following:
 *
 * The copyright notices in the Software and this entire statement, including the above license grant, this restriction
 * and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all
 * derivative works of the Software, unless such copies or derivative works are solely in the form of
 * machine-executable object code generated by a source language processor.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE
 * COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN
 * CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
 * IN THE SOFTWARE.
 *********************************************************************************************************/
#include "Ifx_Types.h"
#include "IfxCpu.h"
#include "IfxScuWdt.h"

IfxCpu_syncEvent g_cpuSyncEvent = 0;

int core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    while(1)
    {
    }
    return (1);
}
```

Edit Project
: main functions for Tri Core

- AURIX TriCore TC275 MCU는 3개의 CPU를 갖고 있지만, 본 실습에서는 1개의 CPU만을 사용. (CPU0)

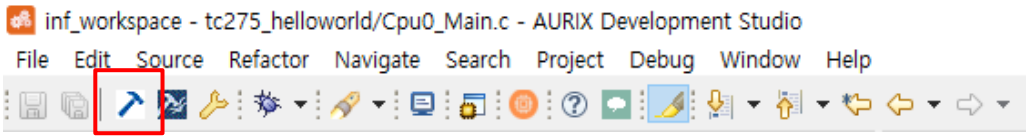
- Cpu1_Main.c 와 Cpu2_Main.c 파일들은 사용하지 않음.

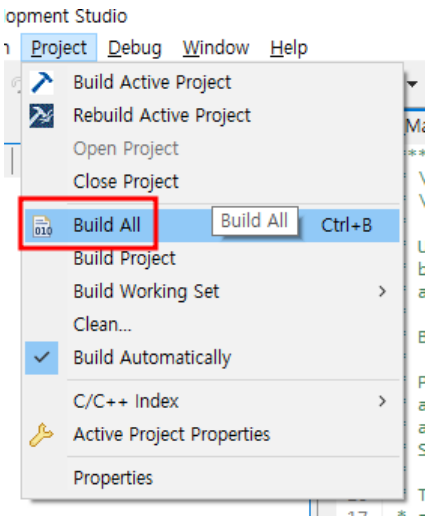


AURIX TC27xT 32-bit TriCore Microcontrollers

Build

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

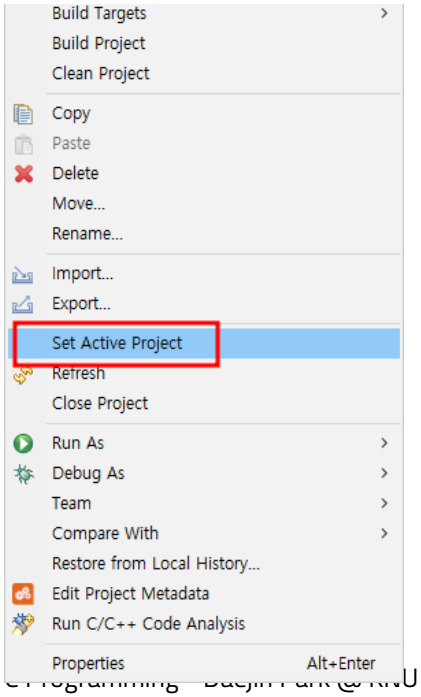




8. (Build/Debug는 Active Project에 대해 수행

Build를 수행할 Project를 Active Project로 미리 설정해야 한다.

