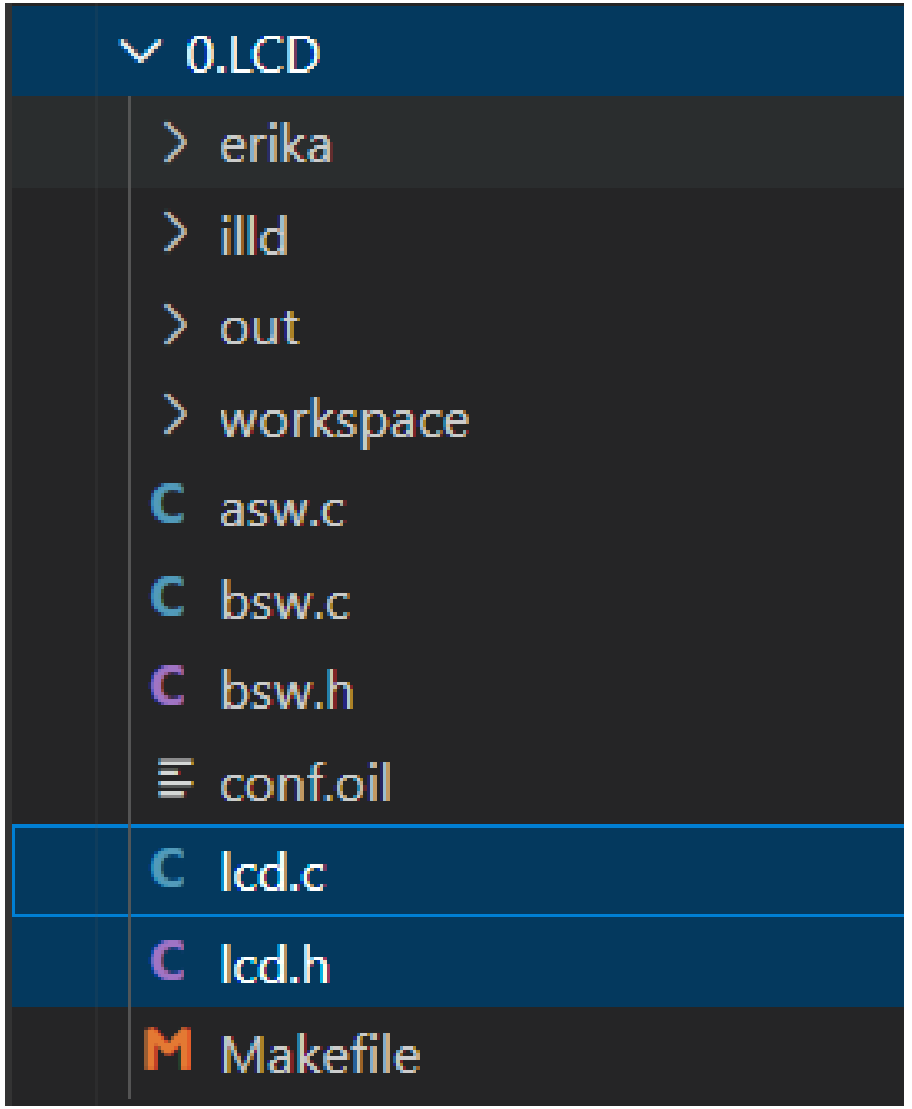
LCD

- 16x2 LCD를 사용하기 위해 간단한 라이브러리 구현
- 프로젝트 내에서 원하는 내용 자유롭게 출력 가능

LCD

- iLLD를 활용한 LCD 제어 함수
- lcd.c, lcd.h로 구성



LCD

- conf.oil

```
APPDATA tricore_mc {
        APP_SRC="illd/src/IfxScuCcu.c";
        APP_SRC="illd/src/IfxScu_PinMap.c";
        …
        APP_SRC="bsw.c";
        APP_SRC="asw.c";
        APP_SRC="lcd.c";
};

TASK TaskLCD {
         PRIORITY = 10;
         STACK = SHARED;
         SCHEDULE = FULL;
         AUTOSTART = FALSE;
         ACTIVATION = 1;
};
```

lcd.c 함수를 사용하기 위해 APP_SRC에 추가

LCD 출력 중 다른 Task에게 동작을 뺏겨 LCD 출력이 망가지는 것을 방지하기 위해 높은 우선순위 적용

LCD

- lcd 함수 사용을 위해 bsw.c 수정

```
#include "illd\src\ConfigurationIsr.h"
…
#include "ee.h“
#include “lcd.h“
#include <string.h>
#include <stdarg.h>
```

LCD 헤더 파일 선언

```
int main(void)
{
    osEE_tc_stm_set_clockpersec();
    osEE_tc_stm_set_sr0(1000000U, 1U);

    UART_init();
    initADC();
    initPeripheralsAndERU();
    lcd_init();
    …
}
```

LCD 초기화 함수

LCD

- asw.c를 수정하여 TaskLCD 구현

```
uint16_t rpm = 1000;

ISR2(TimerISR)
{
    static long c = -4;
    osEE_tc_stm_set_sr0_next_match(1000000U);
    if (c == 0)
        ActivateTask(Task1);
    if (c % 2 == 0)
        ActivateTask(TaskLCD);
    printfSerial("\n%4ld: ", c++);
}
```

