



Real-Time Operating System (Day 1 Lab)

Jong-Chan Kim

Graduate School of Automotive Engineering



국민대학교
KOOKMIN UNIVERSITY

Erika Enterprise



- 이탈리아 EVIDENCE에서 개발된 오픈소스 OSEK/VDX RTOS 구현체
- 듀얼 라이선스 정책 (오픈소스 라이선스 + 상용 라이선스)
- RTOS 연구 및 교육에 널리 사용
- GitHub 리포



evidence / erika3 Public

Watch 25 Fork 72 Star 155

Code Issues 2 Pull requests 1 Actions Projects Wiki Security Insights

master 1 branch 15 tags Go to file Add file Code

eguidieri [tricore] Fixed interrupt regression in... 85c445d on 13 Sep 2019 245 commits

contrib	[kernel] PostTaskHook called twice fixed and Initial ...	3 years ago
doc	[kernel] Fixed WaitEvent, in case of multicore enviro...	4 years ago
mk	[kernel] Fixed WaitEvent, in case of multicore enviro...	4 years ago
pkg	[tricore] Fixed interrupt regression introduced by co...	3 years ago
LICENSE.TXT	Update LICENSE to all files	4 years ago
README.md	[kernel] Fixed WaitEvent, in case of multicore enviro...	4 years ago
THIRDPARTY.TXT	[THIRDPARTY] Rephrased and fixed the statement r...	3 years ago

About

ERIKA Enterprise v3 RTOS

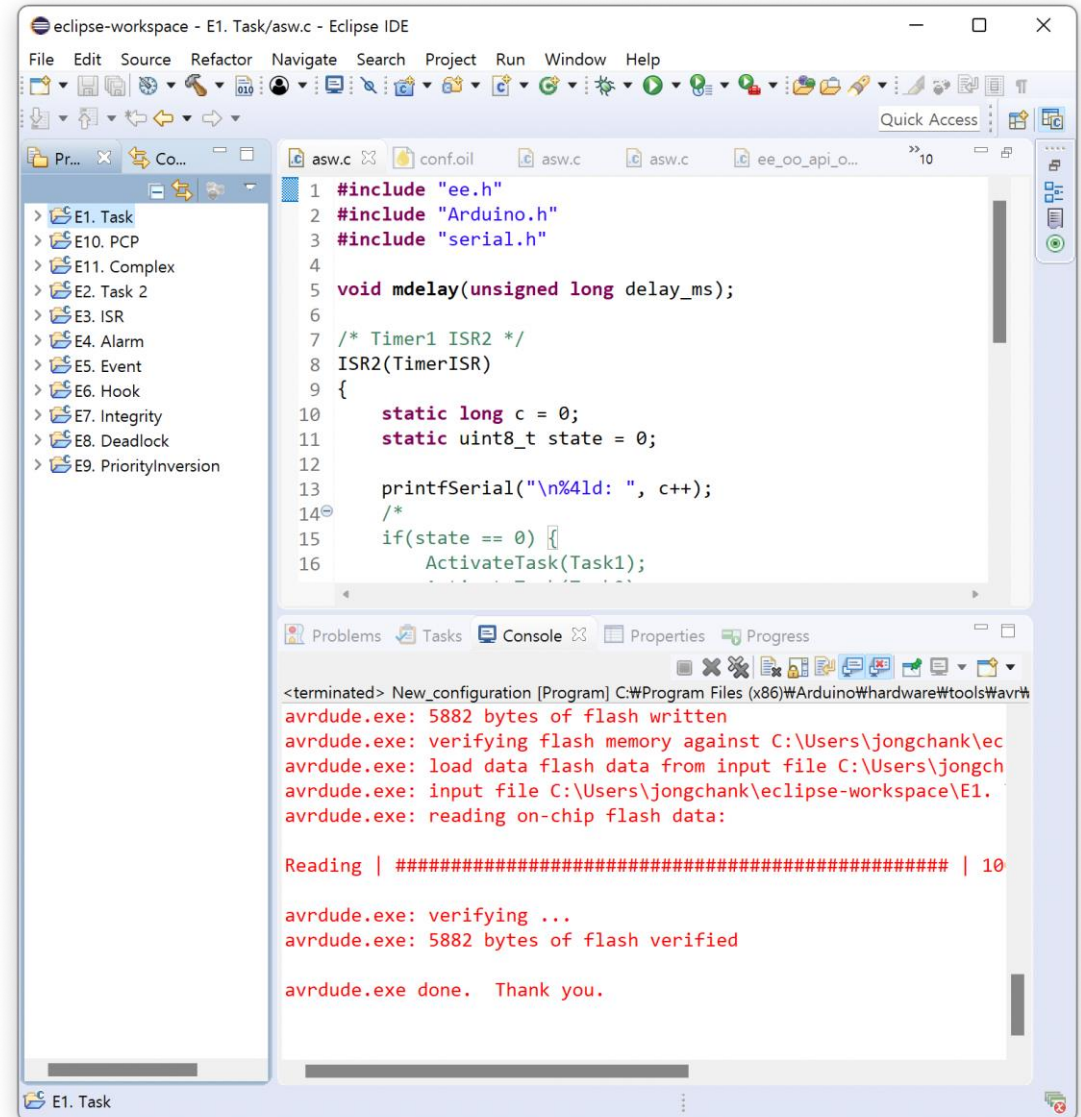
www.erika-enterprise.com

arduino avr arm real-time
cortex-m x86-64 xen rtos
cortex-a autosar osek dspic
aurix tricore-development cortex-r5f
erika-enterprise jailhouse

Readme
View license
155 stars
25 watching

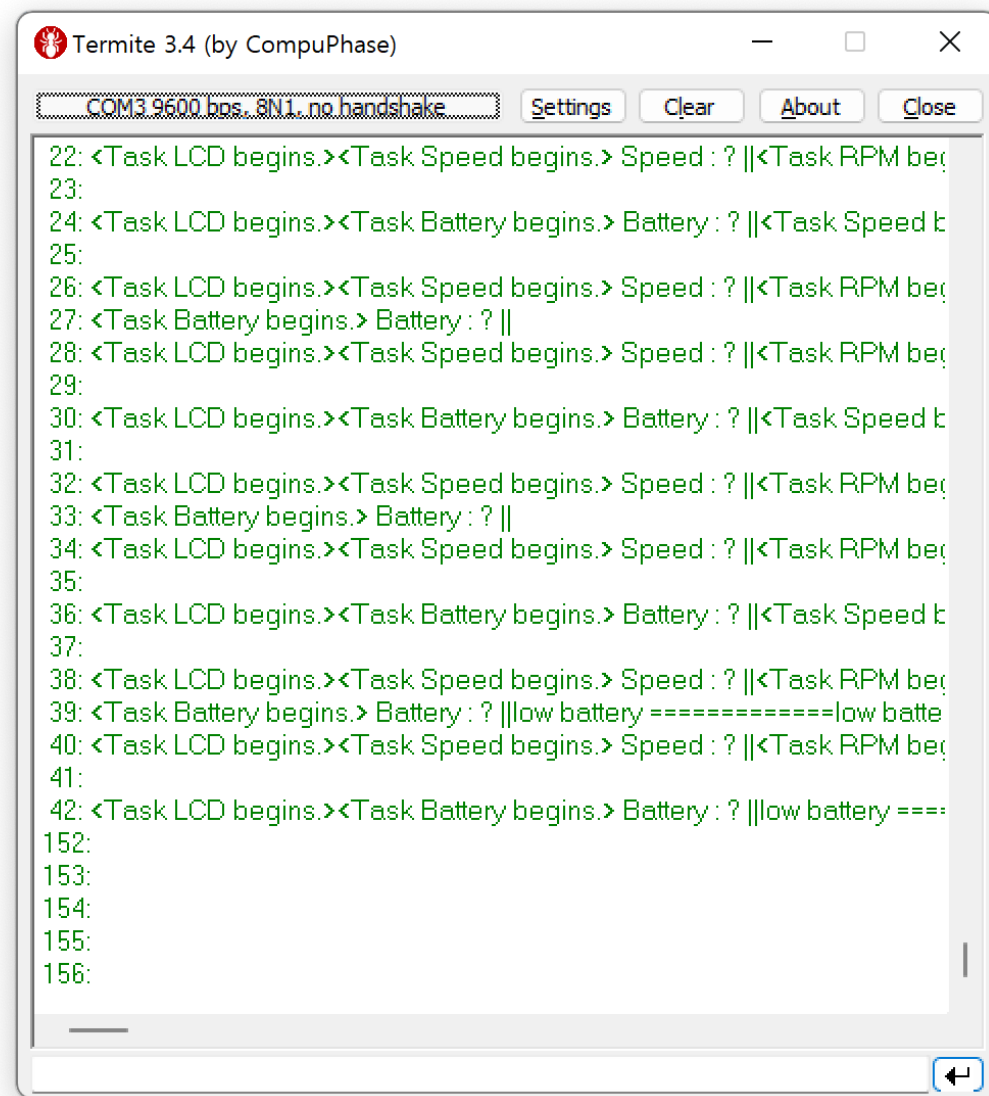
Eclipse 기반 IDE

- 프로젝트 생성
- OIL 파일, C/C++ 파일 편집
- 프로젝트 빌드
- 실행파일 다운로드



Termite 시리얼 콘솔

- 경량 시리얼 터미널



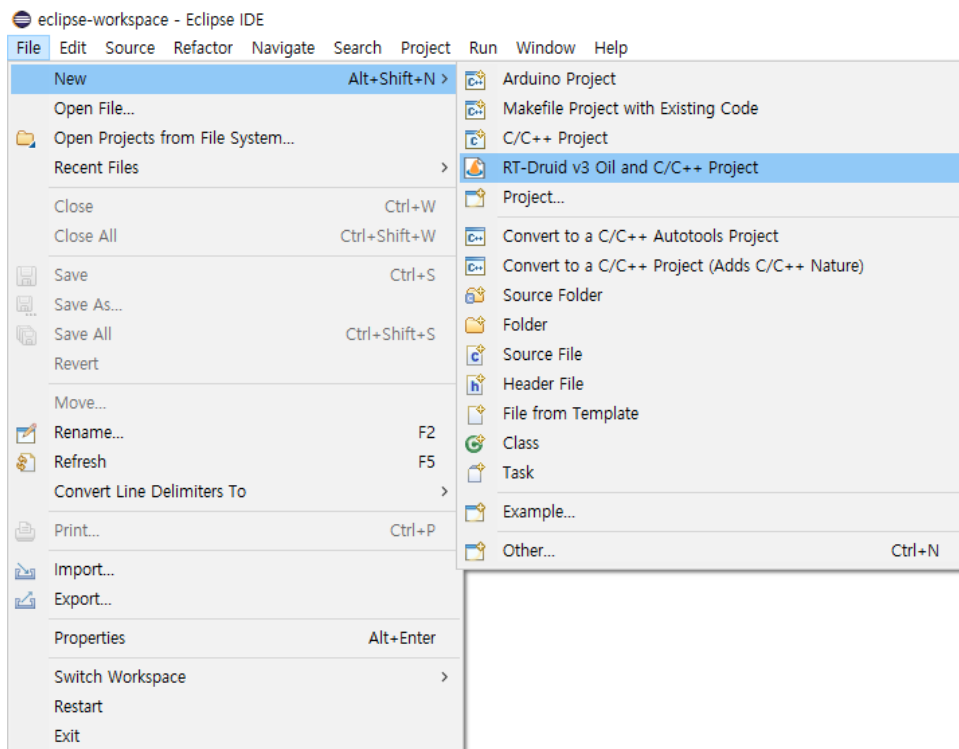
다운로드: https://www.compuphase.com/software_termite.htm

