

# 런타임 검증

본 테스트 환경은 인위적인 더미 태스크가 아닌, 실제 FW 프로젝트에서 사용되는 센서 처리, 연산, 통신 작업을 기반으로 구성되었다.

이를 통해 스케줄러의 동작은 이론적 시나리오가 아닌 실제 시스템 부하 환경에서 검증되었다.

## 테스트 환경

- MCU : STM32F103C8T6
- CLK : 72MHz
- TIMER 정보
  - 1ms 주기 인터럽트 (스케줄러용)
  - 1us 카운터 (온도센서 타이밍 조율용)
- tasksch\_config.h 의 모든 선택사항에 대해 Enable 을 선택함
- 태스크는 총 6개
- 태스크 정보

모든 태스크는 실행시간을 예측 할 수 있는 작업들로 구성됨

태스크이름	주기 (ms)	오프셋 (ms)	역할
tasksch_userTask_1ms	1	0	FND 출력 → 온도 표시
tasksch_userTask_5ms	5	0	온도센서 통신 계층
tasksch_userTask_10ms	10	0	반복문 연산
tasksch_userTask_100ms	100	0	온도센서 관리 계층 → 온도 변환 요청
tasksch_userTask_500ms	500	0	보드 내부 LED 점등, 퍼텐셔미터 전압 계산
tasksch_userTask_1000ms	1000	10 (sec)	UART로 실행시간 및 오버런 카운트 보내기

- 태스크 측정
  - 모든 태스크의 실행시간 측정을 위한 GPIO 핀 존재 → 오실로스코프로 측정
- IWDG 정보
  - LSI CLK : 40KHZ
  - PRESCALER : 256
  - COUNT : 469
    - $256 / 40000 = 6.4 \text{ ms}$
    - $2000 \text{ (tick)} * 6.4 \text{ (ms)} = 12.8 \text{ sec}$
- OVERRUN COUNT : 500
  - 오버런 상황이 500번 발생해야 워치독을 유도함
- 혹 함수 내용

## UART로 단계를 전송

이름	실행 시점	실행 내용
<b>init-cmplt</b>	초기화가 끝난 시점	"Hello Host!" 전송
<b>major-cycle</b>	<code>tasksch_userTask_1000ms</code> 가 실행된 후의 시점	heartbeat 시그널 전송 → "HB n OR : m"
<b>overrun</b>	오버런 카운트가 지정 문턱값을 넘었을 때	"TOO MANY OR! RESET SCH!"
<b>pre-exit</b>	태스크에서 종료요청을 보냈을 때	"Good Bye Host!" 전송
<b>feedFail</b>	워치독 피딩이 실패했을 때	"feeding fail!" 전송

- Host 단의 MCU 통신 플랫폼 : teraterm

## 테스트 케이스

모든 TC들의 조건 이전 상태는 테스트 환경을 기준으로 함

### ▼ 오류테스트 (TC-RUN-00)

#### ▼ TC-RUN-01

- 목적
  - 실행 중 스케줄러 재초기화 시도를 차단하는지 검증
- 조건
  - main 단에서 `tasksch_init()` 호출
  - task 단에서 또 다시 `tasksch_init()` 호출
- 기대 결과
  - runtime assert 진입
  - runErrCode = `TASKSCH_RUN_ERR_INIT AGAIN`
- 결과

Variables	Breakpoints	Expressions	Registers	Live Expressions	SFRs	X	X	:
Expression	Type							
<code>vTaskSch_info</code>	typSchInfo	{...}						
<code>↳ isInitialized</code>	_Bool	true						
<code>↳ initErrCode</code>	eInitErrCode	TASKSCH_INIT_ERR_OK						
<code>↳ initStatus</code>	eInitStatus	TASKSCH_INIT_COMPLETED						
<code>↳ runErrCode</code>	eRunErrCode	TASKSCH_RUN_ERR_INIT AGAIN						
<code>↳ majorCycle_flag</code>	_Bool	false						
<code>↳ majorCycle_taskExecCnt</code>	uint8_t	0 'W000'						
<code>↳ tasksch_Exit_Flag</code>	_Bool	false						
<code>↳ oneCycle_overRun_count</code>	uint16_t	0						
<code>↳ total_overRun_count</code>	uint16_t	0						
> <code>vTaskSch_totalExecCLK</code>	typExecTimer	{...}						
> <code>vTaskSch_taskList</code>	typTaskInfo [6]	[6]						
<code>+ Add new expression</code>								

