



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
- 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

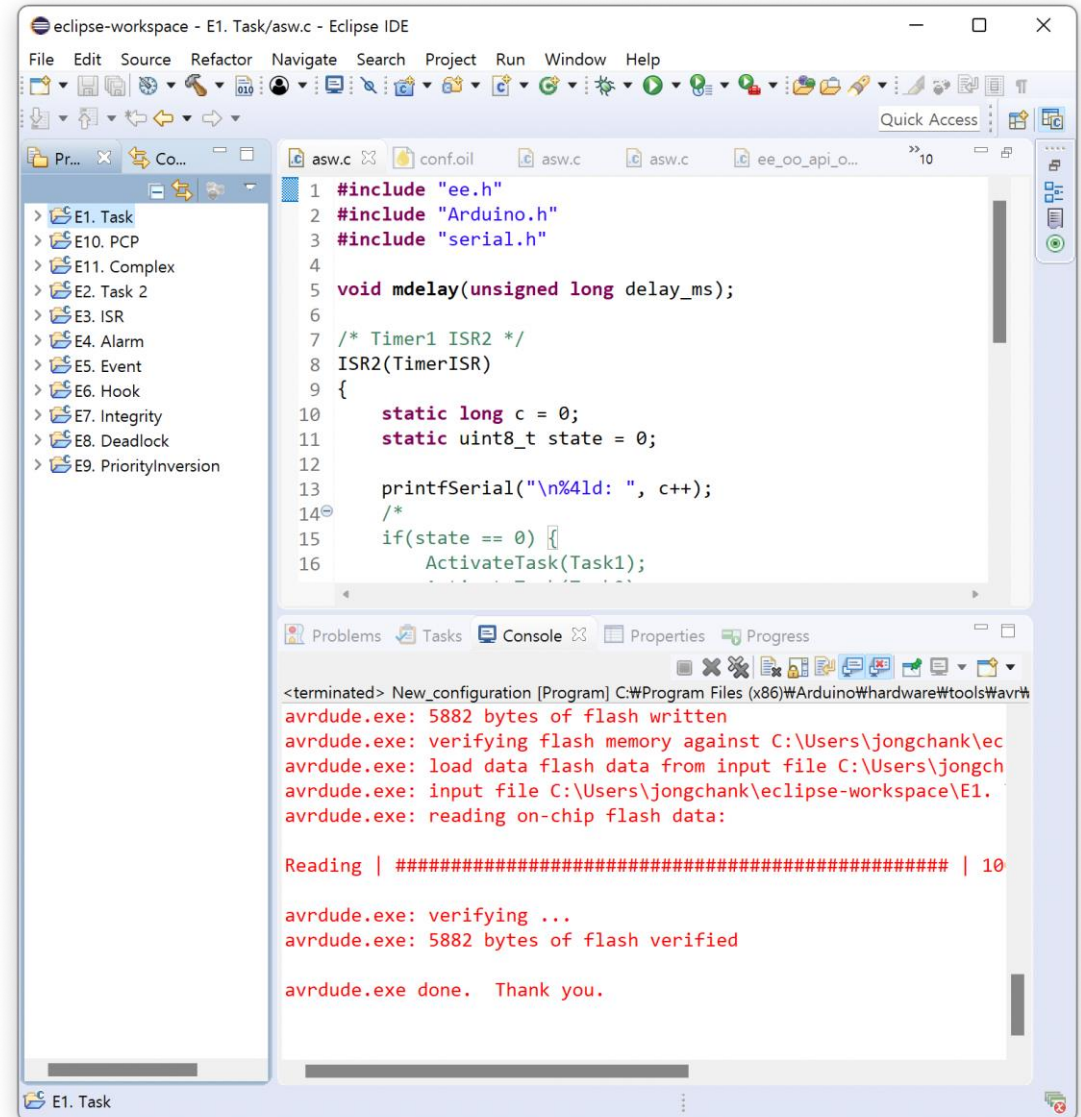
View license

155 stars

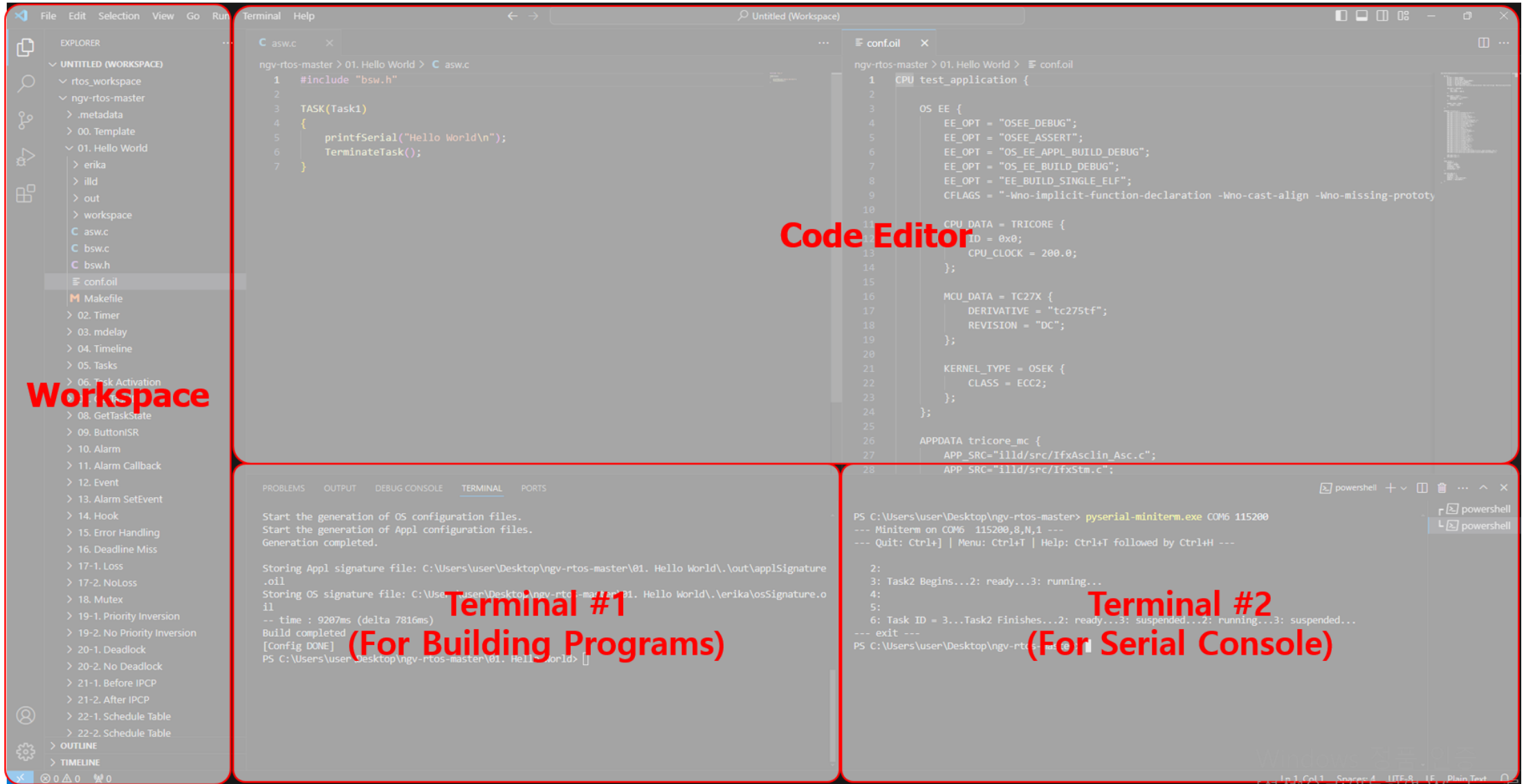
25 watching

Eclipse 기반 IDE (old)

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



Visual Studio Code 기반 개발환경 (new)



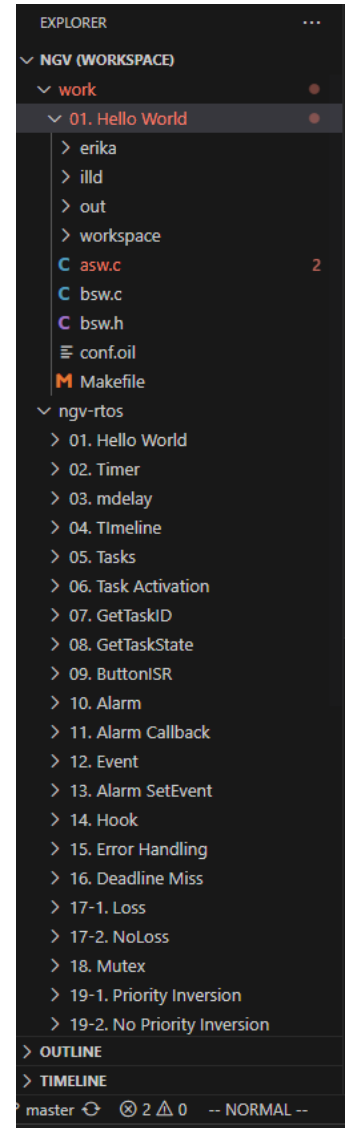
Workspace

- Directory Structure
 - rtos_workspace: working directory
 - ngv-rtos-master: example source files
- Example Projects Directory Structure
 - illd/: Infineon Low Level Driver
 - **asw.c**: Application SW code
 - bsw.c: Basic SW code
 - bsw.h: Basic SW header
 - **conf.oil**: OIL configuration file
 - Makefile: Top-level Makefile

```
work
└─ 01. Hello World
   └─ illd
      asw.c
      bsw.c
      bsw.h
      conf.oil
      Makefile
```