LCD에 출력 할 전역변수 선언

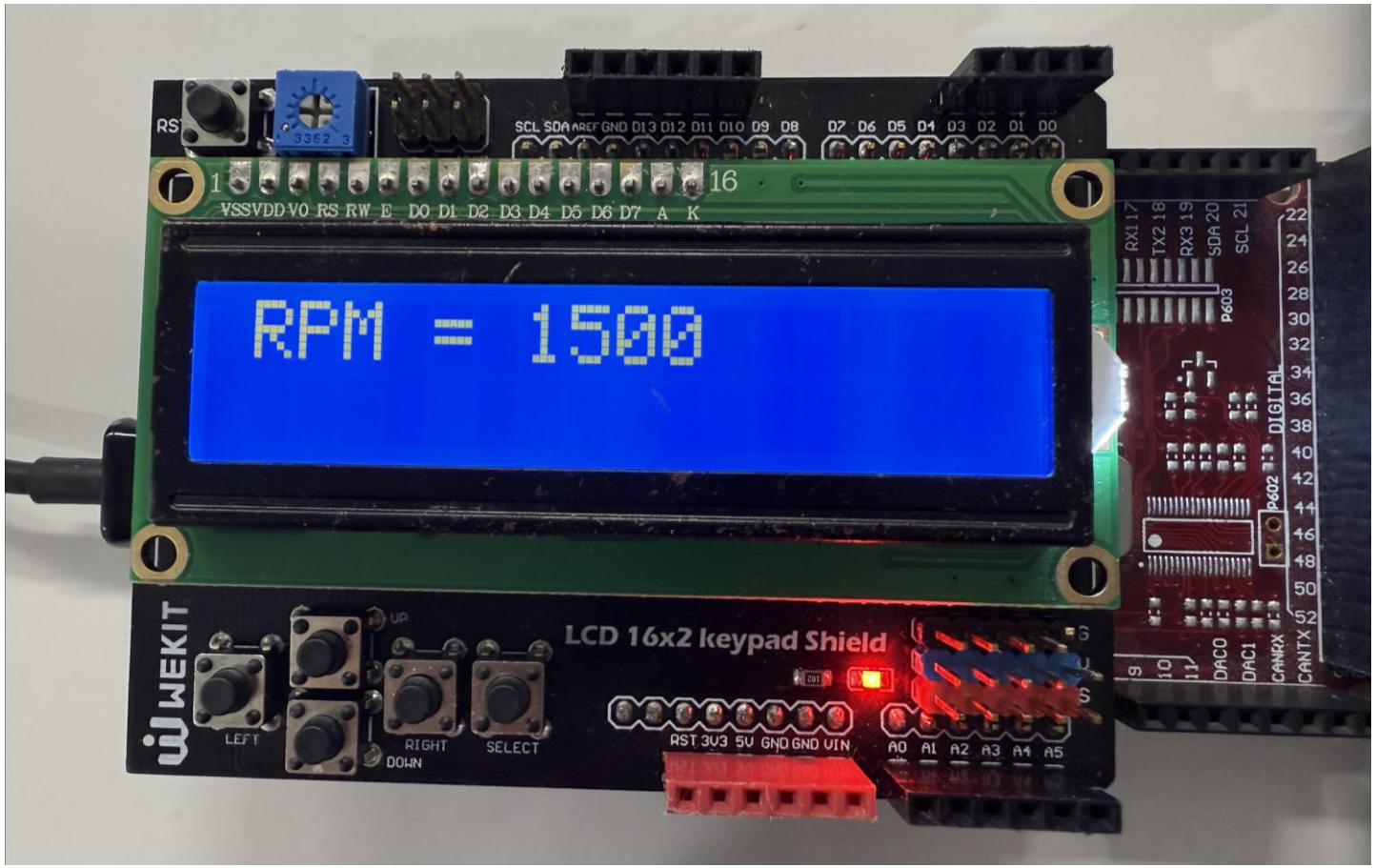
```
TASK(TaskLCD)
{
    rpm += 100;
    lcd_clear();
    char buf[32];
    sprintf(buf, "RPM = %u", rpm);
    lcd_print(buf);
    TerminateTask();
}
```

LCD 화면 초기화 함수

출력할 내용을 문자열로 만들어 LCD에게 전송

LCD

- RPM 값을 일정시간마다 LCD에 출력하는 TaskLCD 생성



LCD

```
TASK(TaskLCD)
{
    rpm += 100;
    lcd_clear();
    char buf[32];
    sprintf(buf, "RPM = %u", rpm);
    lcd_print(buf);
    lcd_goto(1,0);
    lcd_print("Hello World!");
    TerminateTask();
}
```

- lcd_goto(y,x) 함수를 사용하여 LCD 커서 위치 이동
- y 범위 = 0~1
- X 범위 = 0~15

출력 내용이 범위를 넘어가지 않도록 유의

LCD

- 좌표 이동하며 LCD 출력 가능