프로젝트 생성

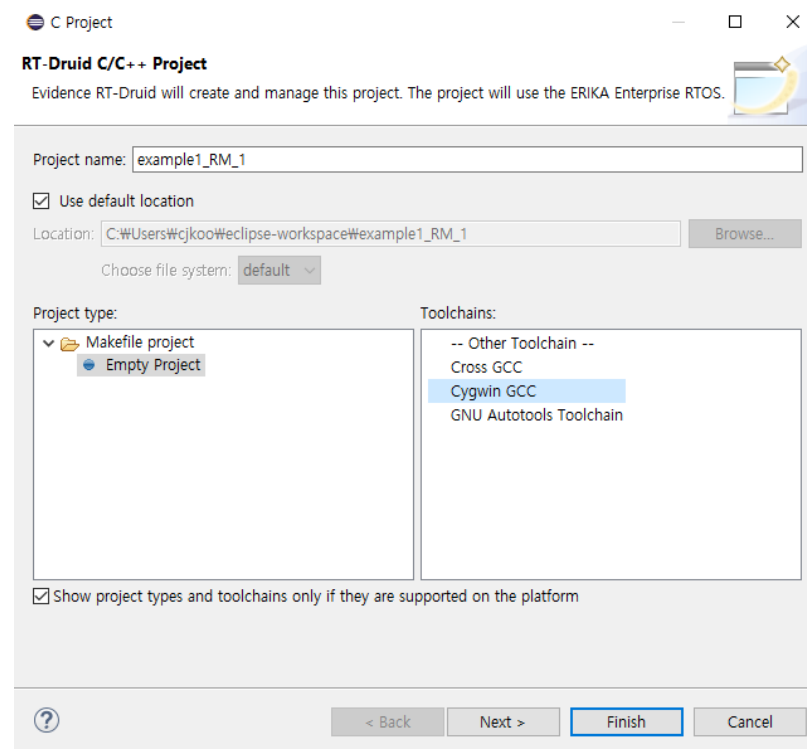
1. 새 프로젝트

File -> New

-> RT-Druid v3 Oil and C/C++ Project

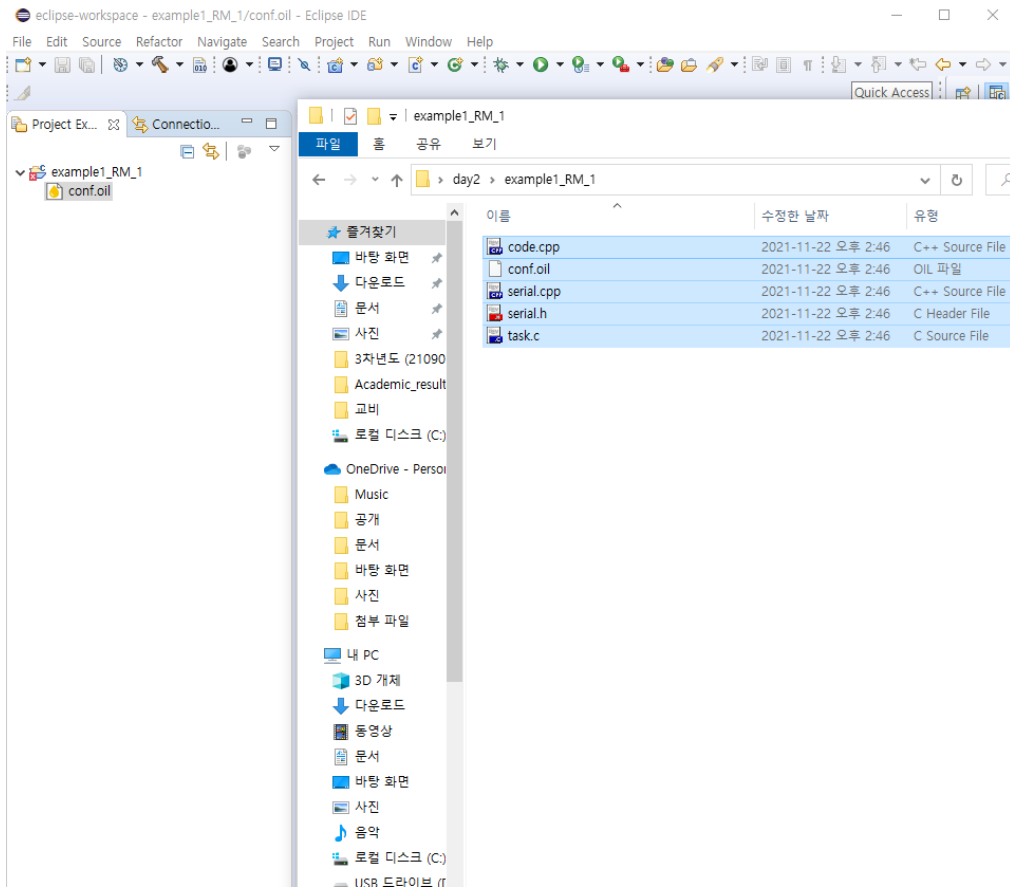


2. 프로젝트 이름 입력 및 Cygwin GCC 선택

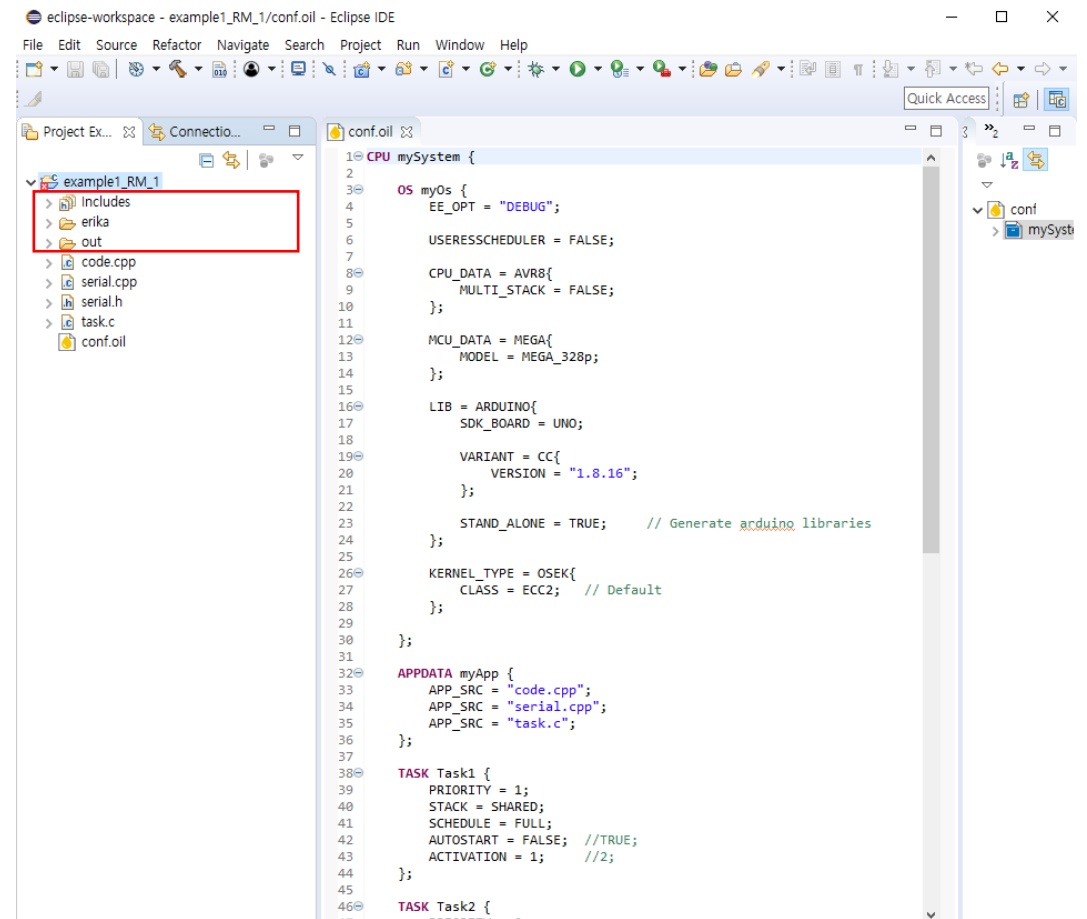


프로젝트 생성

3. 파일 카피

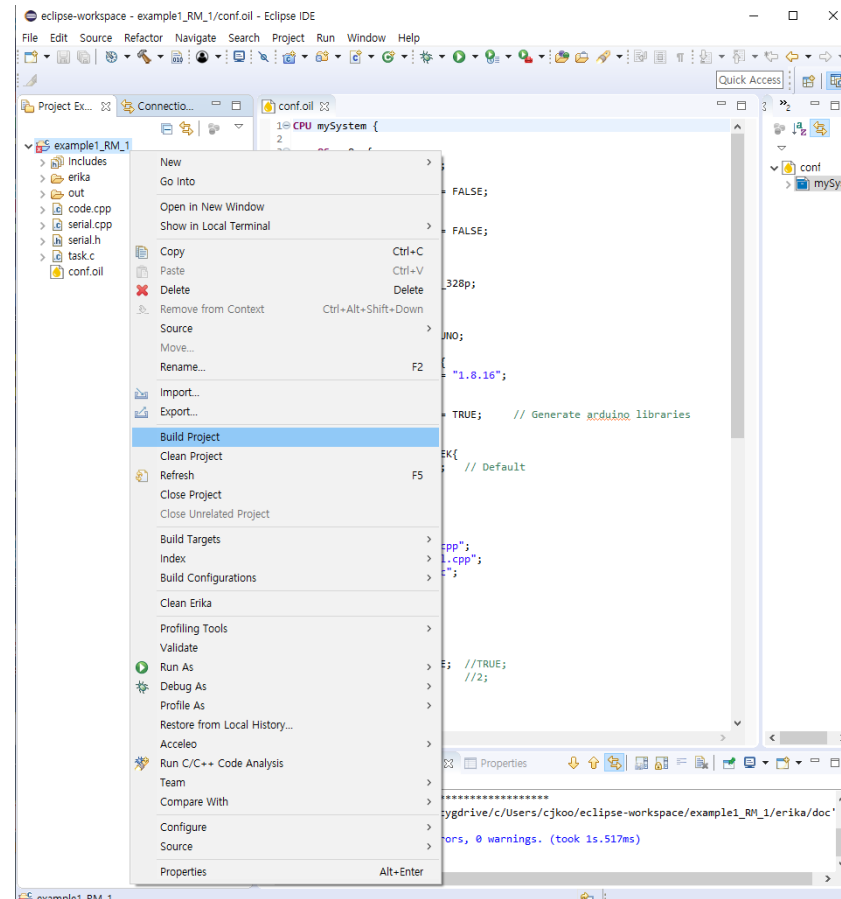


4. 디렉토리 생성 확인



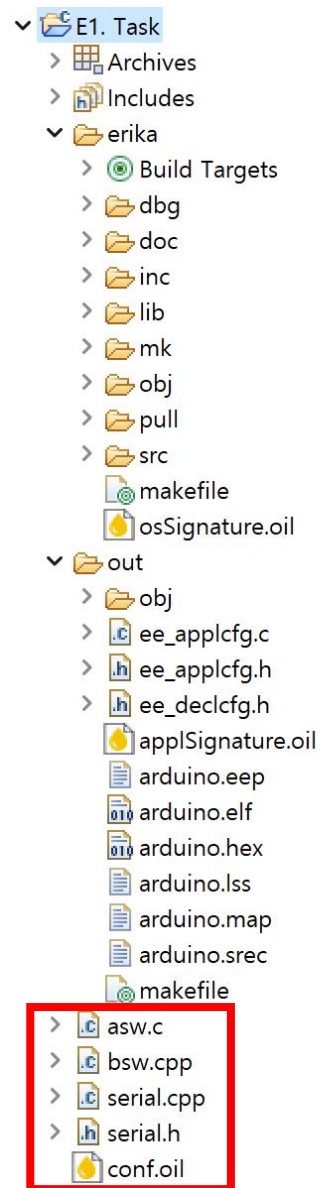
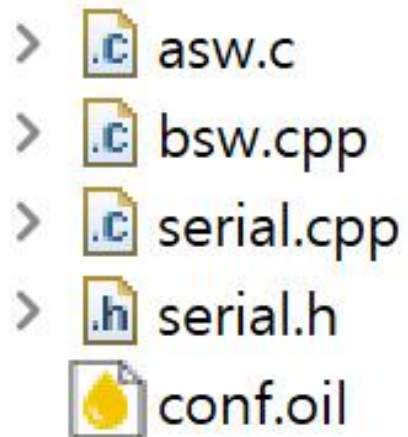
프로젝트 생성

5. 우클릭 & Build Project



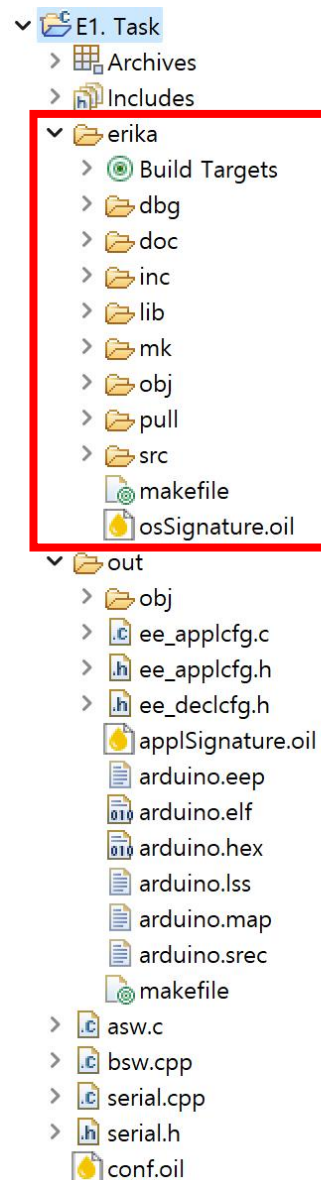
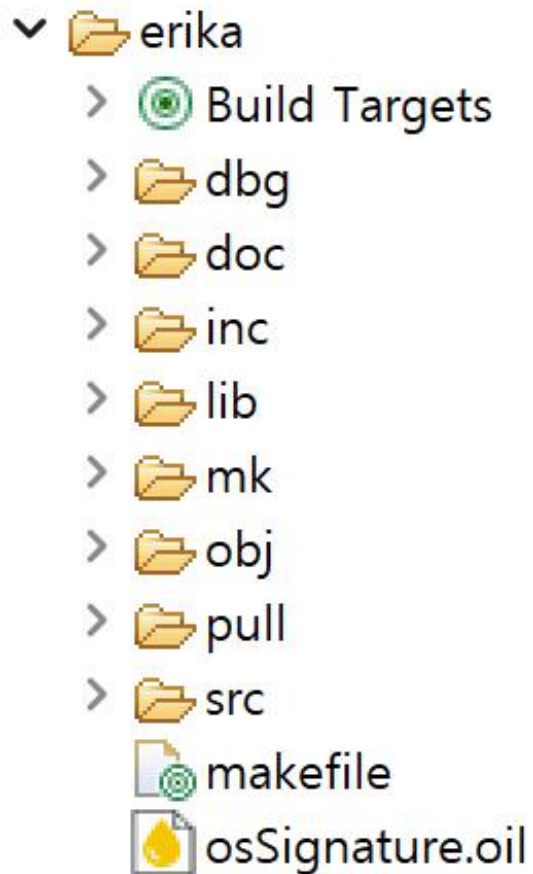
프로젝트 구조

- 사용자 코드
 - OIL 파일
 - C/C++ 파일 (TASK와 ISR)



프로젝트 구조

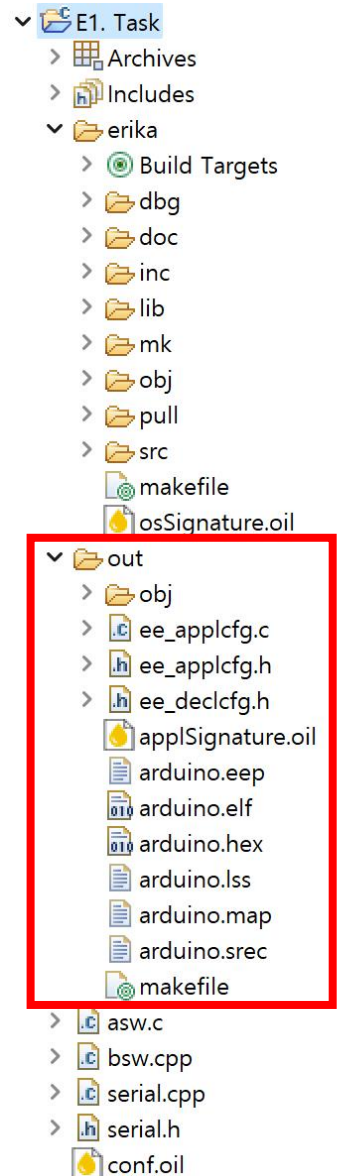
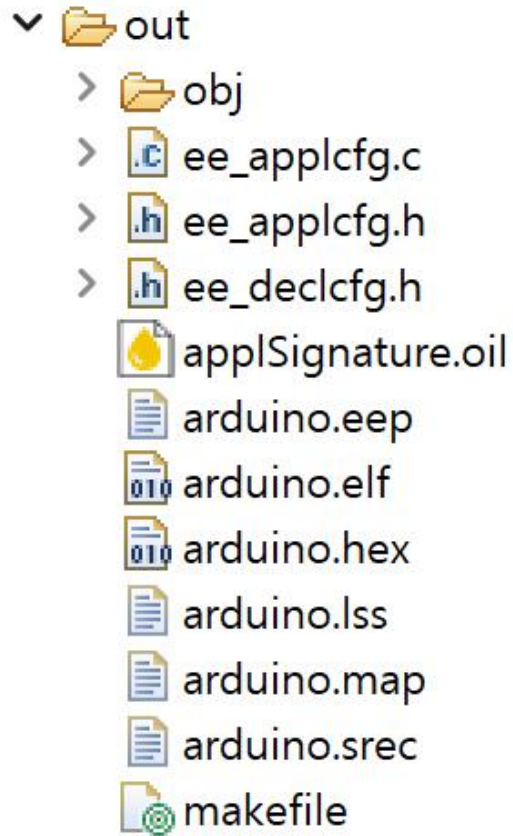
- Erika Enterprise 파일
 - 커널 소스
 - 커널 헤더파일