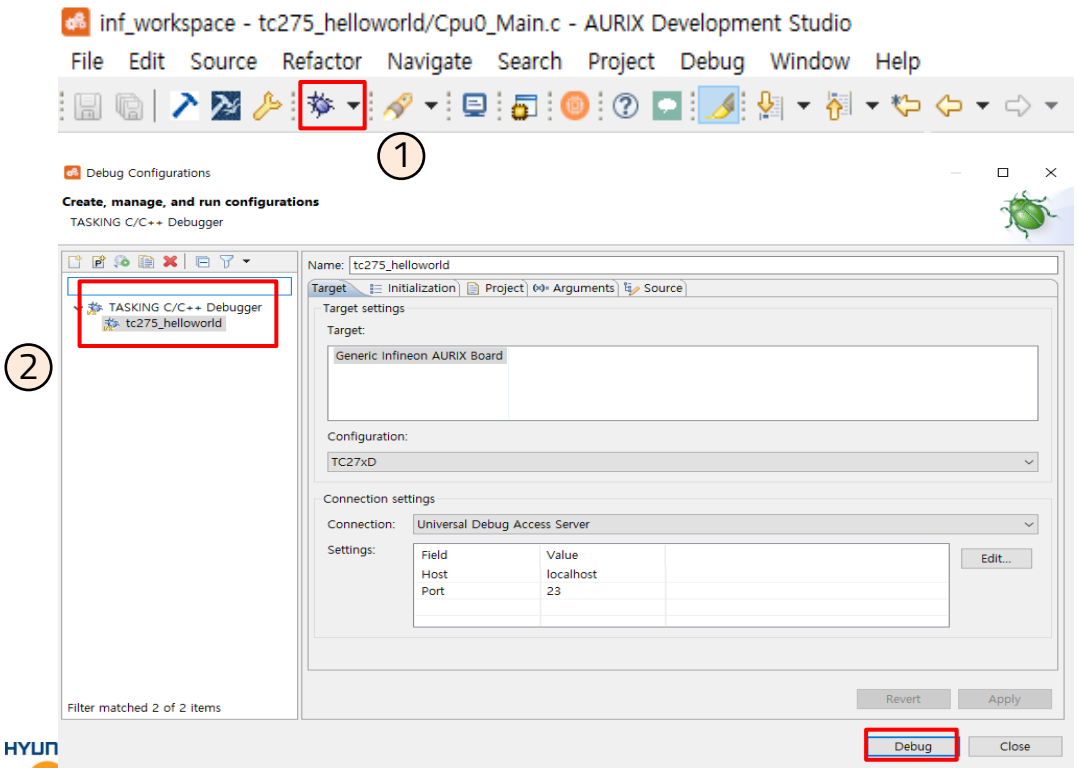
'Project Explorer' 에서 대상 프로젝트를 우클릭 한 뒤, 'Set Active Project'를 클릭하여 Active Project로 설정할 수 있다.)



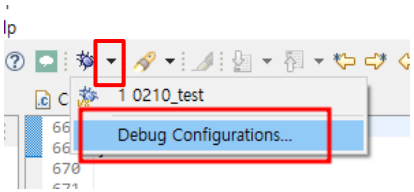
Debug

: 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)