#### ▼ TC-RUN-02

- 목적

### 초기화 미 실행후 태스크 실행 방어 검증

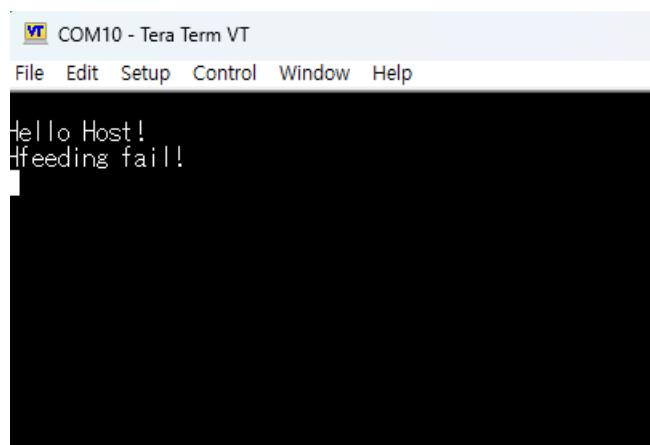
- 조건
  - main 단에서 `tasksch_init()` 호출 X
  - 바로 `tasksch_execTask()` 호출
- 기대 결과
  - runtime assert 진입
  - runErrCode = `TASKSCH_RUN_ERR_INIT_NOT_DONE`
- 결과

↳	vTaskSch_info	typSchInfo
↳	isInitialized	_Bool
↳	initErrCode	eInitErrCode
↳	initStatus	eInitStatus
↳	runErrCode	eRunErrCode
↳	majorCycle_flag	_Bool
↳	majorCycle_taskExecCnt	uint8_t
↳	tasksch_Exit_Flag	_Bool
↳	oneCycle_overRun_count	uint16_t
↳	total_overRun_count	uint16_t
>	vTaskSch_totalExecCLK	typExecTimer
>	vTaskSch_taskList	typTaskInfo [6]
+	Add new expression	[6]

### ▼ TC-RUN-03

- 목적

만약, 위치독 피딩이 실패했을 경우 에러처리가 제대로 이루어지는지 확인
- 조건
  - `tasksch_userFeedFail_watchdogHook` 에 UART로 “feeding fail” 전송하게끔 작성
  - `tasksch_feedWatchdog` 함수에서 피딩이 정상적으로 이루어져도 일부러 `false`를 반환하게끔 코드 수정
- 기대 결과
  - host에서 “feeding fail” 수신
- 결과



dma 버퍼가 깨지기 때문에 어쩔 수 없음.

### ▼ 안정성테스트 (TC-RUN-10)

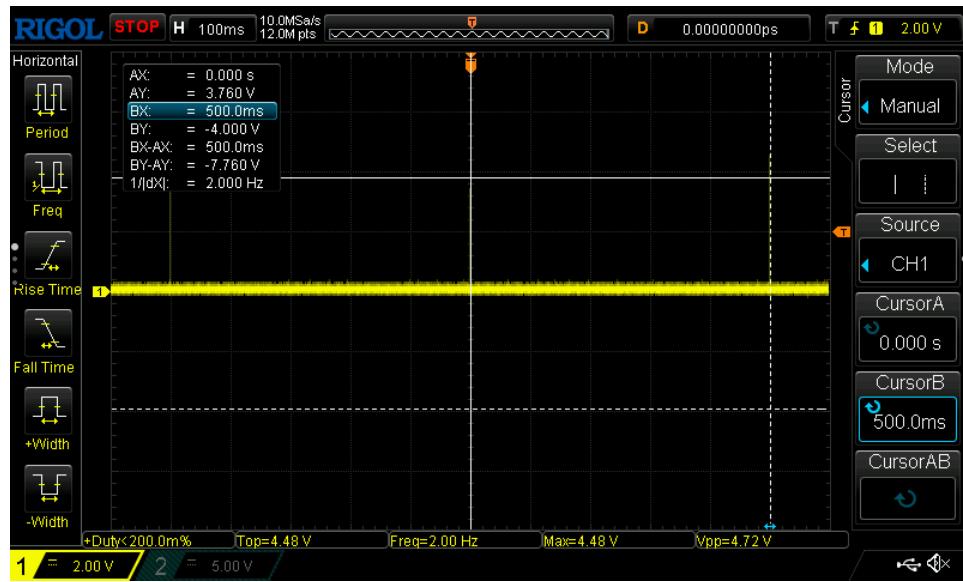
## ▼ TC-RUN-11

- 목적
  - 정상 상태에서 시스템이 불필요하게 깨지지 않음을 검증
- 조건
  - 모든 태스크 내용을 테스트 환경 챕터의 태스크 정보에 맞게끔 작성
  - 이 상태로 30분 동안 동작
  - execClock 객체가 30분에 도달하면, exitReq를 요청 할 것
- 기대 결과
  - teraterm에 "Hello Host!"이 초기화 이후로 확인되지 않을 것
  - 30분 후, "Good Bye Host!" 를 받을 것
- 결과

```
HB_SIG 19, OR : 8
HB_SIG 20, OR : 8
HB_SIG 21, OR : 8
HB_SIG 22, OR : 8
HB_SIG 23, OR : 8
HB_SIG 24, OR : 8
HB_SIG 25, OR : 8
HB_SIG 26, OR : 8
HB_SIG 27, OR : 8
HB_SIG 28, OR : 8
HB_SIG 29, OR : 8
HB_SIG 30, OR : 8
HB
Good Bye Host !
```