Working

Examples



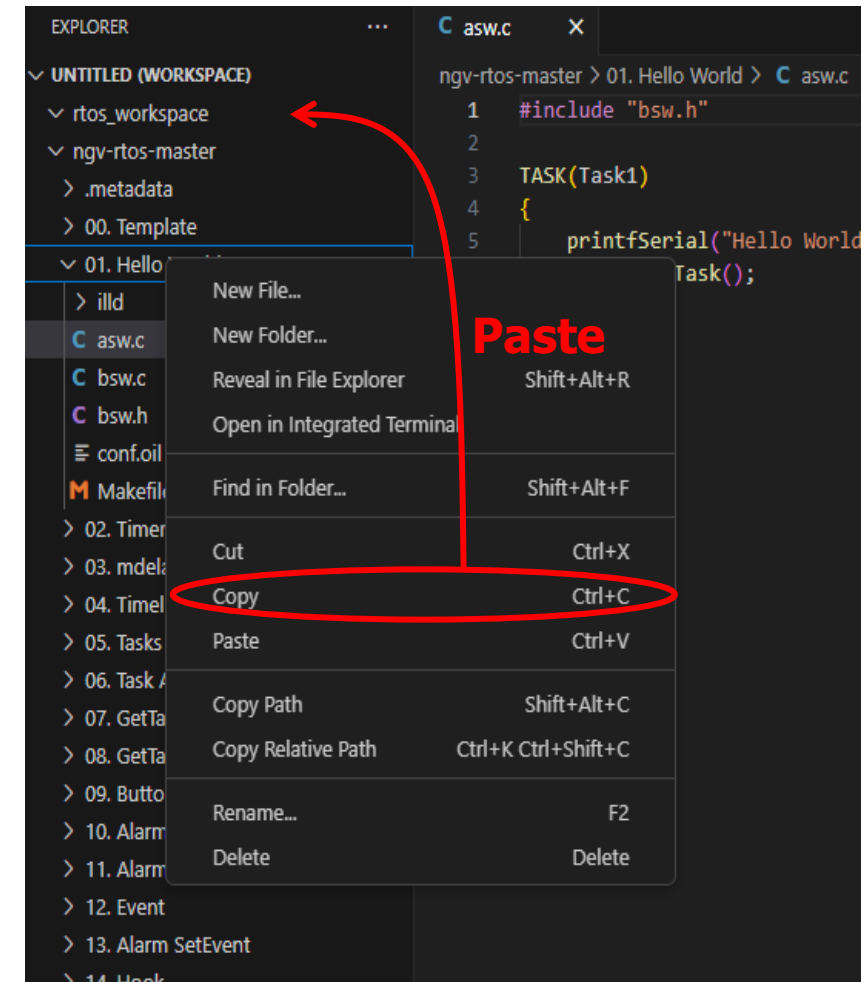
Workflow

- Basic Workflow

- 1) Copy a project (e.g., 01. Hello World) from to working directory
- 2) Edit source files (.C or .H) and OIL files
- 3) Build

- Build Process (in Terminal #1)

- 1) `cd 01. Hello World\`
- 2) `make config`
 - Generate OS kernel files from `conf.oil`
- 3) `make`
 - Generate an executable file

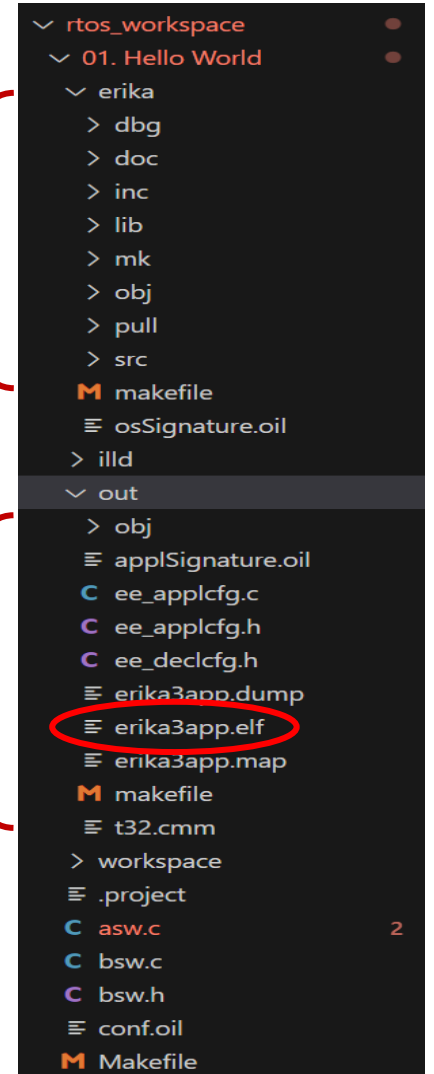


Generated Files

- erika/
 - OS source files
- out/
 - Generated files from the OIL file
 - Object files
 - Executable (ELF) file (erika3app.elf)