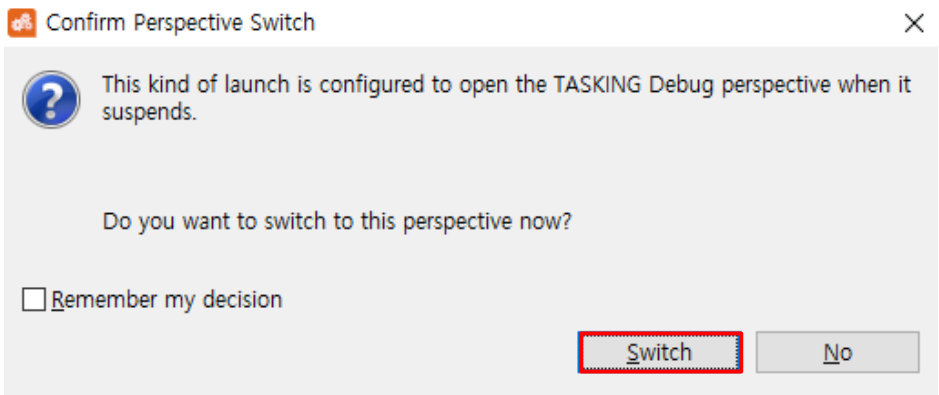


다시 'Debug Configurations' 창을 활성화하려면 'Debug' 버튼 오른쪽 화살표 클릭
→ 'Debug Configurations...' 버튼 클릭



Debug

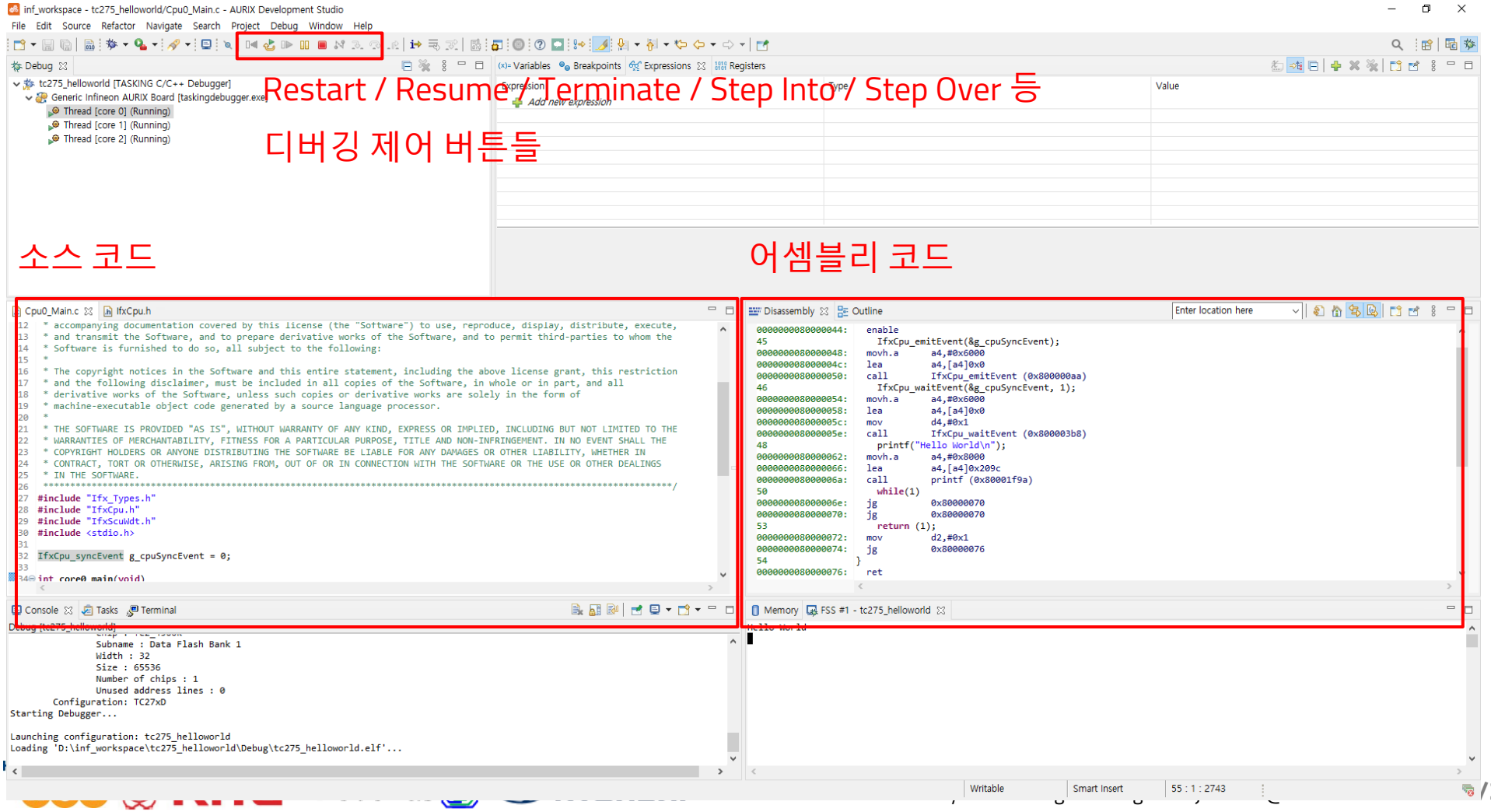
10. 상단의 메뉴에서 **'Debug'** 버튼을 클릭하여 Debug를 실행한다.