프로젝트 구조

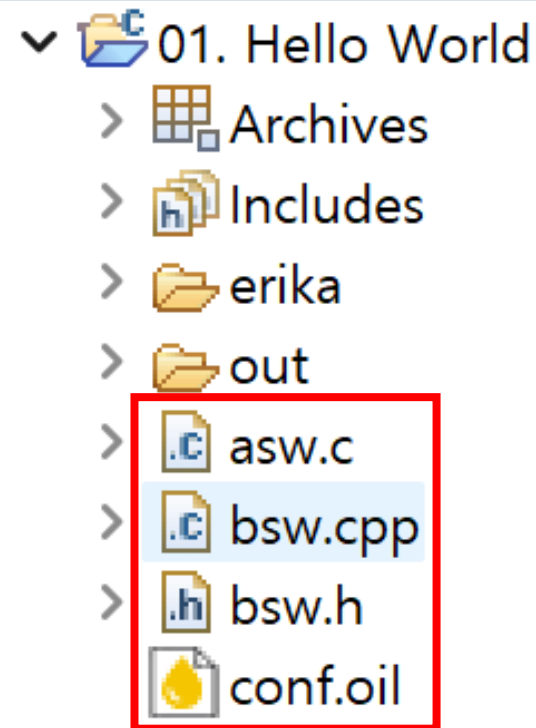
- Output 파일

- OIL에서 자동 생성된 ...cfg.c와 ...cfg.h 파일
- ELF 파일
- 다운로드를 위한 HEX 파일



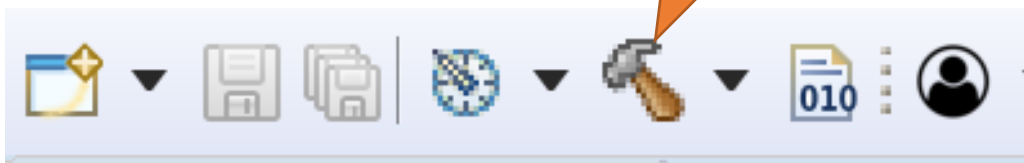
기본 소스코드 구성

- asw.c
 - 사용자 작성 C 코드 (C++도 가능)
- bsw.cpp & bsw.h
 - OS가 시작되는 main() 함수
 - mdelay() ms 단위 delay 함수
 - printfSerial() 콘솔 출력 함수
- conf.oil
 - OSEK 설정 파일



버튼

빌드

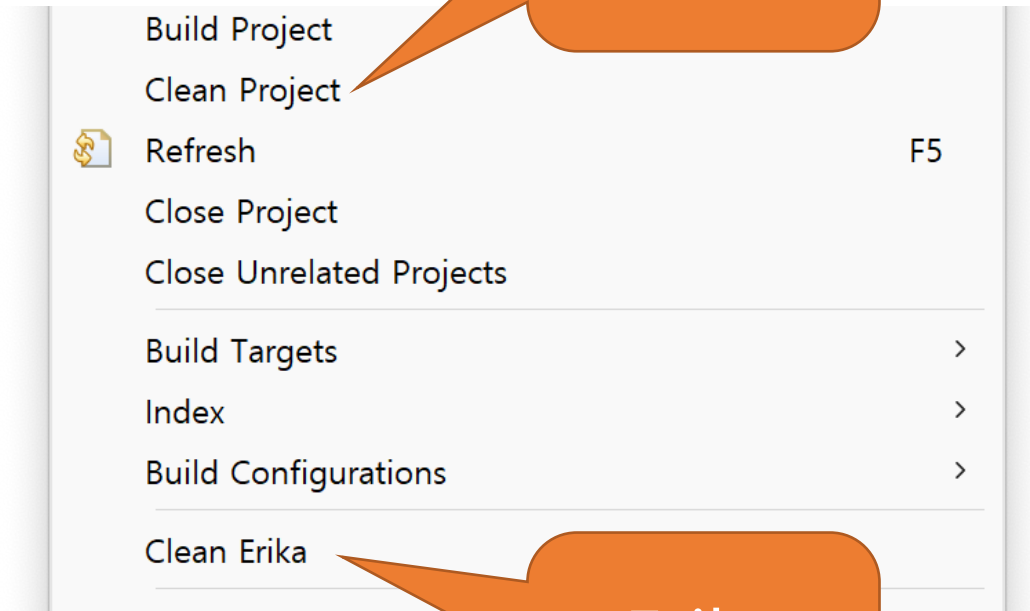


디버그



프로젝트 우클릭

클린
빌드



Erika
재설정

프로젝트 Copy & Paste



프로젝트 이름 변경

01. Hello World

- 다운로드
 - bsw.cpp
 - bsw.h
 - conf.oil
- asw.c 작성
- OIL 파일에 TASK 추가

```
#include "bsw.h"

TASK(Task1)
{
    printfSerial("Hello World\n");

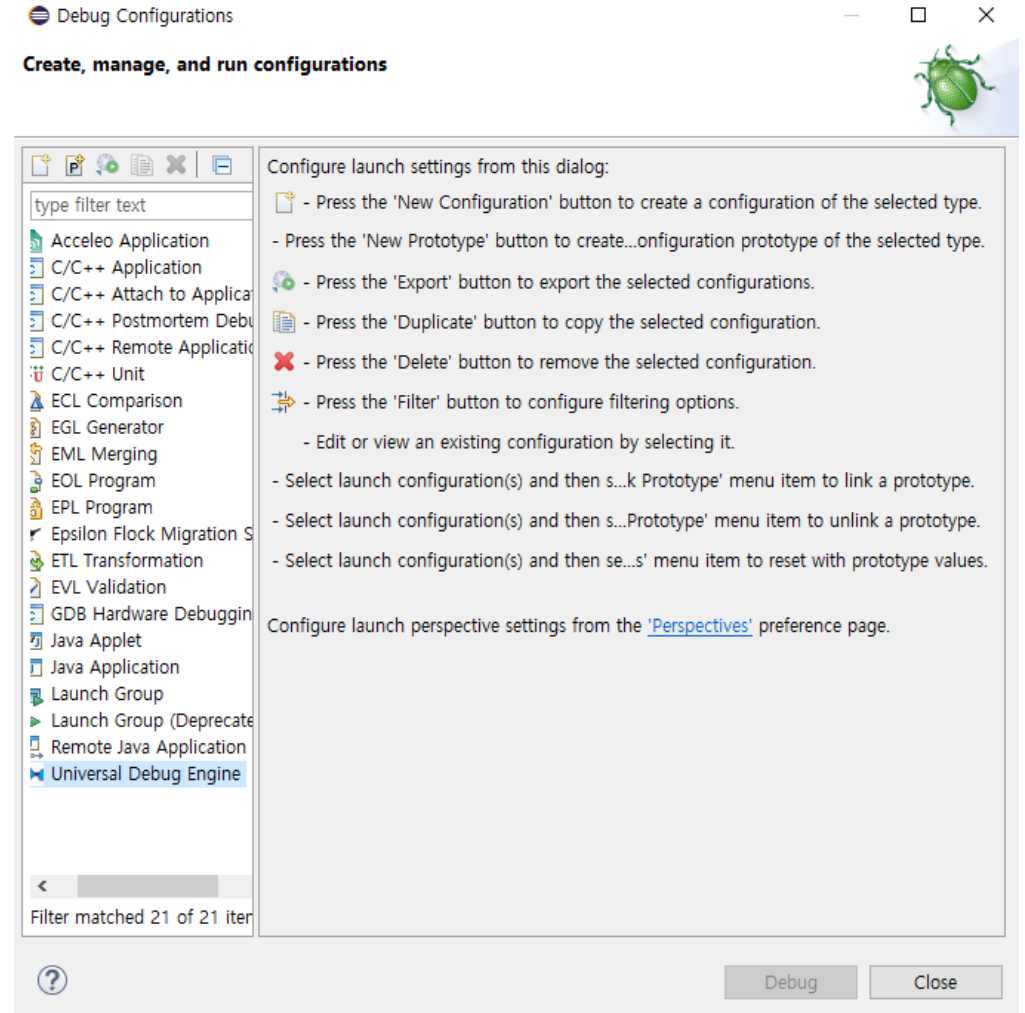
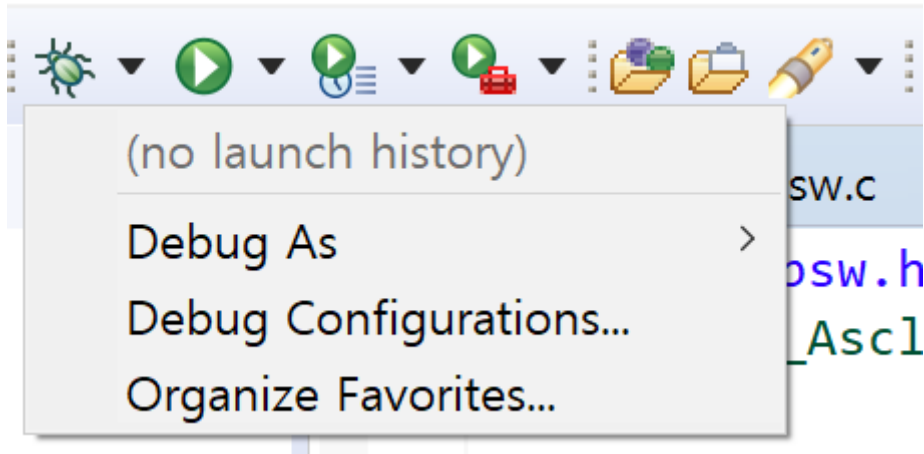
    TerminateTask();
}
```

```
TASK Task1 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL; // preemptive
    AUTOSTART = TRUE;
    ACTIVATION = 1;
};
```