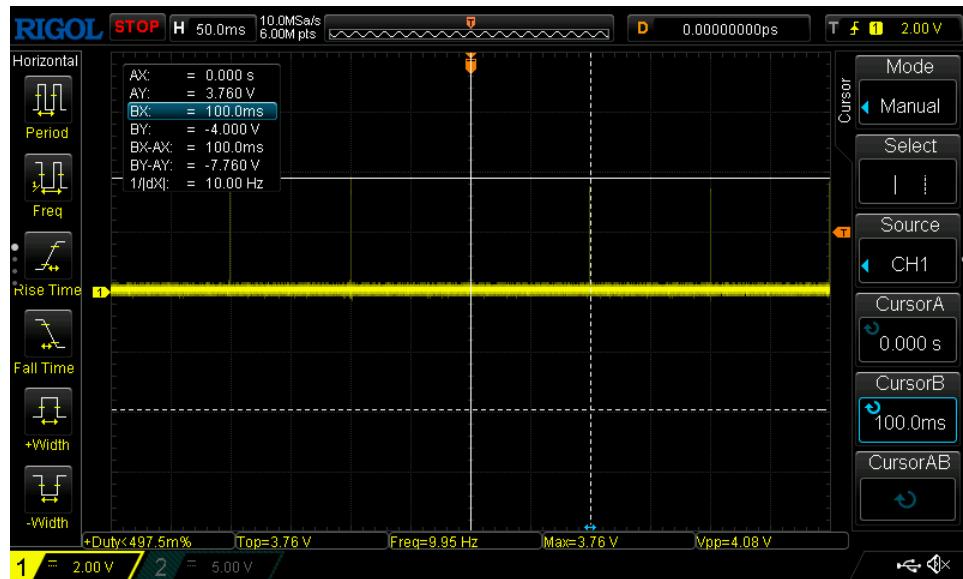
## ▼ TC-RUN-12

- 목적
  - ISR 기반 주기 생성 정확도 검증
- 조건
  - 각 태스크에서 GPIO toggle
  - 오실로스코프로 각 태스크의 실행주기를 측정
- 기대 결과
  - 각 태스크가 주기의 +10% 이내로 실제 주기를 가질 것
- 결과
  - 500ms Task**
    - 커서의 시간차가 500ms 이다