    - ✓ 'Confirm Perspective Switch' 창이 뜨면 Switch를 눌러 디버그 창으로 전환.
    - ✓ (소스 코드 외에 편리한 디버깅을 위한 여러 가지 window 를 포함하는 layout 으로 전환 여부)



Debug

: Control Panel, Debugging Window Layout

11. 기본 Debug 화면 구성

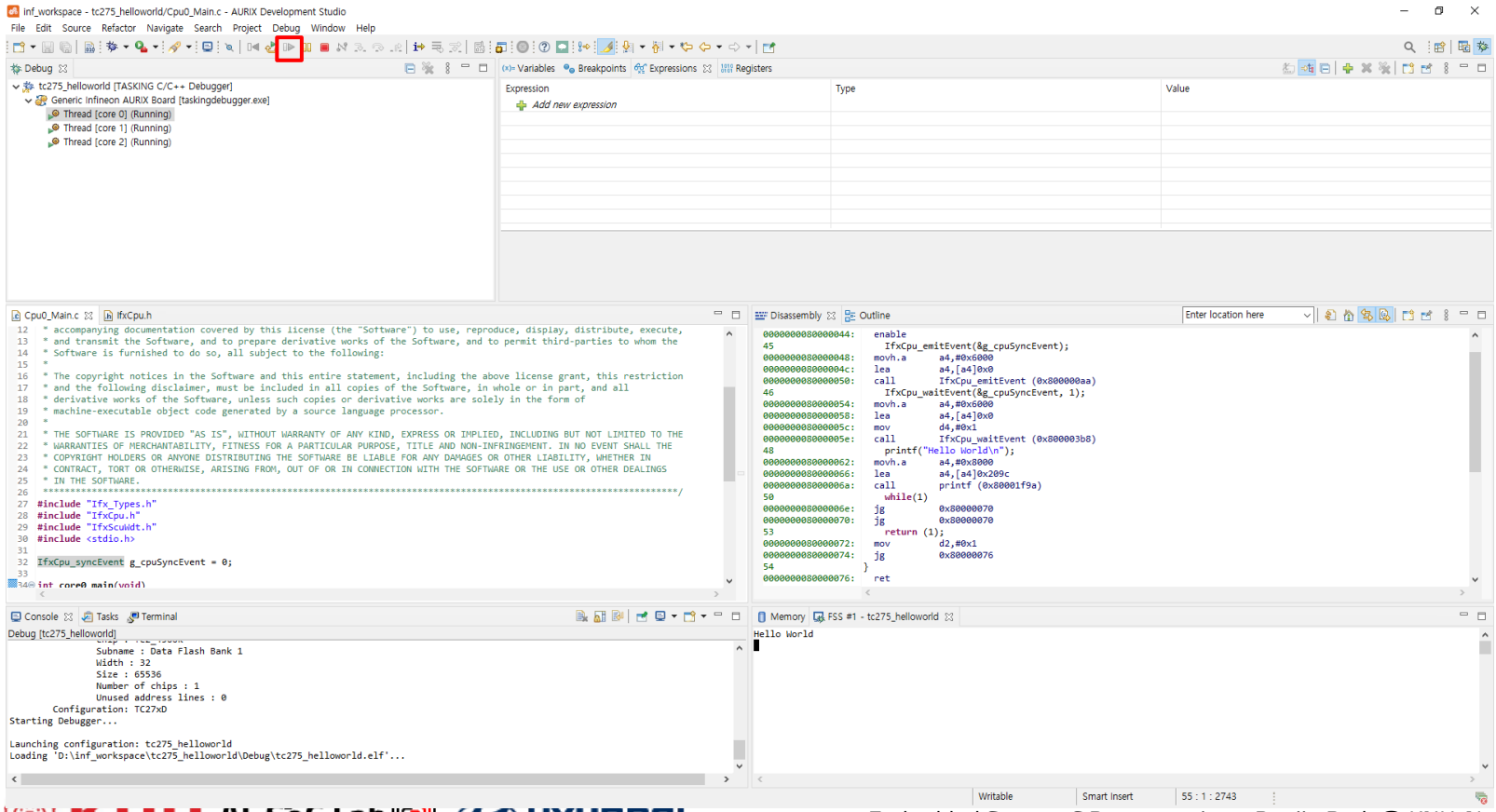


Debug

: 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.