자동
시작

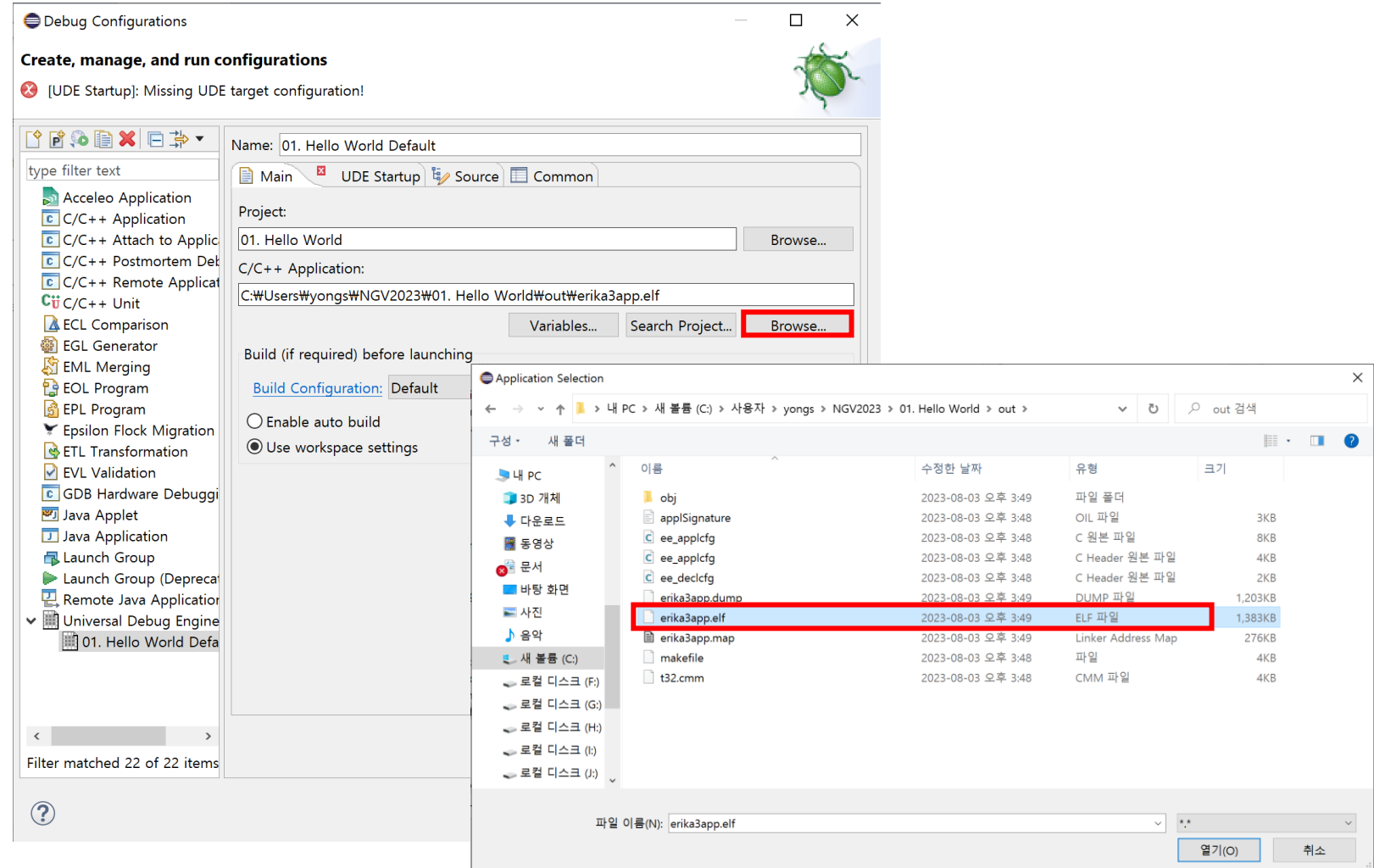
01. Hello World

- 상단 디버그 클릭 → Debug Configurations 클릭



01. Hello World

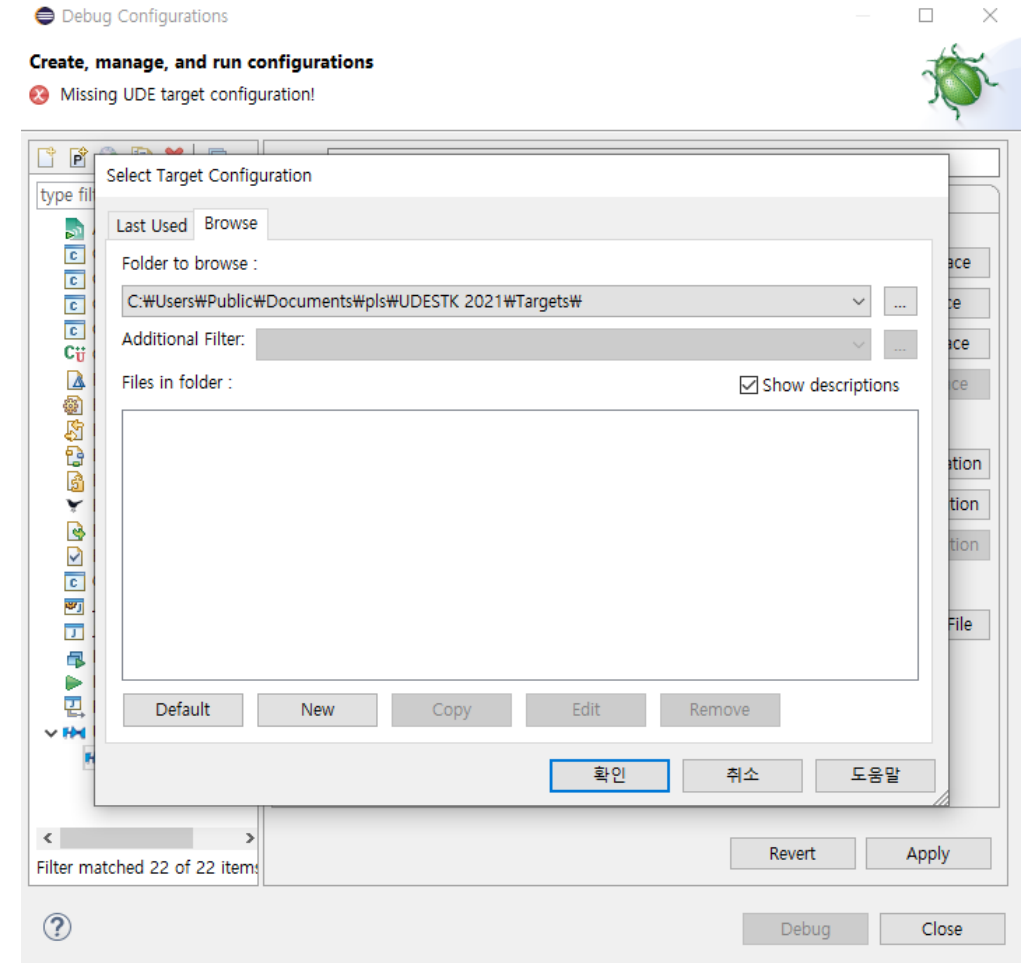
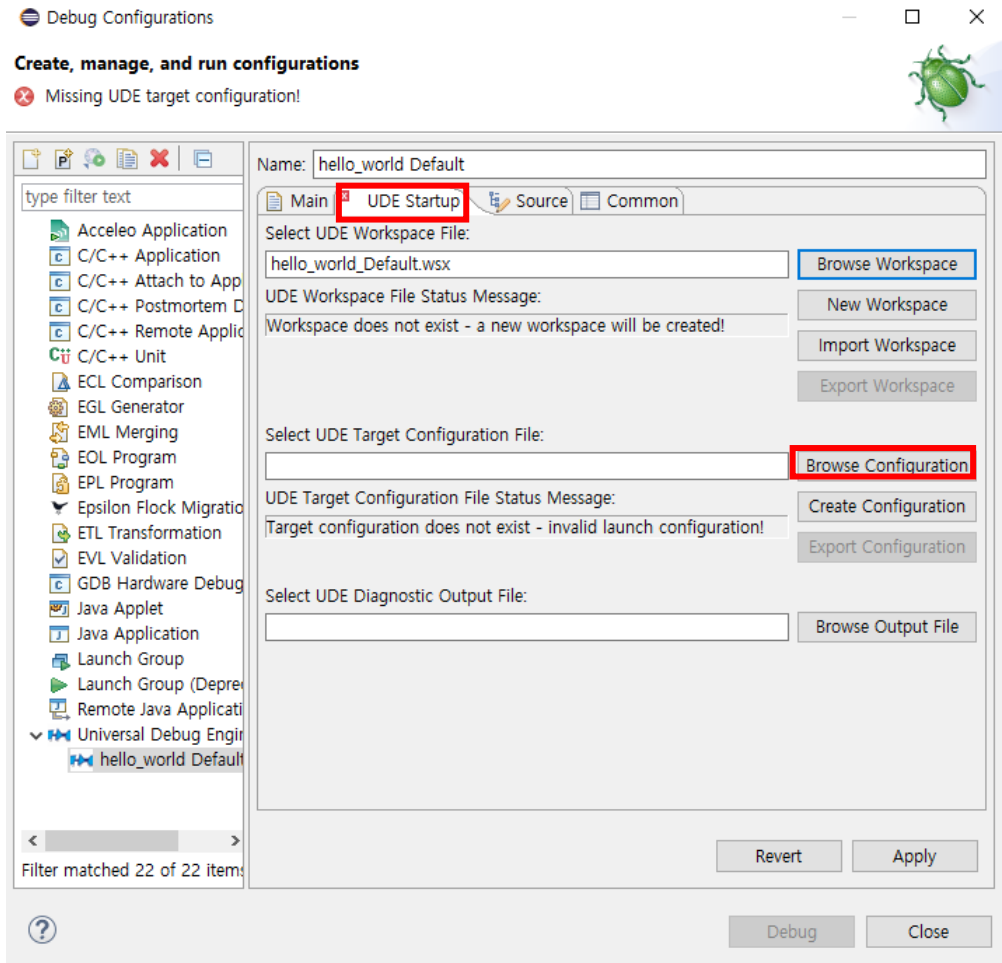
- Universal Debug Engine 더블 클릭 → 새로운 Config 파일 생성



C:\Users\{Username}\{workspace_name}\01.Hello World\out\erika3app.elf

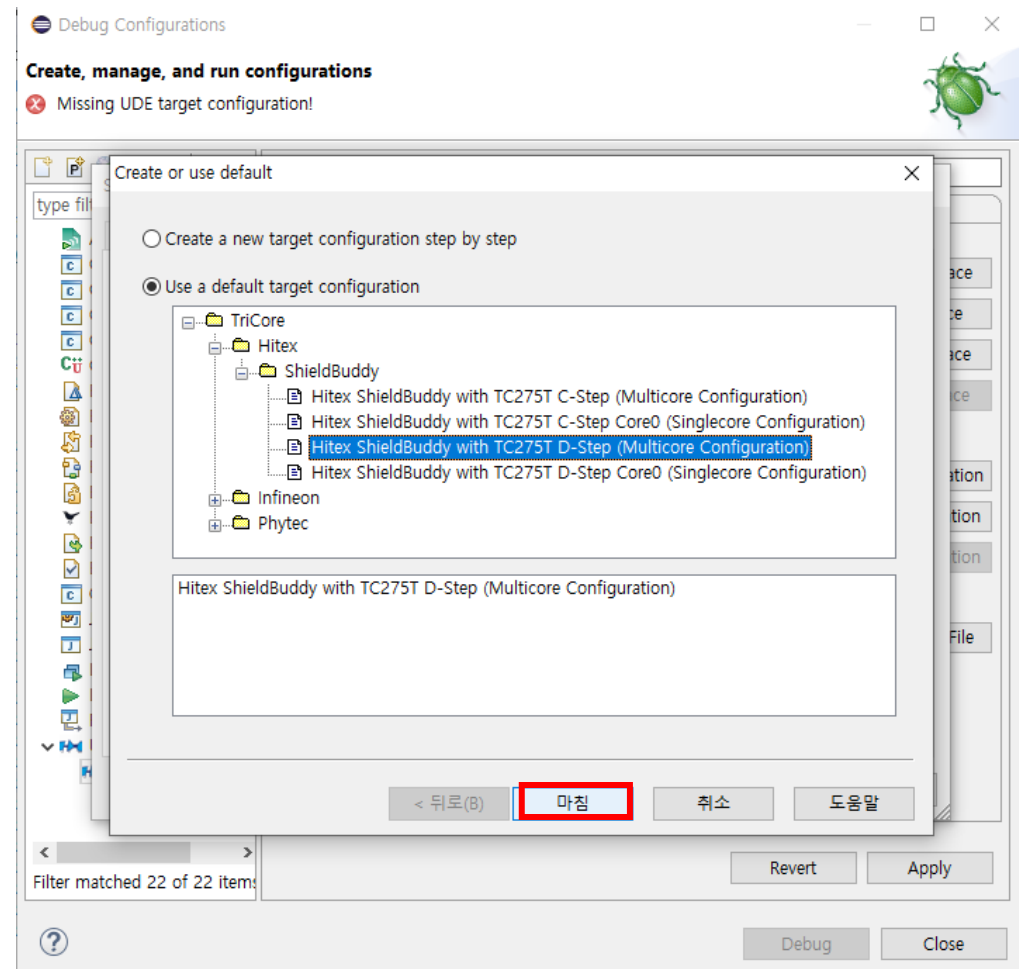
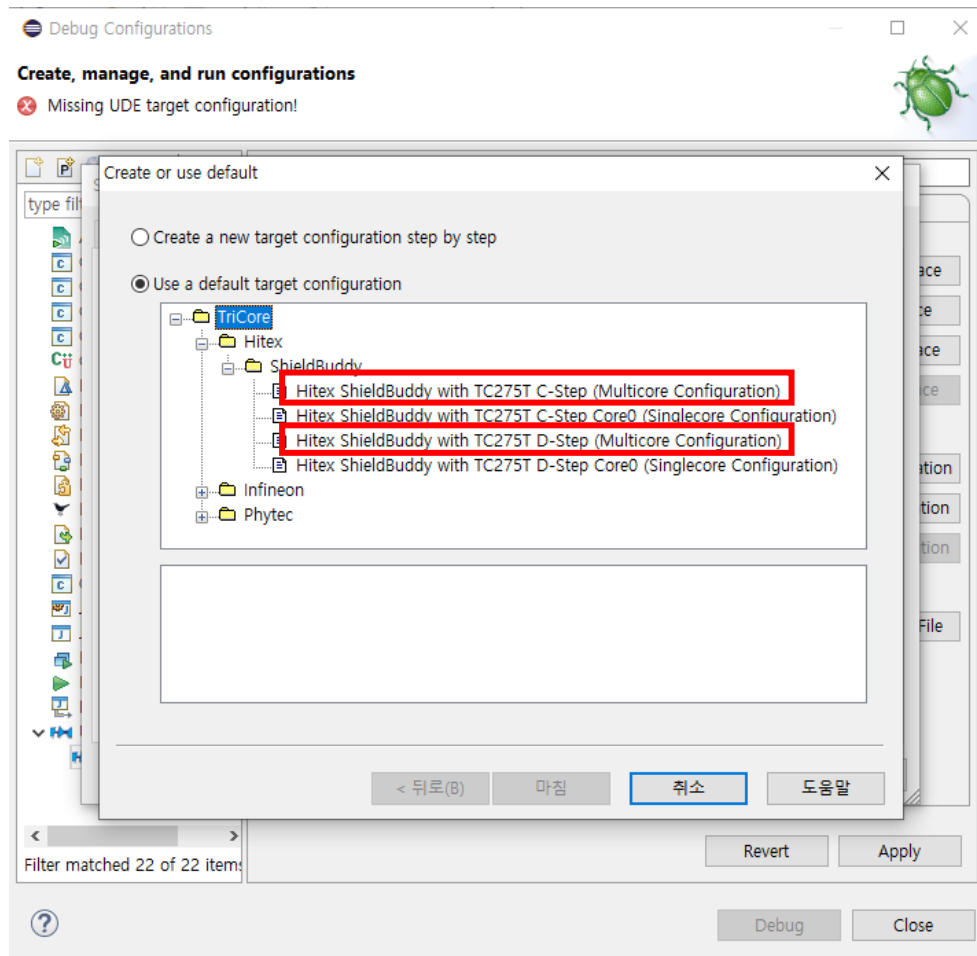
01. Hello World