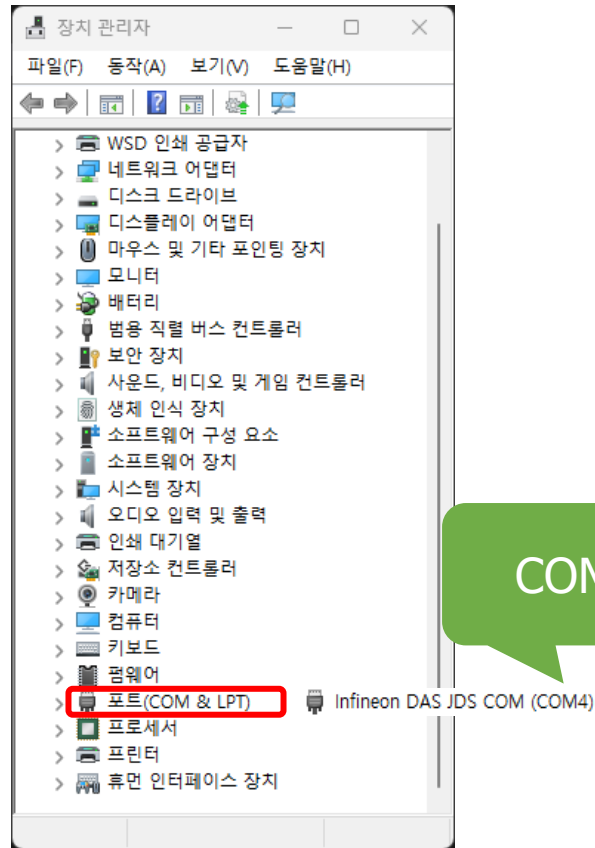
OS source files

Generated files



Serial Console

- 장치관리자에서 COM 포트 확인 (e.g., COM4)
- pyserial-miniterm 이용하여 시리얼 콘솔 시작 (Terminal #2)



COM 포트

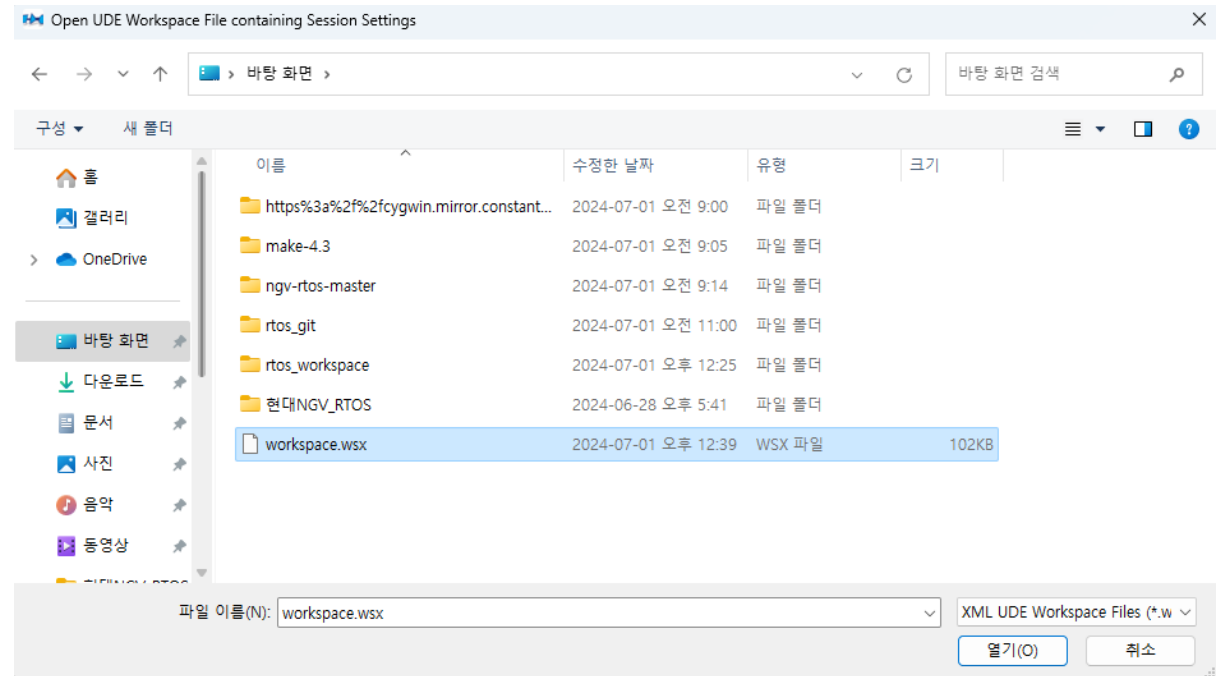
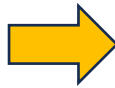
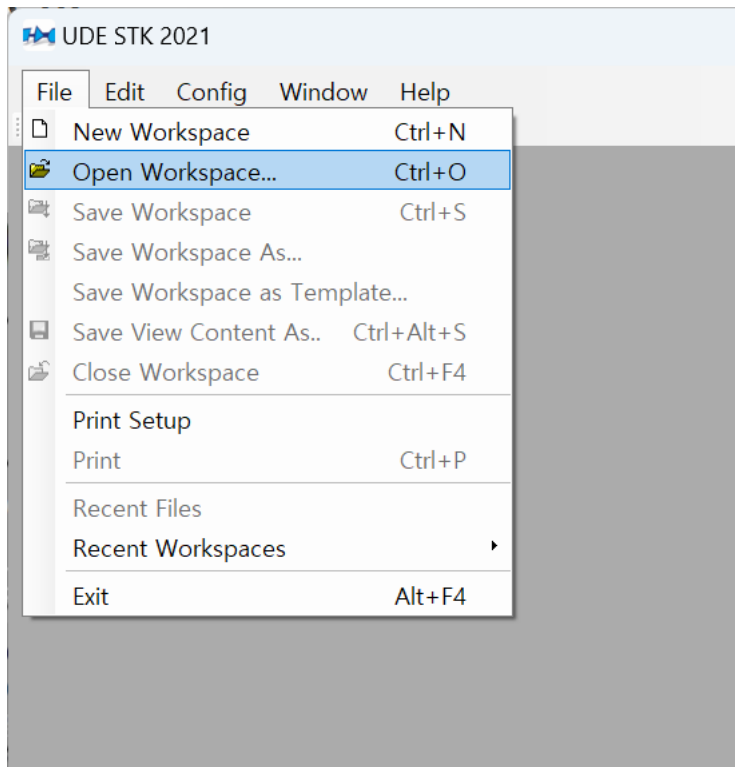
Baudrate

```
PS C:\Users\jongchank\work> pyserial-miniterm.exe COM4 115200
--- Miniterm on COM4 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