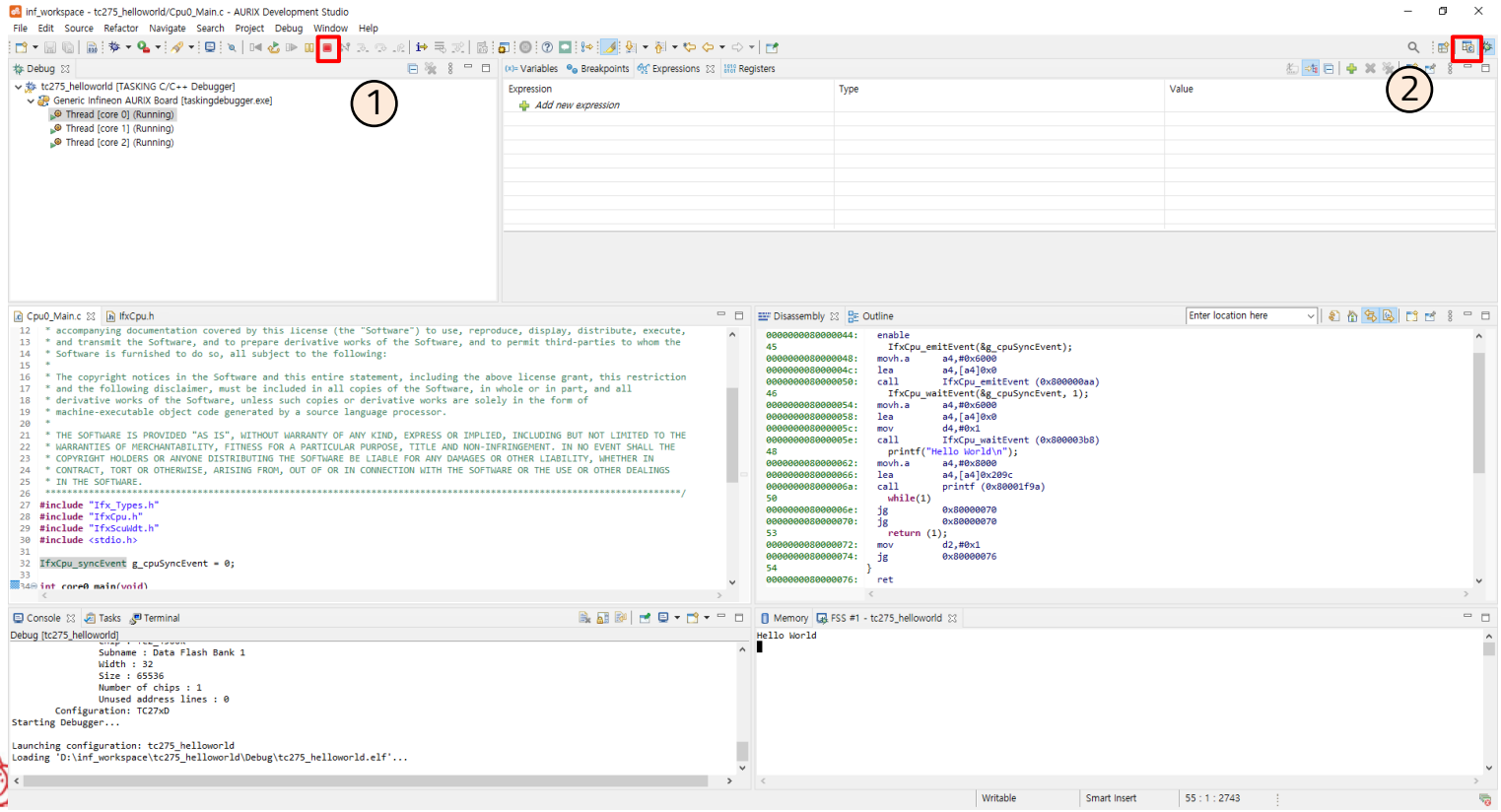
Debug