- UDE Startup 클릭 → Browse Configuration 클릭



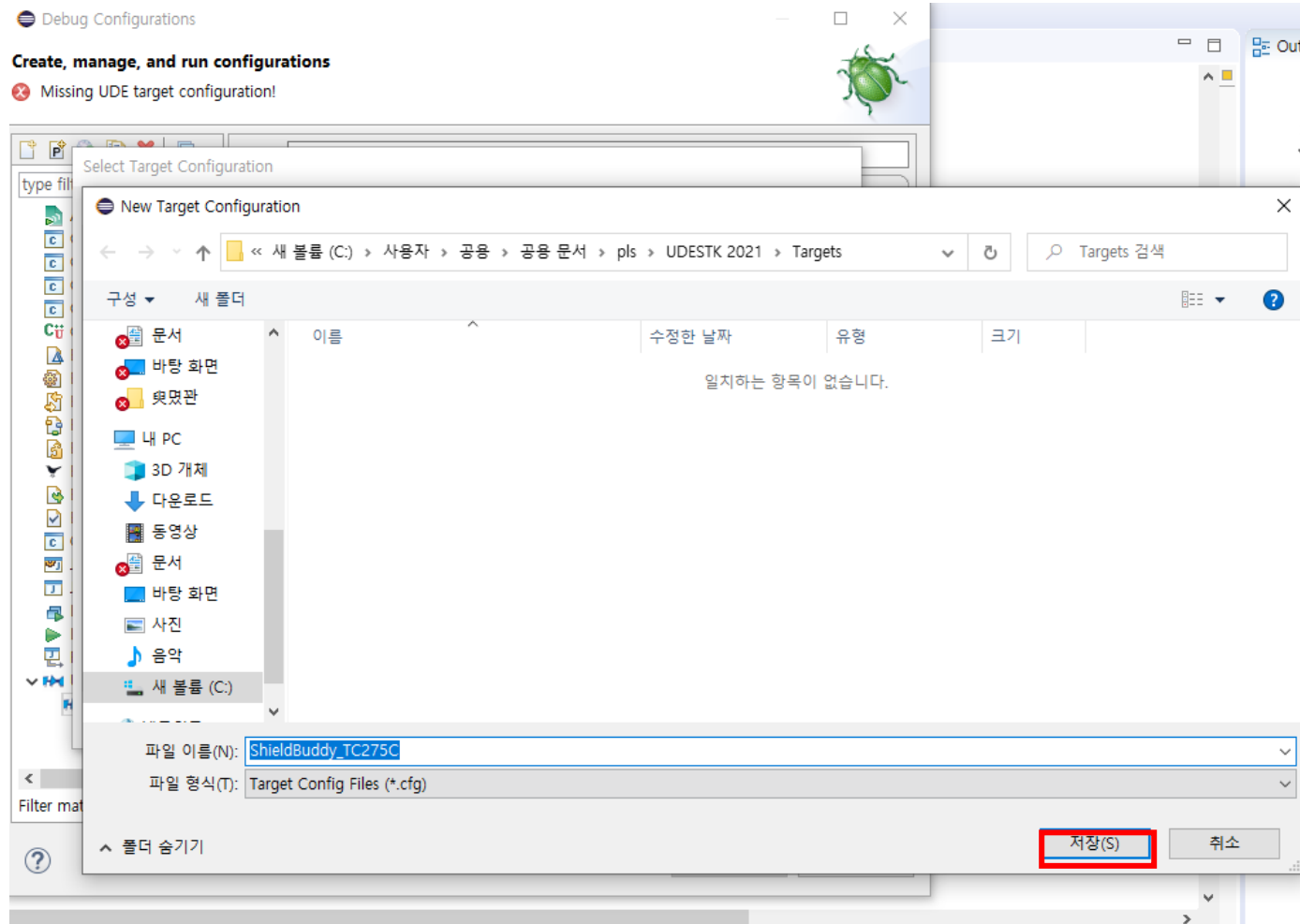
01. Hello World

- Tricore → Hitex → ShieldBuddy → Aurix에 맞는 버전(C-STEP or D-STEP) 클릭



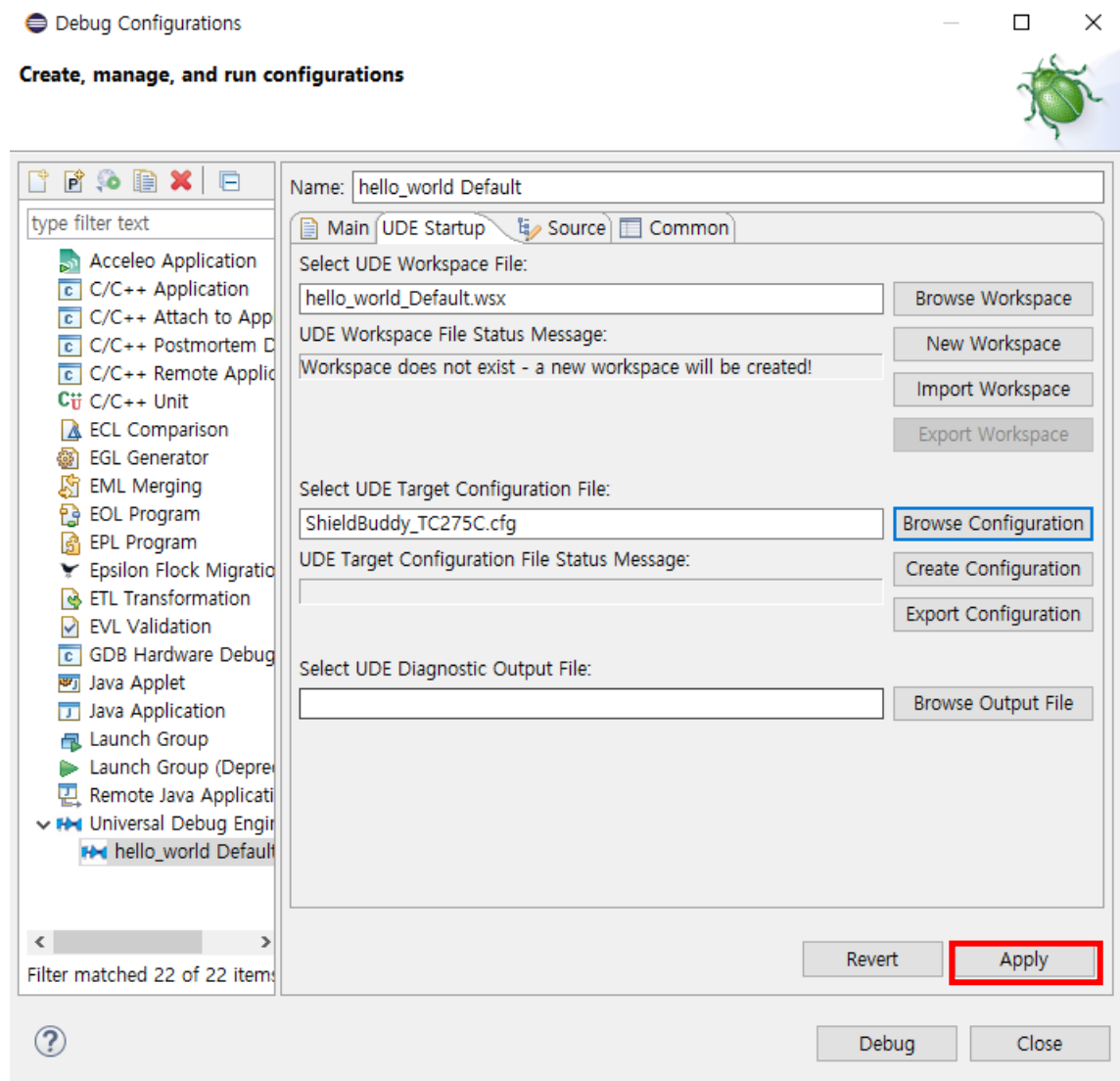
01. Hello World

- 저장 클릭



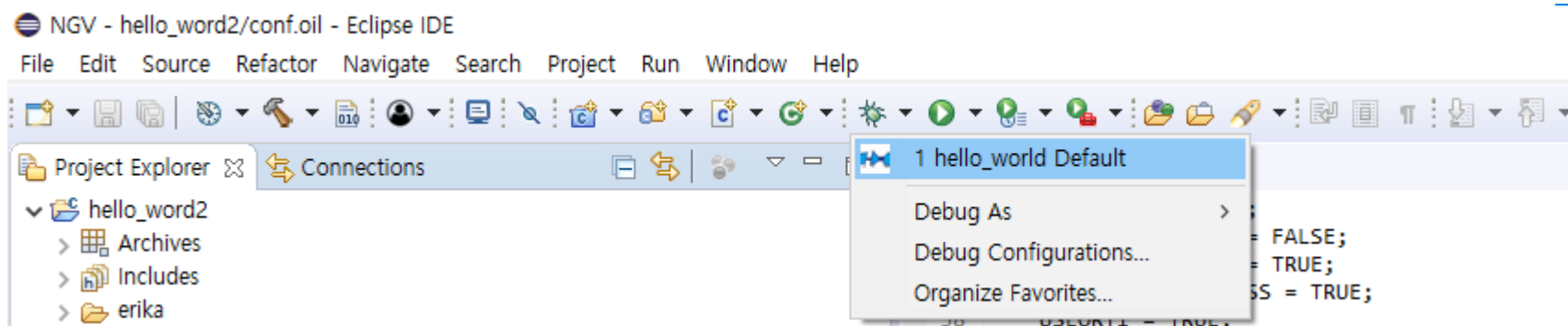
01. Hello World

- Apply 클릭



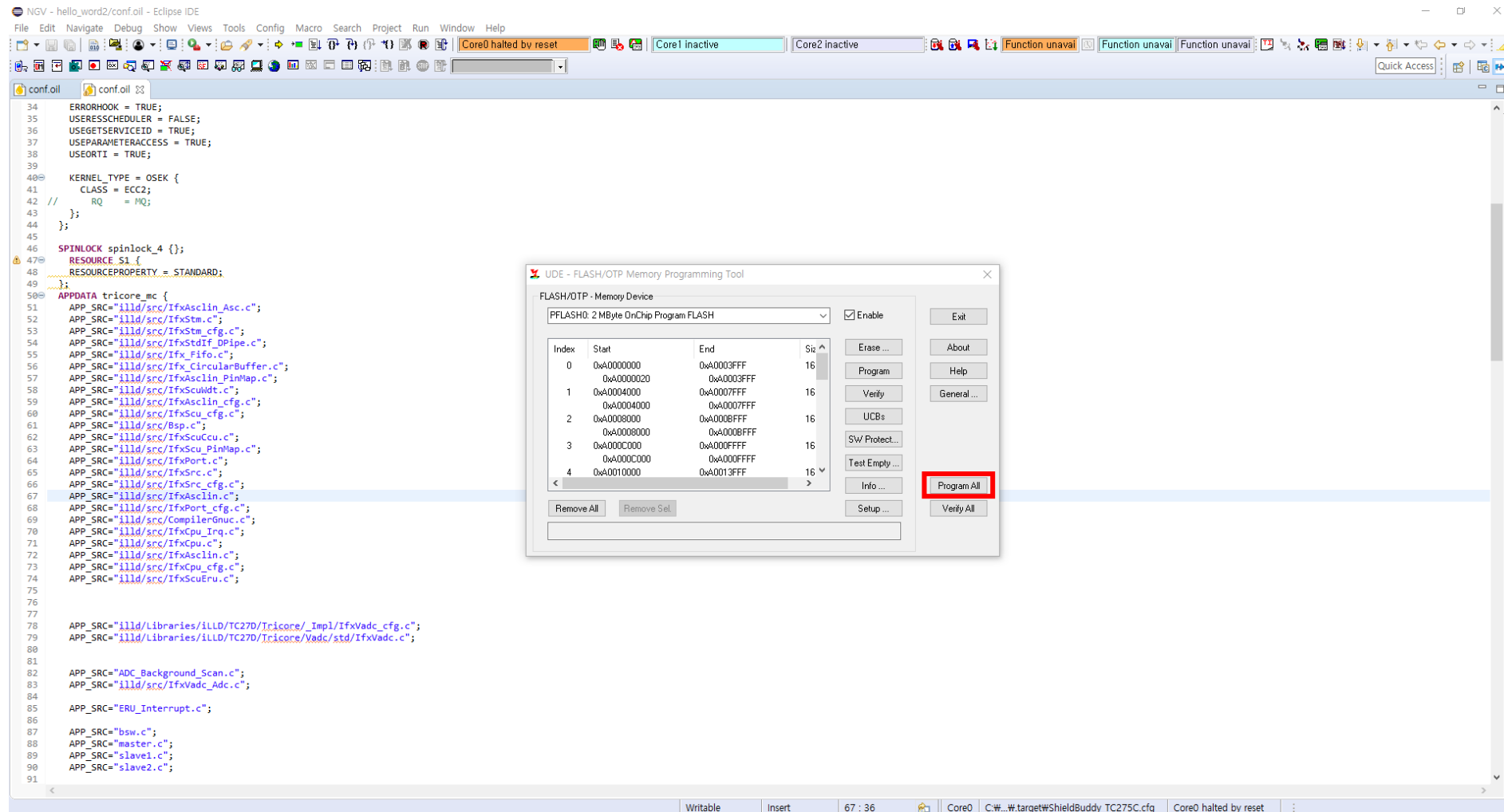
01. Hello World

- 디버그 클릭 → 방금 설정한 Debug 세팅 클릭



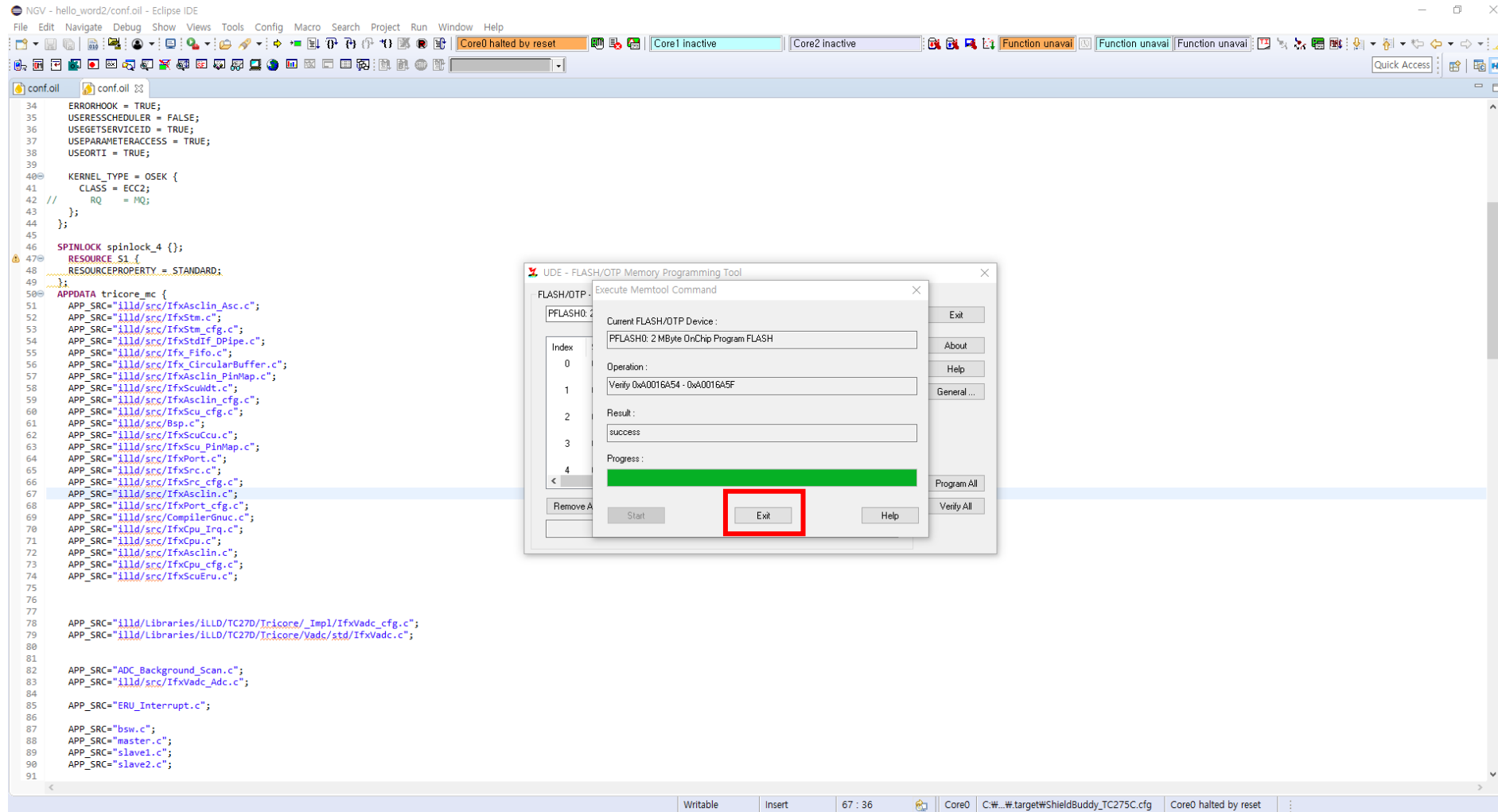
01. Hello World

- 다음 화면이 나오면 Program All 을 눌러 Flashing



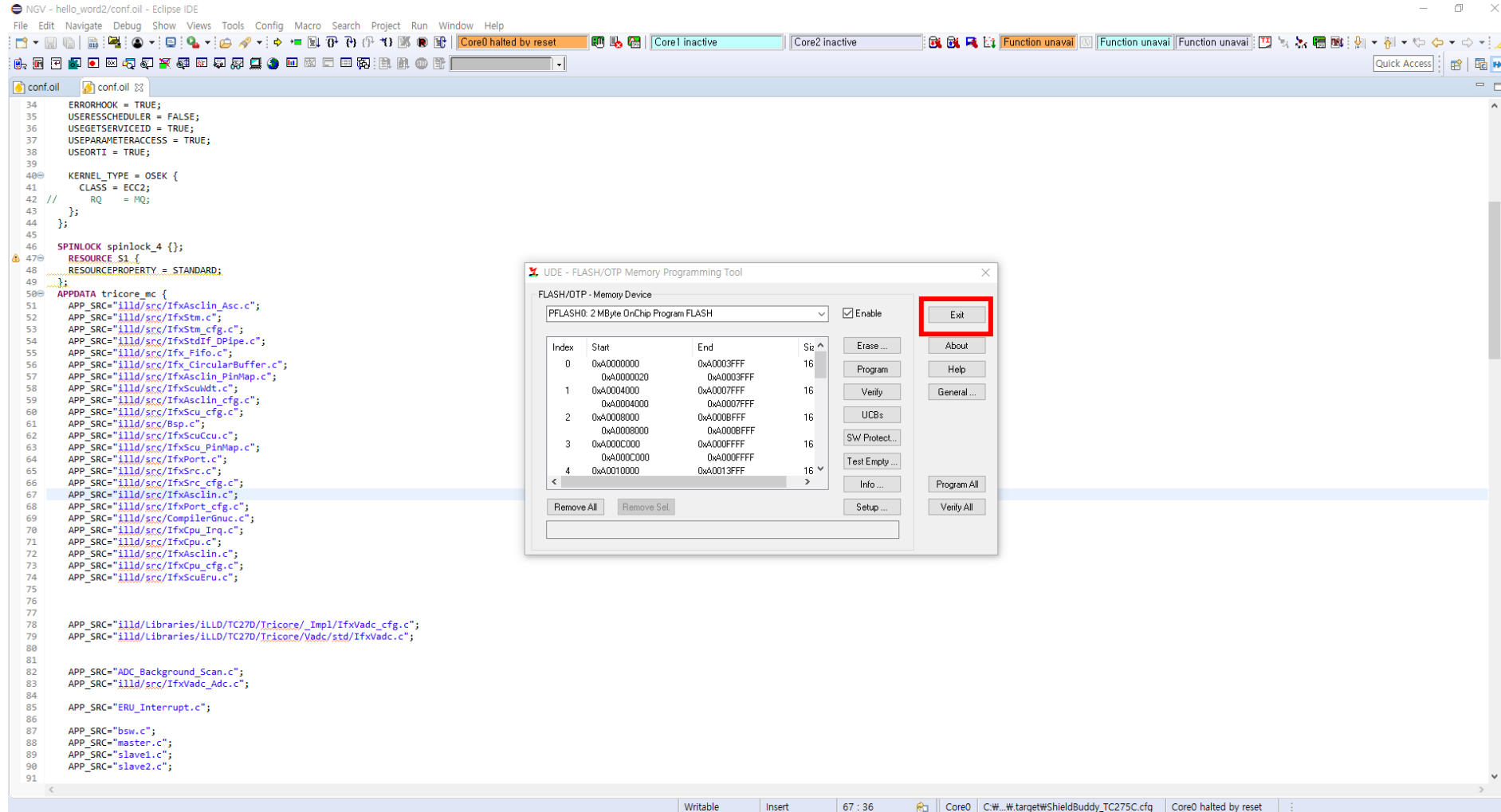
01. Hello World

- 완료 후 Exit