```

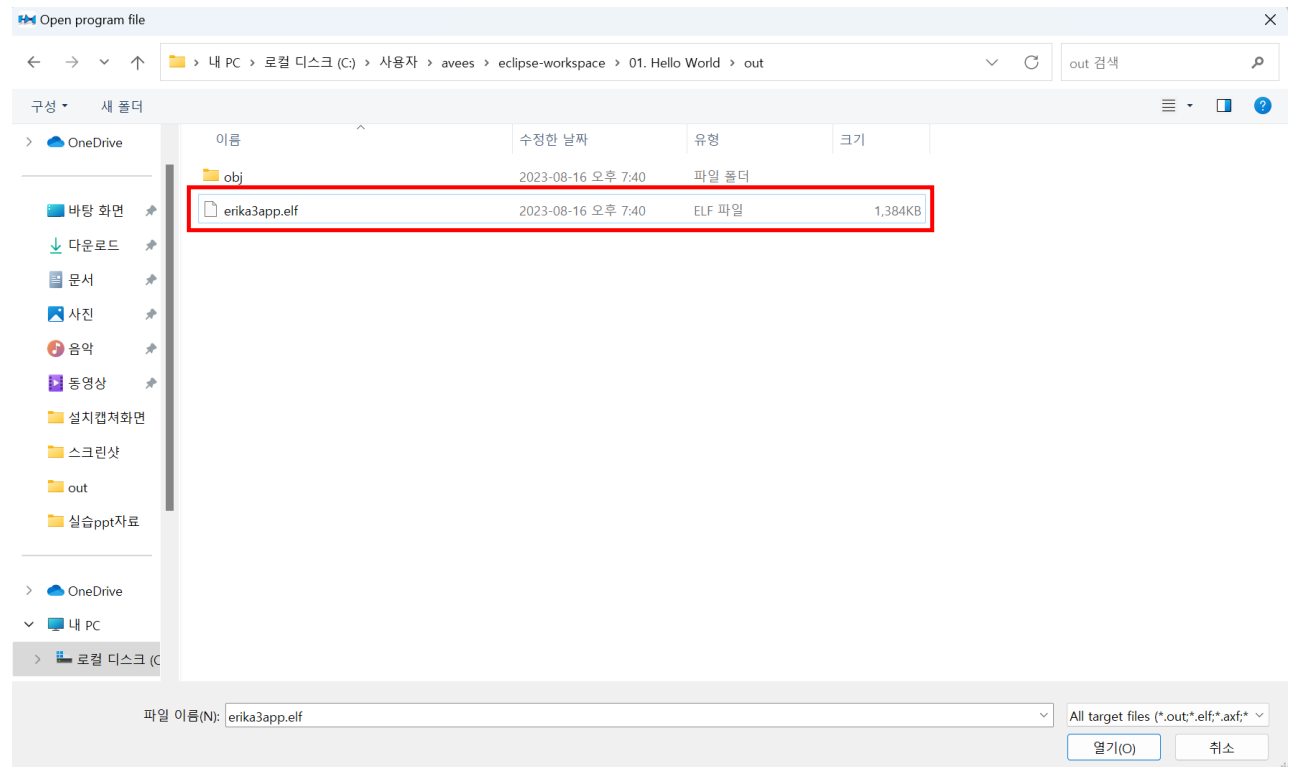
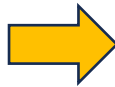
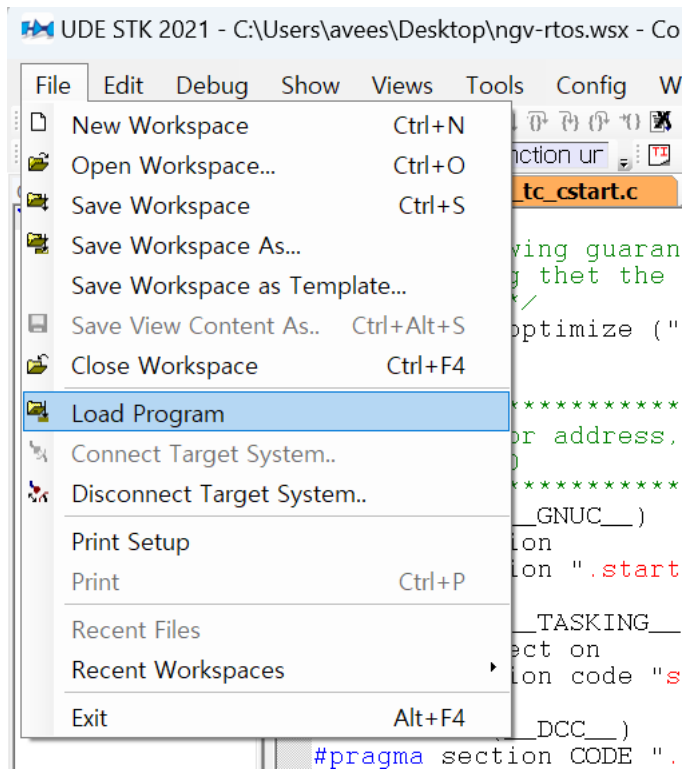