13. Debug를 종료한다.

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

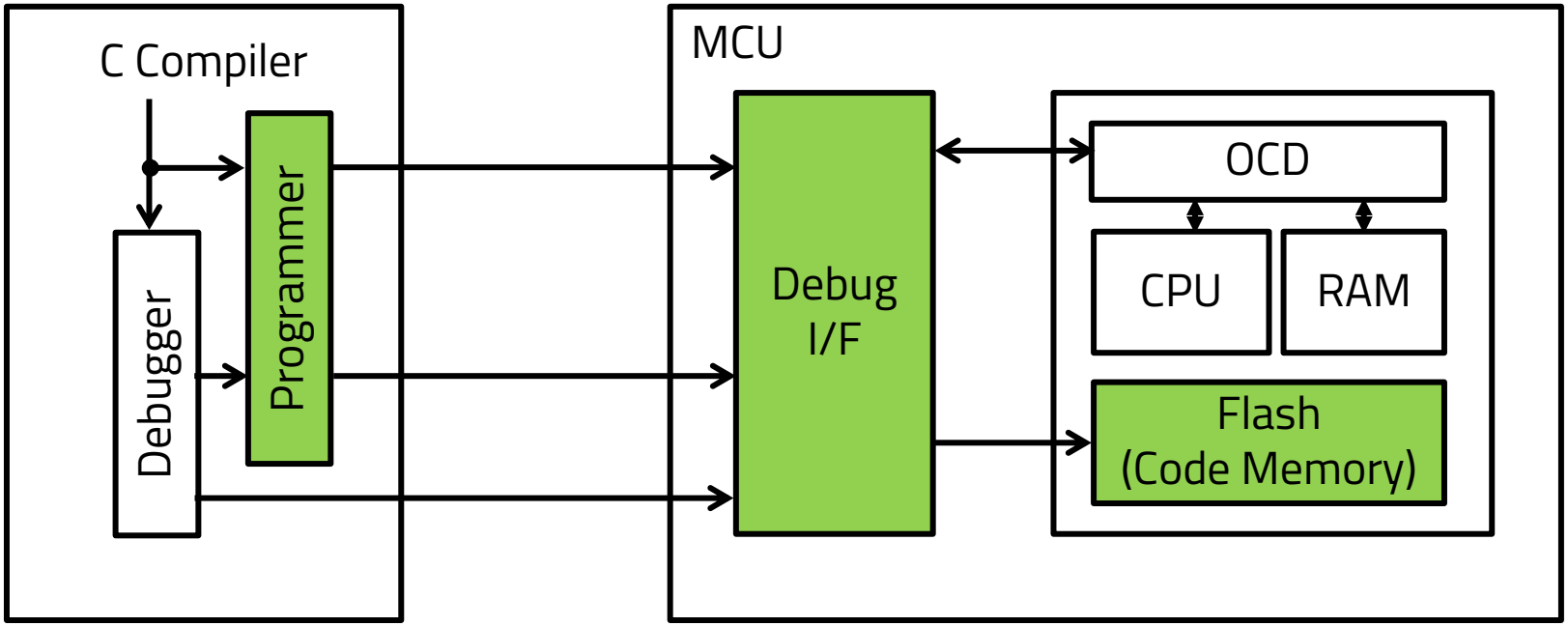
  (Debug 종료 시, 반드시 'Terminate' 버튼을 클릭하여 정상적으로 종료한다.)