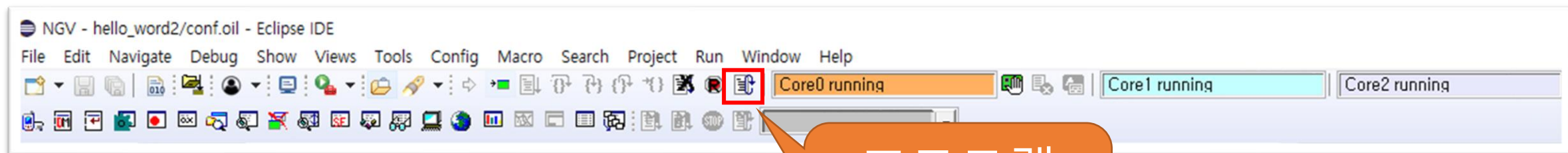
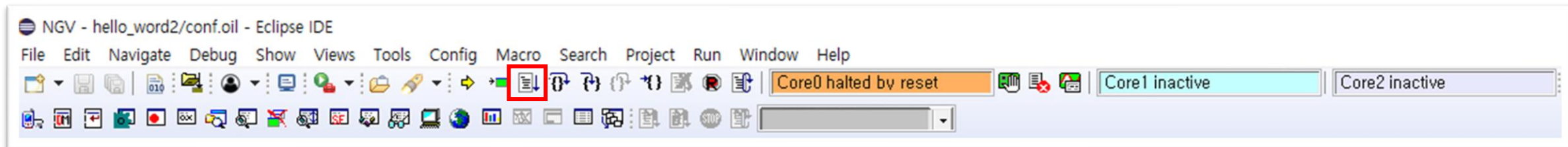
01. Hello World

- Exit



01. Hello World

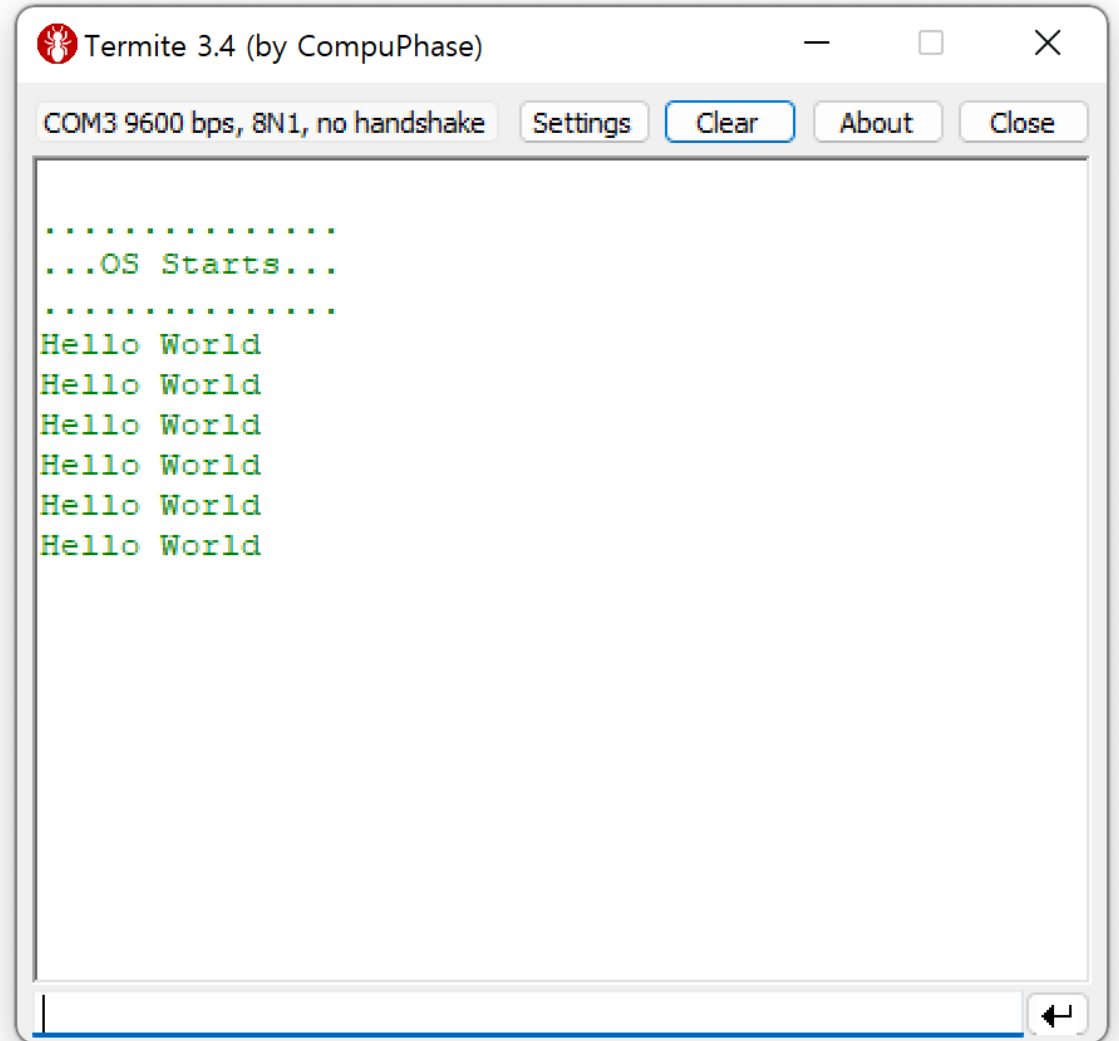
- 빨간 네모 박스 체크하여 프로그램 START



프로그램
재시작

01. Hello World

- OS 시작 후
- Hello World 반복 출력
 - 타이머 초기화 안됨



02. Timer

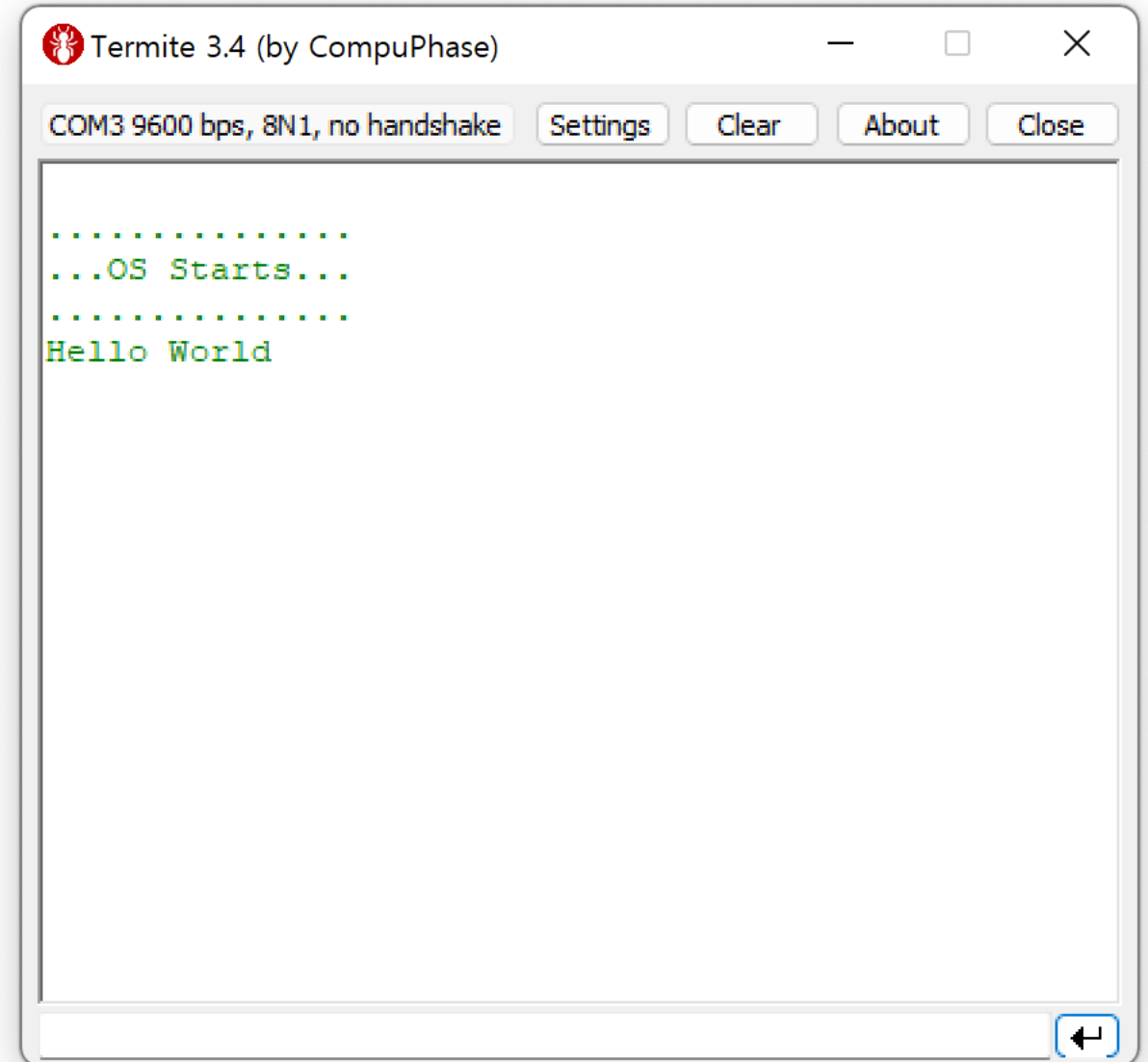
- C 파일에 ISR2로 빈 TimerISR 추가
- OIL 파일에 TimerISR 추가
 - Category 2
 - Timer 1 Compare Match A

```
ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1
000000U);
}
```

```
ISR TimerISR {
    CATEGORY = 2;
    SOURCE = "STMSR0";
    PRIORITY = 2;
};
```