Program Flashing by UDE #1

- File → Open Workspace
- 바탕화면의 workspace.wsx 열기



Program Flashing by UDE #2

- File → Load Program
- 프로젝트 폴더의 out/erika3app.elf 열기

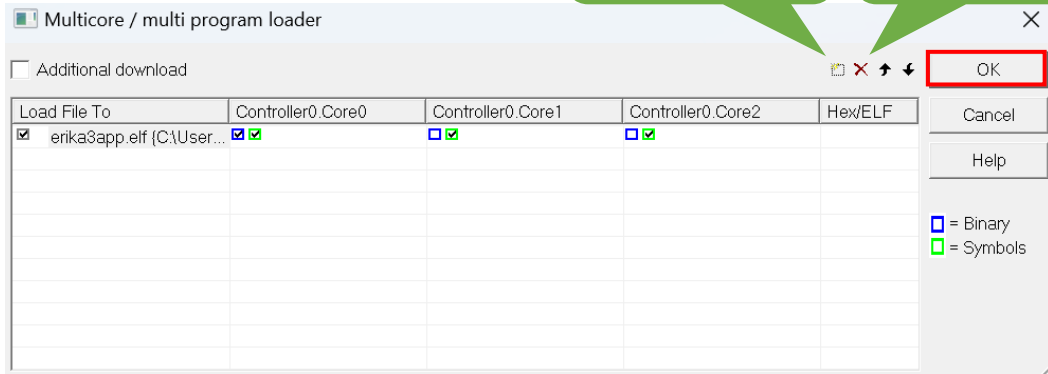


Program Flashing by UDE #3

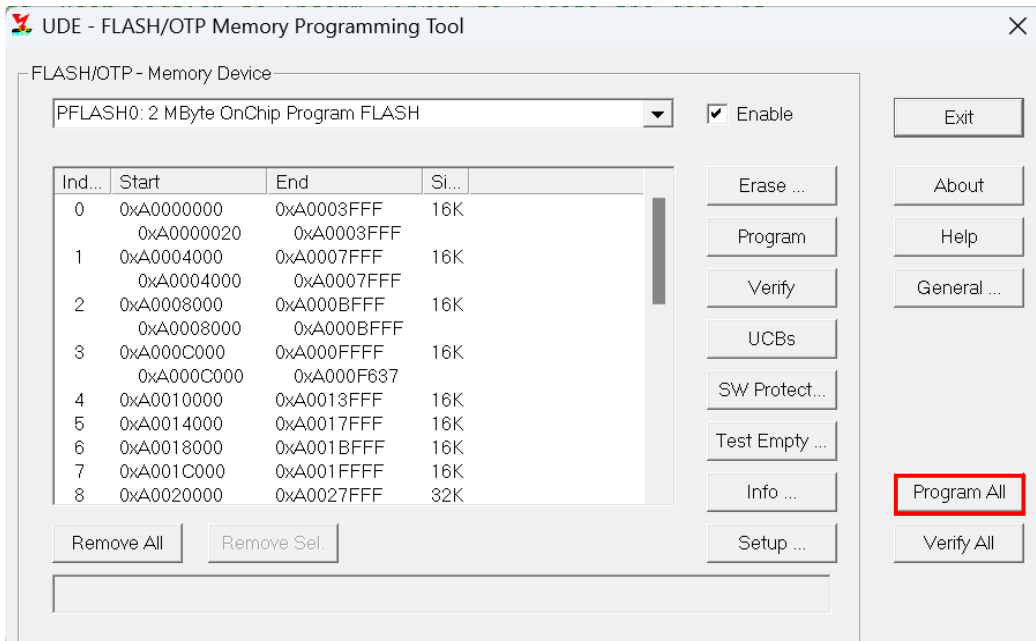
새 파일
열기

기존 파일
해제

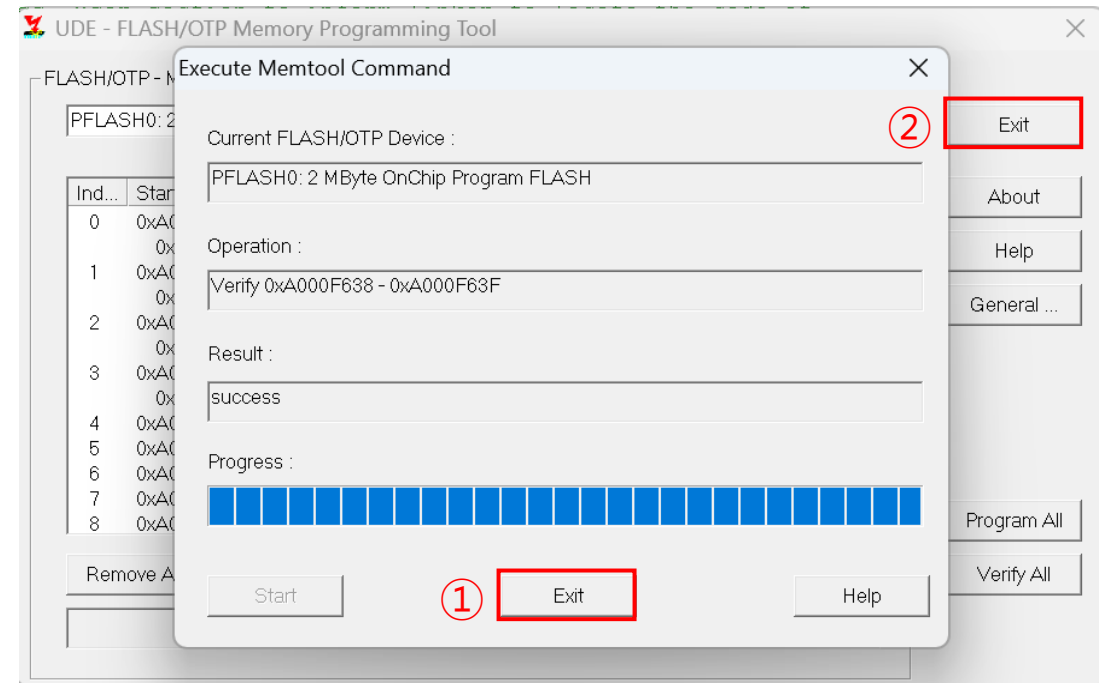
• Step1



• Step2

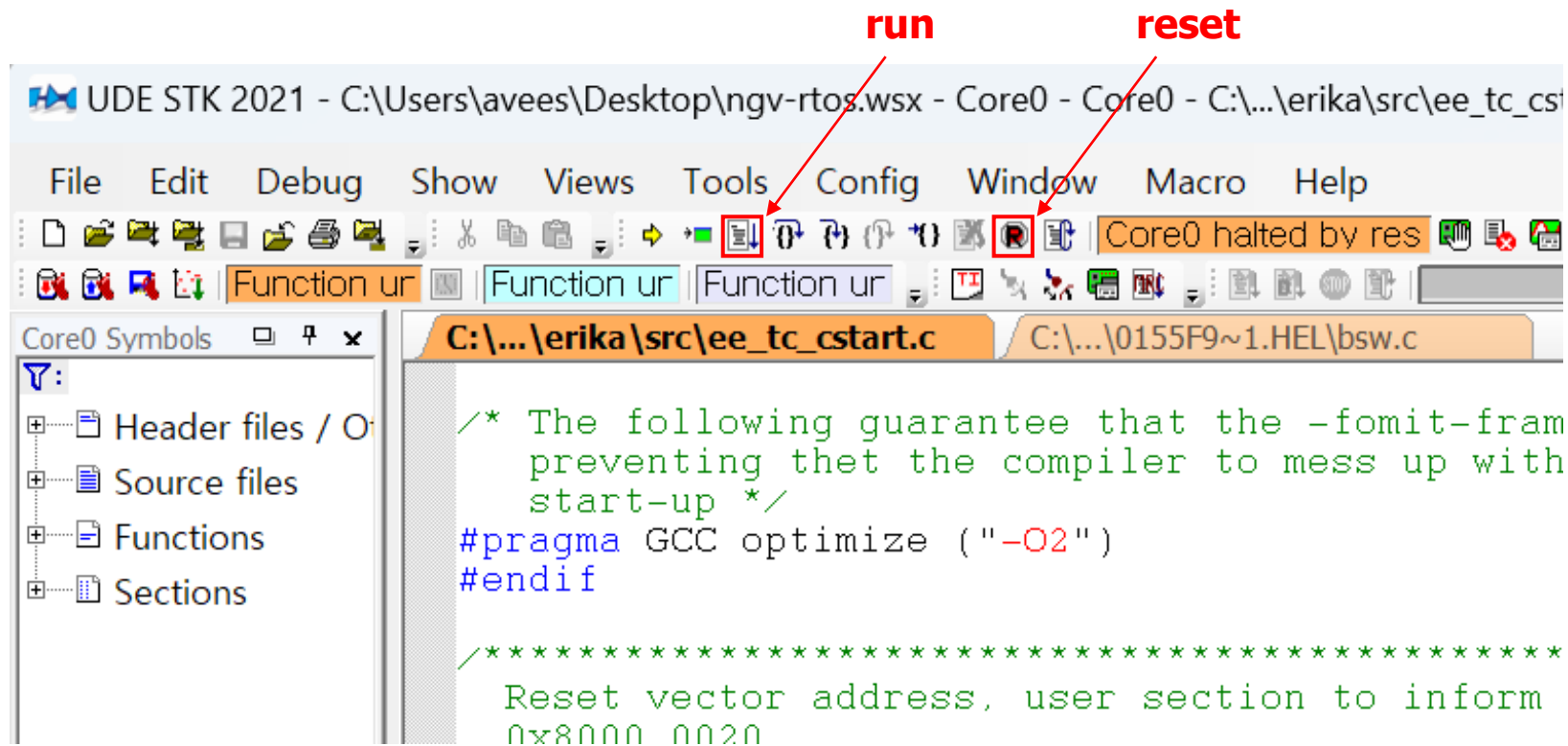


• Step3



Program Flashing by UDE #4

- 프로그램 실행 (run)
- 기존 프로그램 실행중일 경우 reset 후 실행



01. Hello World

- 00. Template 복사
- asw.c에 TASK 추가
- OIL 파일에 TASK 추가
- printfSerial() 함수 사용 (시리얼 콘솔 출력)

Make config는 OIL
변경시에만

```
$ make config  
$ make
```

```
#include "bsw.h"  
  
TASK(Task1)  
{  
    printfSerial("Hello World\n");  
  
    TerminateTask();  
}
```

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

자동
시작

01. Hello World

- OS 시작 후
- Hello World 출력

```
PS C:\Users\jongchank\work> pyserial-miniterm.exe COM4 115200
--- Miniterm on COM4 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

.....
...OS Starts...
.....
