1, 용어

1) Enabled : 가능한
2) Disabled : 무능력해진
3) Scan Conversion Mode : 스캔 변환모드
4) Continuous Conversion Mode : 연속 변환 모드

5) 메인 클럭
    a) 외부 클럭(Crystal/Ceramoc Resonator)
    b) 내부 클럭(BYPASS Clock Source)
    c) 내부 클럭보다는 외부 클럭이 정확

6) USART 통신
    a) 비동기식(Asynchronous)
    b) 데이터 수신(HAL_UART_Receive)
    c) 데이터 송신(HAL_UART_Transmit)

7) ADC
    a) ADC(Analong to Digital Converter) : 아날로그-변환기
        i) 마이크로컨트롤러들은 디지털 구성처리기 때문에 이진 및 이진화된 신호만 이해할 수 있다. 그러므로 센서의 데이터를 알고자 하면 센서의 물리적 변화를 전압이나 전류로 변환시켜 ADC로 읽으면 된다.
    b) Sampling Time : 샘플링 타임
        i) 짧은 Sampling Time:
            (1) 장점: 빠른 샘플링 속도를 제공하므로 더 높은 샘플링 속도로 빠른 데이터 변환을 가능하게 합니다.
            (2) 단점: 짧은 샘플링 시간 동안 아날로그 입력 신호를 충분히 안정화하지 못할 수 있으며, 이로 인해 잡음이나 불안정성이 발생할 수 있습니다.
        ii) 중간 Sampling Time:
            (1) 장점: 적절한 안정성과 샘플링 속도의 절충점을 제공합니다. 대부분의 경우에서 충분히 정확한 결과를 얻을 수 있습니다.
            (2) 단점: 빠른 샘플링에 비해 약간 느릴 수 있으며, 정확성과 속도 사이의 균형을 유지합니다.
        iii) 긴 Sampling Time:
            (1) 장점: 아날로그 입력 신호를 충분히 안정화하는 데 시간을 더 많이 허용하므로 정확하고 안정된 결과를 얻을 수 있습니다.

(2) 단점: 느린 샘플링 속도를 가질 수 있어, 빠른 동작이 필요한 애플리케이션에는 적합하지 않을 수 있습니다.

2, 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.

3, 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 putchar가 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, &ch는 위에 int ch주소를 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

   Timers를 사용하면 작업의 스케줄링 및 시간 지연을 관리하고, Semaphores를 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

c) Timer_Task_Priority : 타이머 서비스(Task)의 우선순위를 나타낸다
d) Timer_Queue_Length : 타이머 큐(Queue)의 길이를 설정합니다. 큐에 동시에 저장될 수 있는 타이머 이벤트의 최대 수를 나타냅니다.
e)

4, Timer

1) 용어
   a) Internal clock : 내부 클럭
   b) ARR(Auto reload Register) : 자동 재배치(재적재) 레지스터

2) 계산
   a) PWM주파수(Hz) = TimClock / (Prescaler * Counter Period(ARR))
   b) PWM주기(s) = 1 / PWM주파수 * 초(s)
   c) Pulse = Dutysycle * Counter Period