02. Timer

- Hello World 한번 출력



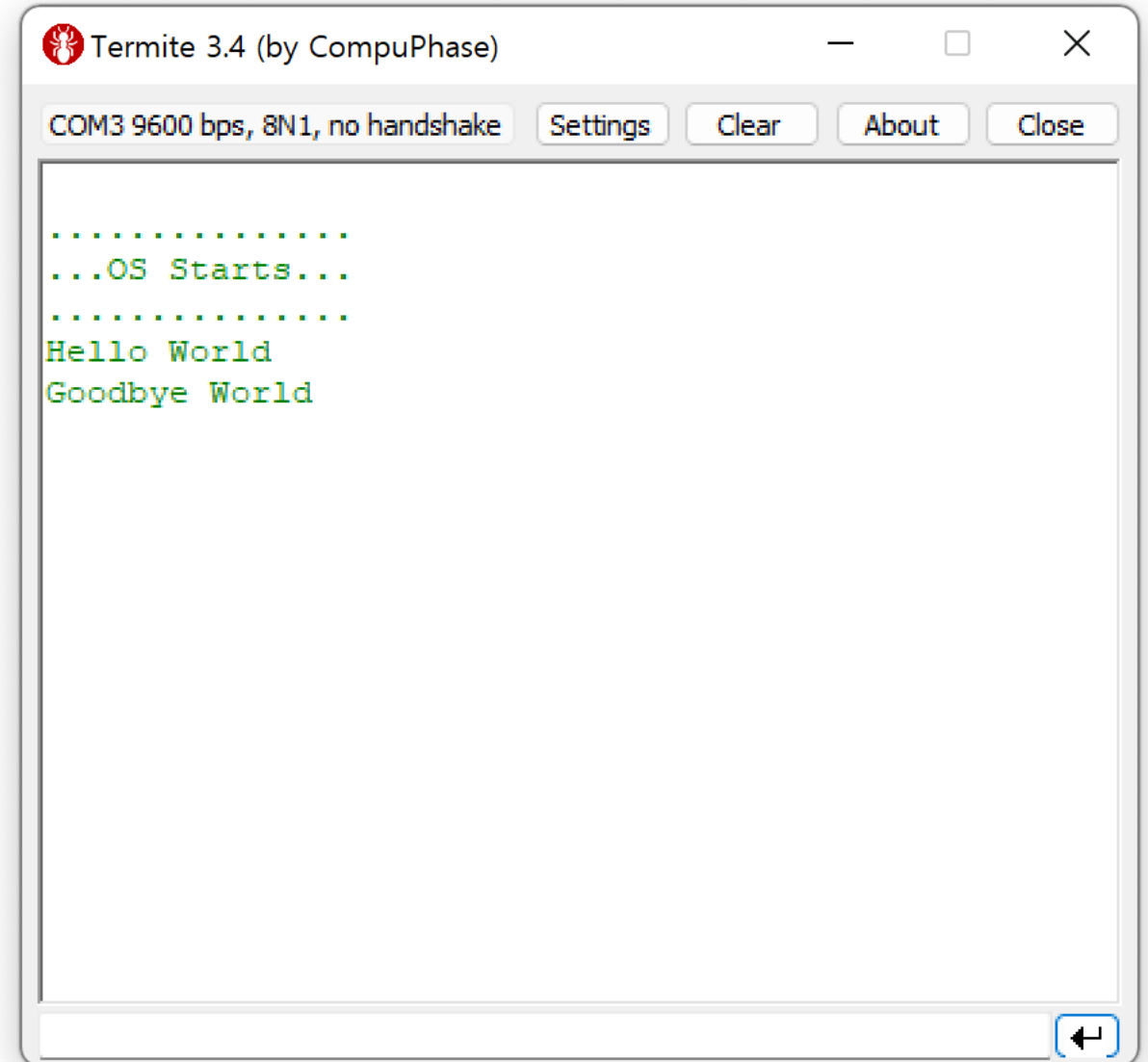
03. mdelay

- mdelay 함수 이용 3초 실행시간

```
TASK(Task1)
{
    printfSerial("Hello World\n");

    mdelay(3000);

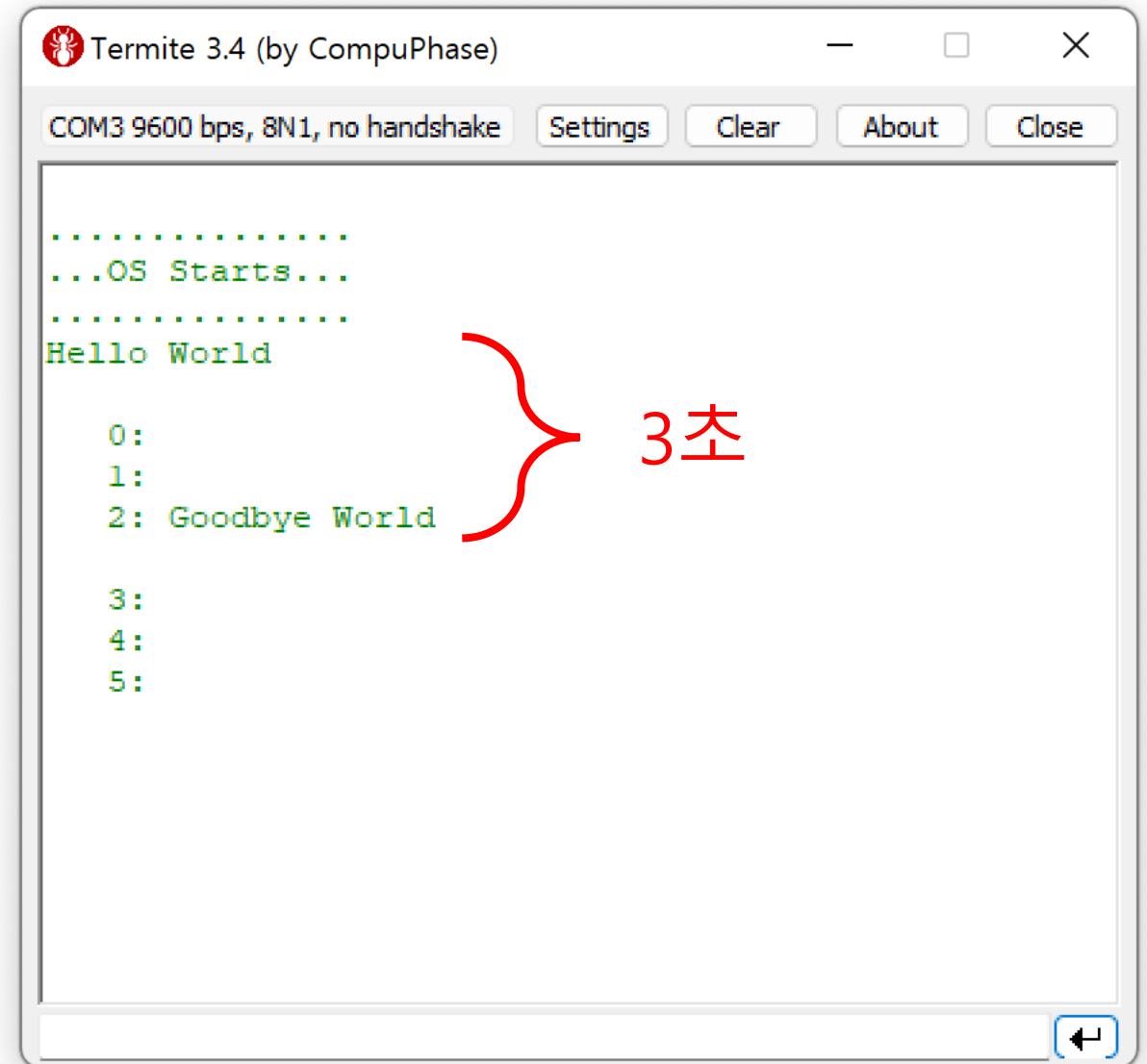
    printfSerial("Goodbye World\n");
    TerminateTask();
}
```



04. Timeline

- TimerISR 이용 초단위 시간 출력

```
ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1
000000U);
    static long c = 0;
    printfSerial("\n%4ld: ", c++);
}
```



05. Tasks

```
TASK(Task1)
{
    printfSerial("Task1 Begins...");
    mdelay(3000);
    printfSerial("Task1 Finishes...");

    TerminateTask();
}

TASK(Task2)
{
    printfSerial("Task2 Begins...");
    mdelay(3000);
    printfSerial("Task2 Finishes...");

    TerminateTask();
}
```

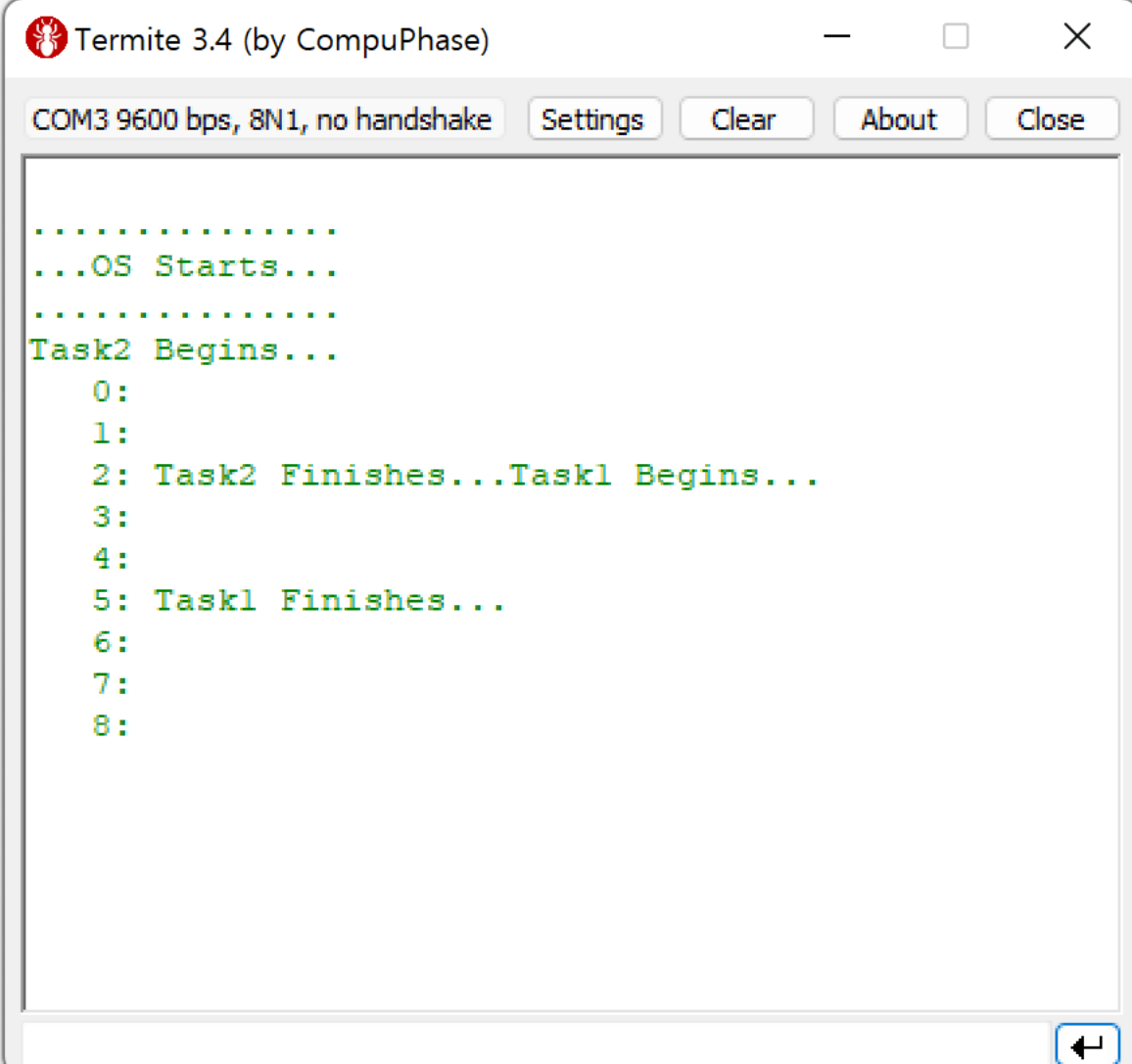
- 우선순위 2의 Task2 추가

클수록 높은
우선순위

```
TASK Task2 {
    PRIORITY = 2;
    STACK = SHARED;
    SCHEDULE = FULL; // preemptive
    AUTOSTART = TRUE;
    ACTIVATION = 1;
};
```

05. Tasks

- Task2가 먼저 시작
- Task2 종료 후 Task1 시작
- 우선순위를 바꾼다면?



The screenshot shows a terminal window titled "Termite 3.4 (by CompuPhase)". The window has a menu bar with "COM3 9600 bps, 8N1, no handshake", "Settings", "Clear", "About", and "Close". The terminal output is as follows:

```
.....  
...OS Starts...  
.....  
Task2 Begins...  
  0:  
  1:  
  2: Task2 Finishes...Task1 Begins...  
  3:  
  4:  
  5: Task1 Finishes...  
  6:  
  7:  
  8:
```

At the bottom right of the terminal window, there is a button with a left arrow and a return key symbol.