Hello World
█
```

02. Timer

- C 파일에 ISR2로 TimerISR 추가
- OIL 파일에 TimerISR 추가
 - Category 2

1초 뒤
interrupt 등록

```
ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000000U);
    printfSerial("Timer\n");
}
```

```
ISR TimerISR {
    CATEGORY = 2;
    SOURCE = "STM0SR0";
    PRIORITY = 2;
};
```

02. Timer

- Hello World 출력후 Timer 반복

```
PS C:\Users\jongchank\work> pyserial-miniterm.exe COM4 115200
--- Miniterm on COM4 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

.....
...OS Starts...
.....
Hello World
Timer
Timer
Timer
Timer
Timer
Timer
[]
```


03. mdelay

- mdelay 함수 이용 3초 실행시간

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

    mdelay(3000);

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

```
PS C:\Users\jongchank\work> pyserial-miniterm.exe COM4 115200
--- Miniterm on COM4 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

.....
...OS Starts...
.....
Hello World
Goodbye World
█
```

04. Timeline

- TimerISR 이용 초단위 Timeline 출력

```
ISR2(TimerISR)
{
    static long c = 0;
    osEE_tc_stm_set_sr0_next_match(1000000U);
    printfSerial("\n%4ld: ", c++);
}
```

```
.....
...OS Starts...
.....
Hello World

0:
1:
2: Goodbye World

3:
4:
5:
6:
7:
8:
9: []
```

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;
    AUTOSTART = TRUE;
    ACTIVATION = 1;
};
```

05. Tasks

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

```
PS C:\Users\jongchank\work> pyserial-miniterm.exe COM4 115200
--- Miniterm on COM4 115200,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

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

06. Task Activation

```
ISR2(TimerISR)
{
    static long c = -4;
    osEE_tc_stm_set_sr0_next_match(1000000U);
    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

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

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
- Unique ID일 뿐 정의된 의미 없음

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

Task2의
ID

Task1의
ID

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?

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

Questions