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



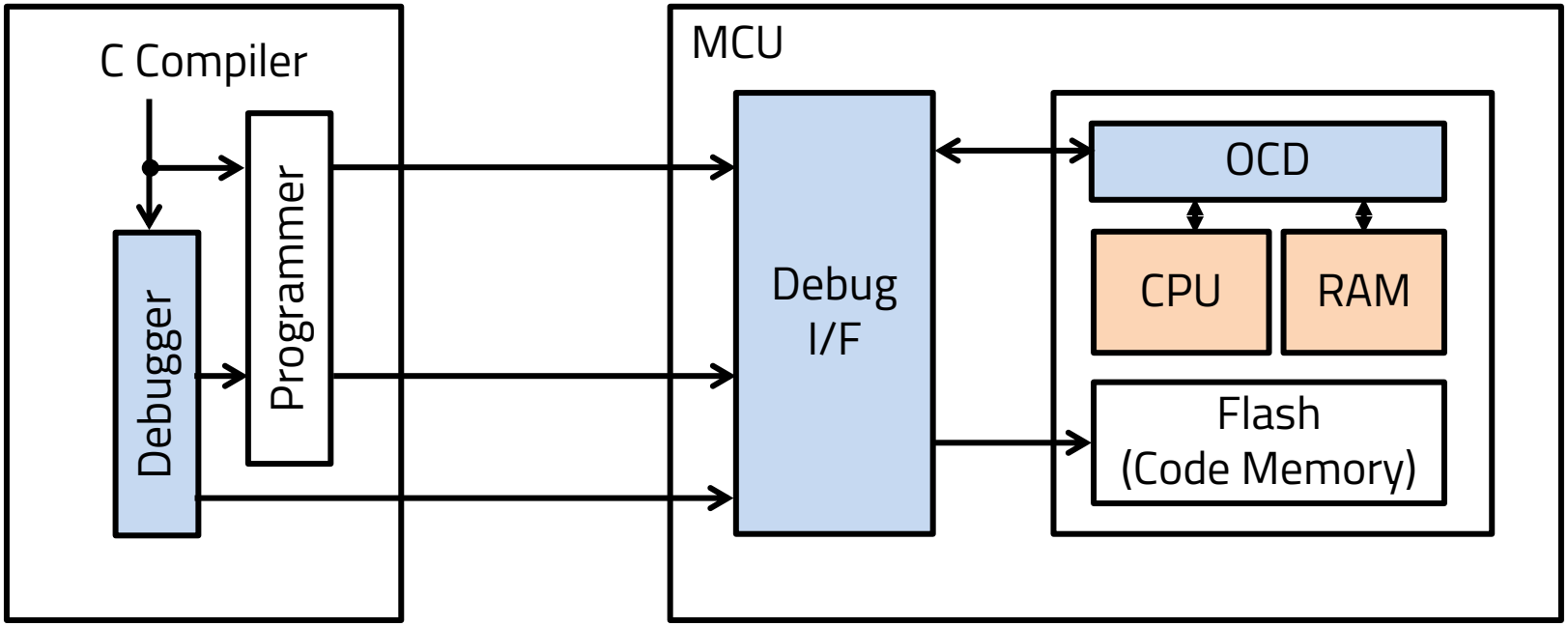
Programmer(Downloader) vs Debug

Download Stage



Programmer(Downloader) vs Debug

Debugging Stage



감사합니다. 휴식~~

FAQ

- 처음 시작할때 Resume
- 디버깅 끝나고 다시 에디터로 돌아갈때 Terminate