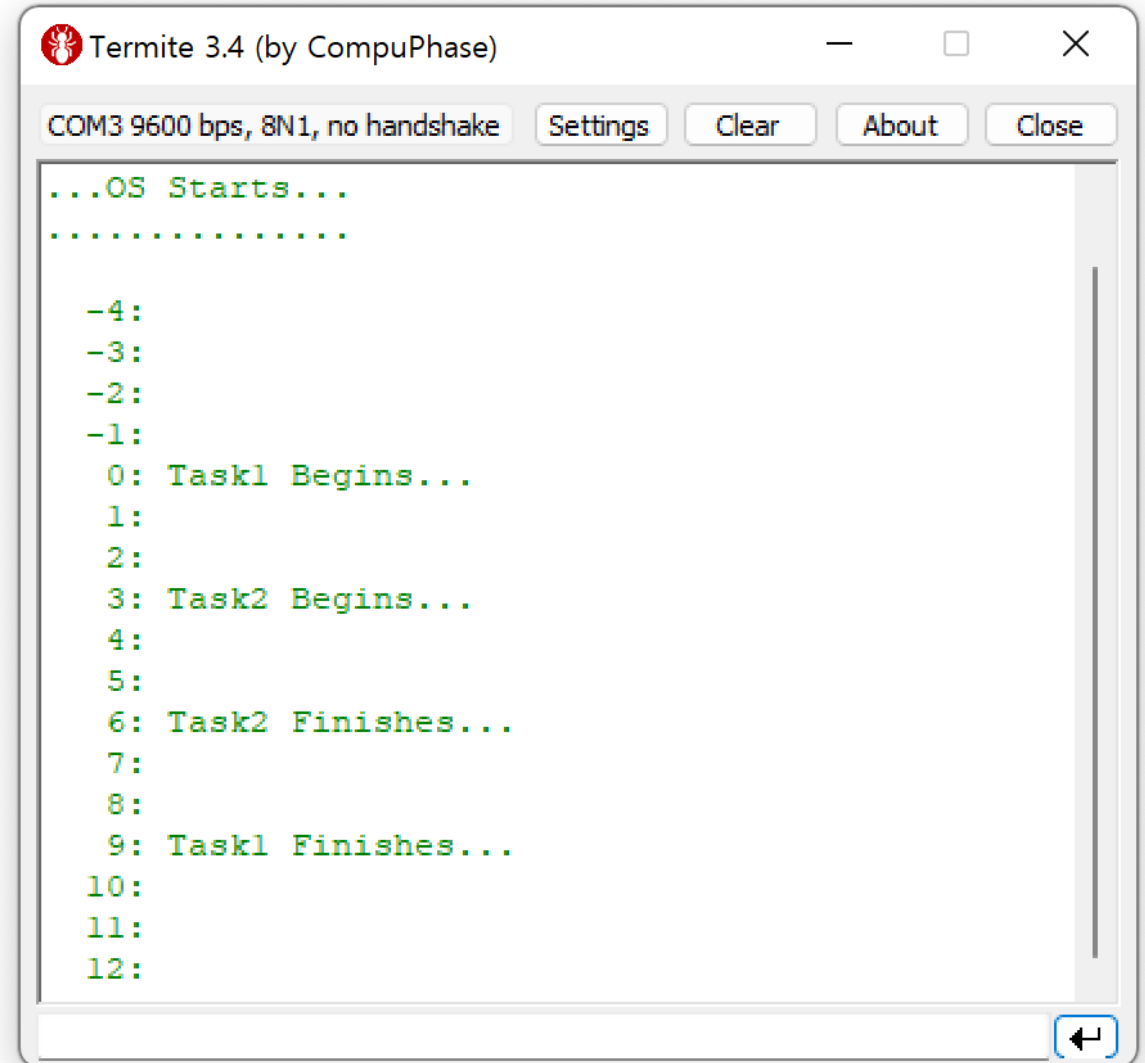
06. Task Activation

```
ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000
000U);
    static long c = -4;
    if (c == 0)
        ActivateTask(Task1);
    printfSerial("\n%4ld: ", c++);
}
TASK(Task1)
{
    printfSerial("Task1 Begins...");
    mdelay(3000);
    ActivateTask(Task2);
    mdelay(3000);
    printfSerial("Task1 Finishes...");
    TerminateTask();
}
```

```
TASK(Task2)
{
    printfSerial("Task2 Begins...");
    mdelay(3000);
    printfSerial("Task2 Finishes...");
    TerminateTask();
}
TASK Task1 {
    ...
    AUTOSTART = FALSE;
    ...
};
TASK Task2 {
    ...
    AUTOSTART = FALSE;
    ...
};
```

06. Task Activation

- Timeline -4부터 카운트다운
- Task2의 Task1 선점 확인
- ActivateTask 위치 바꾸면?
- 우선순위가 바뀌면?
- ChainTask 활용
- Task3까지 만들어서 연쇄 실행



The screenshot shows a terminal window titled "Termite 3.4 (by CompuPhase)". The window has a menu bar with "Settings", "Clear", "About", and "Close". The terminal output shows a sequence of events:

```
...OS Starts...  
.....  
-4:  
-3:  
-2:  
-1:  
0: Task1 Begins...  
1:  
2:  
3: Task2 Begins...  
4:  
5:  
6: Task2 Finishes...  
7:  
8:  
9: Task1 Finishes...  
10:  
11:  
12:
```

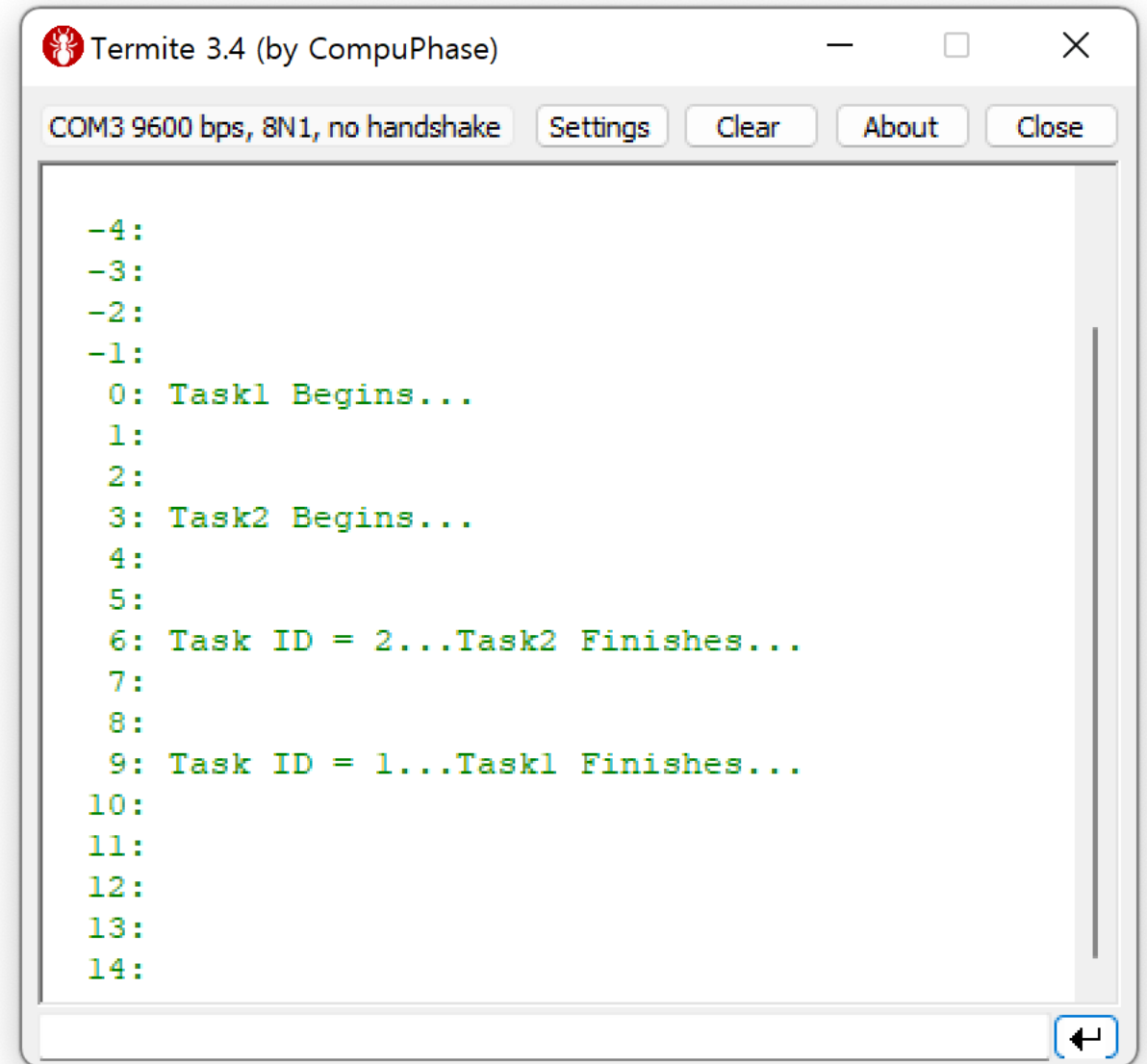
07. GetTaskID

```
TASK(Task1)
{
    TaskType id;
    printfSerial("Task1 Begins...");
    mdelay(3000);
    ActivateTask(Task2);
    mdelay(3000);
    GetTaskID(&id);
    printfSerial("Task ID = %d...", id);
    printfSerial("Task1 Finishes...");
    TerminateTask();
}
```

```
TASK(Task2)
{
    TaskType id;
    printfSerial("Task2 Begins...");
    mdelay(3000);
    GetTaskID(&id);
    printfSerial("Task ID = %d...", id);
    printfSerial("Task2 Finishes...");
    TerminateTask();
}
```

07. GetTaskID

- 자연수 Task ID 확인



The screenshot shows a terminal window titled "Termite 3.4 (by CompuPhase)". The window has a menu bar with "COM3 9600 bps, 8N1, no handshake", "Settings", "Clear", "About", and "Close". The main text area displays a log of task execution with line numbers from -4 to 14. The log shows two tasks, Task1 and Task2, starting and finishing, with their respective Task IDs (1 and 2) being reported.

```
-4:
-3:
-2:
-1:
 0: Task1 Begins...
 1:
 2:
 3: Task2 Begins...
 4:
 5:
 6: Task ID = 2...Task2 Finishes...
 7:
 8:
 9: Task ID = 1...Task1 Finishes...
10:
11:
12:
13:
14:
```

08. GetTaskState

```
TASK(TaskM)
{
    printState(Task1);
    printState(Task2);

    TerminateTask();
}
```

```
TASK TaskM {
    PRIORITY = 3;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
};
```

```
void printState(TaskType id) {
    TaskStateType state;

    if (GetTaskState(id, &state) == E_OK) {
        switch (state) {
            case SUSPENDED:
                printfSerial("%d: suspended...", id);
                break;
            case READY:
                printfSerial("%d: ready...", id);
                break;
            case WAITING:
                printfSerial("%d: waiting...", id);
                break;
            case RUNNING:
                printfSerial("%d: running...", id);
                break;
        }
    }
}
```

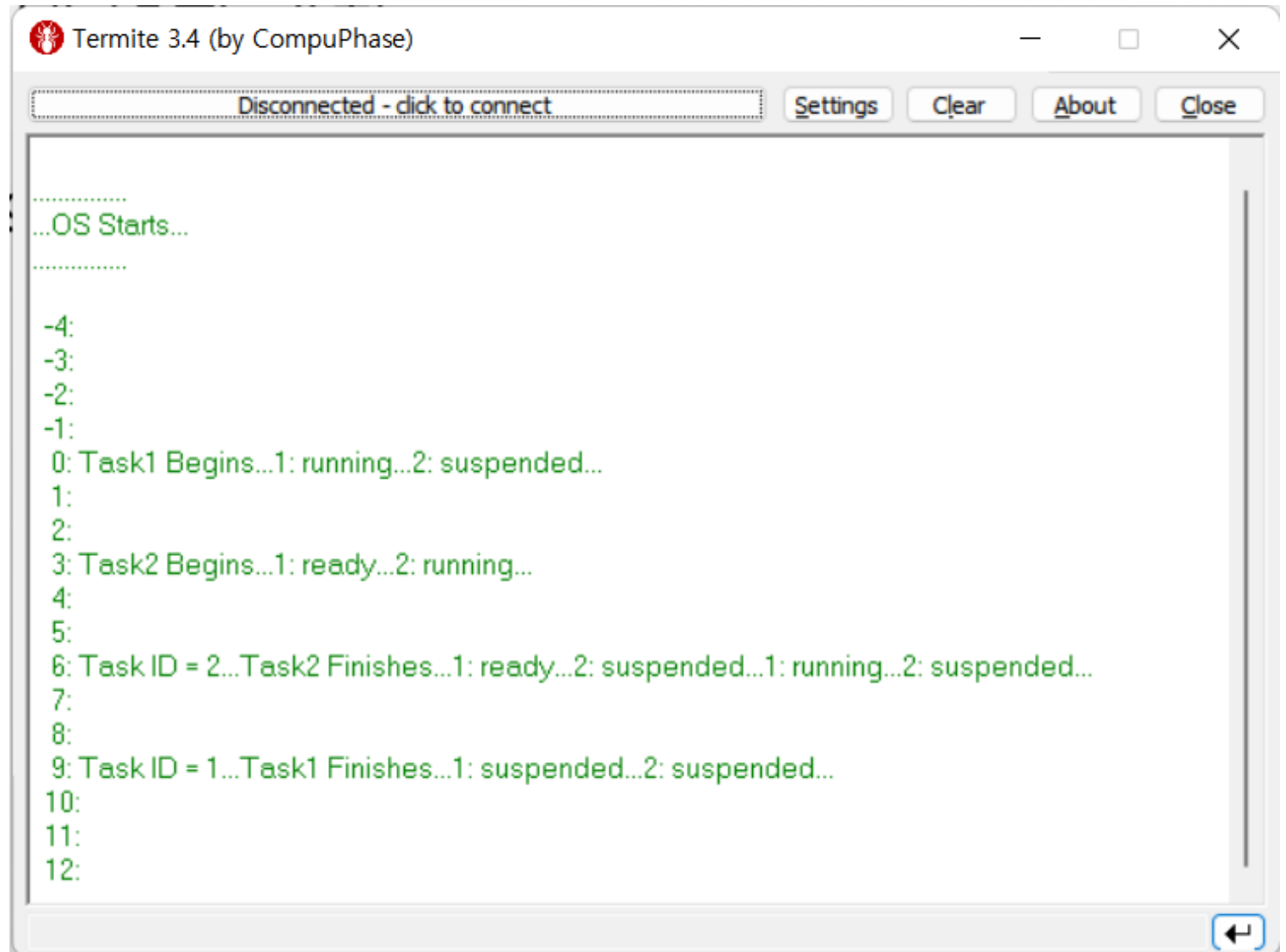
08. GetTaskState

```
TASK(Task1)
{
    TaskType id;
    printfSerial("Task1 Begins...");
    PrintState(Task1);
    PrintState(Task2);
    mdelay(3000);
    ActivateTask(Task2);
    PrintState(Task1);
    PrintState(Task2);
    mdelay(3000);
    GetTaskID(&id);
    printfSerial("Task ID = %d...", id);
    printfSerial("Task1 Finishes...");
    ChainTask(TaskM);
}
```

```
TASK(Task2)
{
    TaskType id;
    printfSerial("Task2 Begins...");
    PrintState(Task1);
    PrintState(Task2);
    mdelay(3000);
    GetTaskID(&id);
    printfSerial("Task ID = %d...", id);
    printfSerial("Task2 Finishes...");
    ChainTask(TaskM);
}
```

08. GetTaskState

- Task 상태 변화 관찰
- 우선순위, Activation 패턴 변화의 영향은?



The screenshot shows the Termite 3.4 application window. The title bar reads "Termite 3.4 (by CompuPhase)". Below the title bar is a status bar that says "Disconnected - click to connect". To the right of the status bar are four buttons: "Settings", "Clear", "About", and "Close". The main area of the window is a text log displaying the following text:

```
.....  
...OS Starts...  
.....  
-4:  
-3:  
-2:  
-1:  
0: Task1 Begins...1: running...2: suspended...  
1:  
2:  
3: Task2 Begins...1: ready...2: running...  
4:  
5:  
6: Task ID = 2...Task2 Finishes...1: ready...2: suspended...1: running...2: suspended...  
7:  
8:  
9: Task ID = 1...Task1 Finishes...1: suspended...2: suspended...  
10:  
11:  
12:
```

At the bottom right of the window, there is a blue button with a left-pointing arrow.

Questions