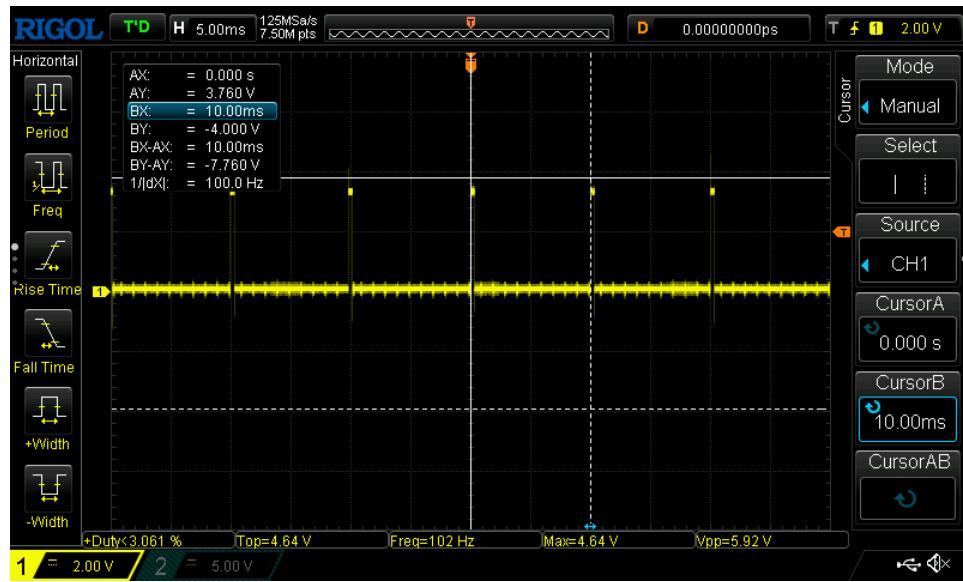
### ○ 100ms Task

커서의 시간차가 100ms 이다



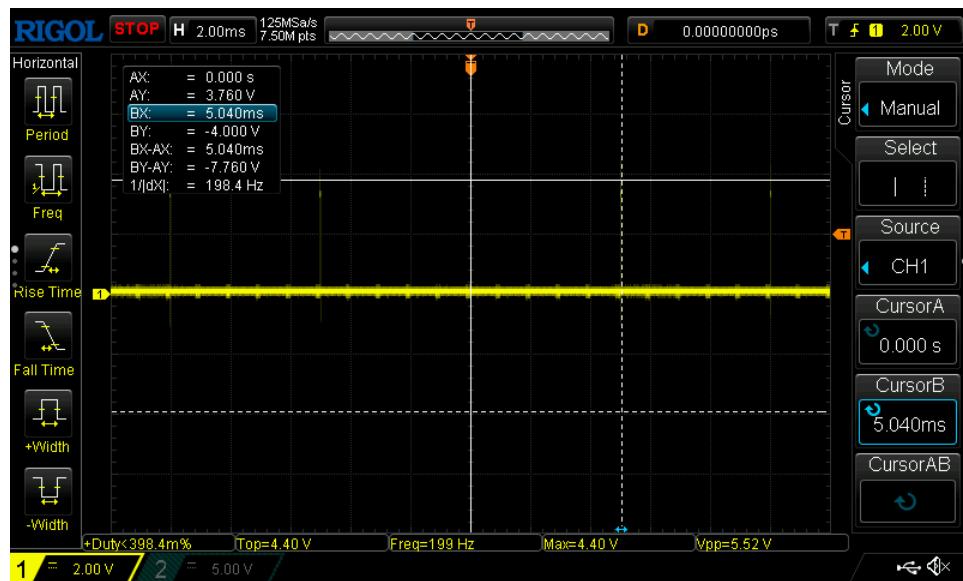
### ○ 10ms Task

커서의 시간차가 10ms이다



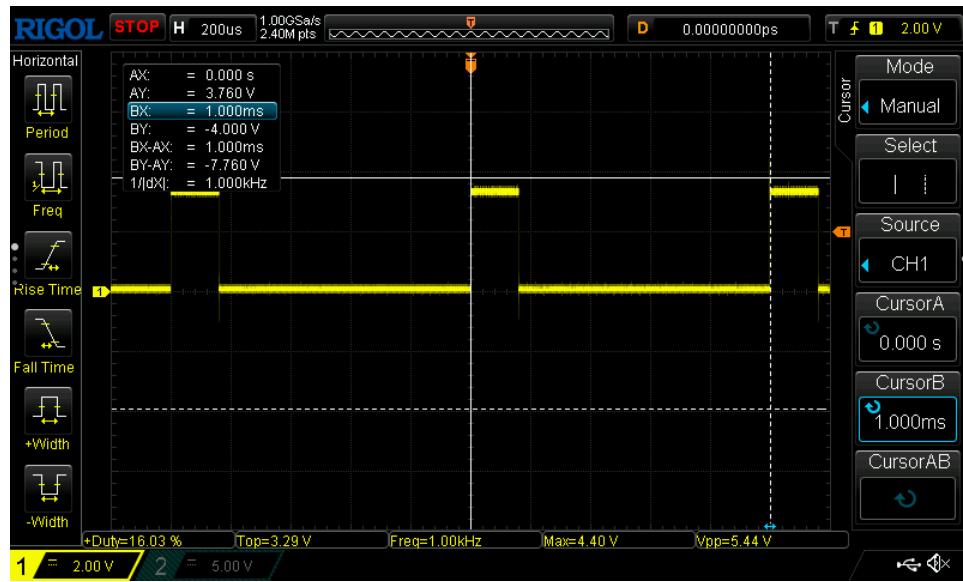
- **5ms Task**

커서의 시간차가 5ms 이다



- **1ms Task**

커서의 시간차가 1ms 이다.



- major cycle

- 아쉽게도 1초주기의 아주짧은 펄스를 저가형 오실로스코프에서 감지할 수 없었음

## ▼ 부하테스트 (TC-RUN-20)

### ▼ TC-RUN-21

- 목적

태스크 starvation 상태를 누적 관찰 후 치명 오류로 판정하는지 검증

- 조건

- dummy\_calculation의 반복변수 5000으로 고정

- 기대 결과

- overrun\_count 누적
- threshold 도달 시, `tasksch_overRun_thrshldExceedHook` 호출됨
- “TOO MANY OR! RESET SCH!”가 출력됨

- 결과

```
Hello Host!
HB_SIG 1, OR : 0
HB_SIG 2, OR : 117
HB_SIG 3, OR : 159
HB_SIG 4, OR : 195
HB_SIG 5, OR : 235
HB_SIG 6, OR : 276
HB_SIG 7, OR : 316
HB_SIG 8, OR : 353
HB_SIG 9, OR : 394
HB_SIG 10, OR : 435
HB
TOO MANY OR! RST SCH!
```