개발 히스토리

1. 메인의 while 문에서 5ms 간격으로 명령을 하나씩 보내는 방식 성공
2. 태스크 스케줄러를 작성해서 5ms마다 명령을 하나씩 보내는 방식 성공
3. 5ms 태스크의 ow를 돌리면서, 명령을 한번에 와르르 보내는 방식 성공
4. 100ms 태스크의 ts를 돌리면서 req 명령 하나만 보내보기

   → CRC 검증로직에서 결과 불일치로 malfunc 에러 발생

   - 5ms 간격때문에 잘못된 값을 받는것이라고 생각하고 crc 검증로직을 임시삭제
   - 값들이 제대로 들어오지만, 다시한번 요청을 보내면, 타임아웃 에러발생
   - 타임아웃의 원인 -> resolTime이 uint8_t인데 750이 저장되니 오버플로우때문에 이상한 값이 전송됨
   - ow의 entry액션의 대기시간->대기틱 로직을 삭제하고, param 받는 값은 이제 대기틱을 받게끔 수정
5. 수정했더니 read 바이트에서 0xff 값을 많이 받음

   -> 아직 못받았다고 판단 -> pending 상태 유지

   - entry 액션쪽에 브레이크 포인트를 없애니까 값이 완전 정상적으로 들어옴 -> 반복 req_data도 문제없이 처리완료
6. CRC 값이 아예 맞지 않는 버그가 있었음

   → malfunc 관련 로직 합격

   - 코드 원본과 대조 결과, 비트정렬 방식 및 crc 계산 함수의 오타로 확인됨
7. OW계층에서는 stopProc 관련 로직이 거의 없었음 -> rdy_chk에서 이를 감지하면 즉시 종료하고 플러싱 하는걸로 교체
8. 데이터 시트에서의 변환시간과 실제 변환시간에 큰 차이가 있음

   - 시트 ms : 94, 188, 375, 750
   - 실제 ms : 510,510,510,510 → 10번씩 측정해본 결과 모두 102tick을 필요로함
9. OW_ERR_BUS_TEMP_OPEN_CIRCUIT의 판단기준은 틀렸다. 단선되면, floating 되어서 꼭 high신호가 나올 수 없음. 오히려 단선되면 rdy_chk에서 timeout으로 판정됨

   - 애초에 단선감지가 불가능함

-> 센서 반응안함=>응답없음 으로 바꾸는게 더 명확함

10. FND 모듈이 gpio bit bang 방식이기 때문에 core Hz에 영향을 받음

    → SPI 기반 통신 드라이버가 더 적합함 (드라이버 수정완료)

    - 현재 프로젝트의 코어 CLK는 72MHz, FND 드라이버 제작 CLK는 8MHz

      -> 전혀 통신이 되지 않는 상황

11. FND 세그먼트에 표시되는 숫자중 가장 첫번째 숫자만 밝게 빛나고 나머지 숫자는 조금 희미하게 출력됨

    - 어딘가에서 태스크 스케줄러의 1/10 정책이 깨지고 있는 것 같음

      → fnd의 통신라인을 오실로스코프로 확인결과 정책 준수 확인됨

      -> 이 부분은 아직까지 해결못함