

## 1, SYS(SYS mode and Configuration)

Debug모드 선택에서 'Serial Wire' 모드를 선택하면

Pinout View에서 PA13, PA14가 각각 TMS, TCK로 생성된다.

용도는 나중에 프로그램 다운로드 후 ST\_Link를 통해 디버깅이 가능하여 실시간으로 코드 진행 상황이나 변수 확인이 가능하여 잘못된 부분 확인 및 수정이 용이하다.

## 2, Uart

```
1) int __io_putchar(int ch){  
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF);  
    return ch;  
}
```

해석

- `__io_putchar` : `__io_`를 붙임으로서 내장 함수 `putchar`가 아닌 임의 함수 이름을 지정
- `HAL_UART_Transmit` : Uart통신에서 Transmit(송신)하겠다는 함수
- `&huart2` : Uart통신에 Uart2포트의 함수를 포인터
- `(uint8_t *)&ch` : 주어진 변수나 표현식의 데이터 형식을 `uint8_t(8bit)`로 형 변환하기 위한 연산자, `&ch`는 위에 `int ch`주소를 얻기 위해 사용
- `1` : 1bit씩 전송
- `0xFFFF` : 전송 작업이 완료되기를 기다리는 타임아웃(전송이 완료될 때까지 무한정 대기)

### 3, FreeRTOS

#### 1) Task and Queues

작업과 큐를 조합하여 복잡한 실시간 응용 프로그램을 개발

예시) 센서 데이터를 수집하고 처리하는 작업과 통신 작업 간에 큐를 사용하여 데이터를 전송하고, 다른 작업에서 데이터를 사용하여 화면에 표시하거나 특정 작업을 수행

- a) **Task(작업)** : 다양한 작업을 병렬로 실행하여 시스템의 처리량을 향상시키고 다양한 기능을 분산하여 처리
- b) **Queues(큐)** : 작업 간에 데이터를 안전하게 공유하거나 통신할 때 사용하여 데이터를 수신, 큐를 사용하면 데이터의 손실 없이 효율적으로 데이터를 전달
- c) **Priority(우선 순위)** : **Task** 스케줄링과 작업 우선 순위에 따라 시스템의 동작을 제어하고 각 작업의 중요도를 결정하는 데 사용되며 이를 통해 실시간 응용 프로그램에서 중요한 작업을 우선적으로 처리하고 효율적으로 작업을 조직

#### 2) Timers and Semaphores

**Timers**를 사용하면 작업의 스케줄링 및 시간 지연을 관리하고, **Semaphores**를 사용하면 작업 간의 협력적인 동작을 구현하고 데이터 공유 및 동기화를 관리

- a) **Timers (타이머)** : 일정한 시간 간격으로 반복 실행되거나 일회성 이벤트를 스케줄링하는 데 사용
- b) **Semaphores (세마포어)** : 공유 자원에 대한 접근을 제어하고 스레드 또는 작업 간의 동기화를 달성하는 데 사용

#### 3) Software Timer

- a) **One-Shot Timer** : **Callback** 함수가 단 한 번만 실행되고 수동으로 재시동을 할 수 있고, 스스로 자동적으로 재시동이 불가하다.
- b) **Auto-Reload Timer** : **Callback** 함수가 실행된 후 자동적으로 재시동이 되고 **Callback** 실행이 주기적으로 이루어짐