

BlackBox Test

▼ 결과

테스트 케이스 이름	결과
BB-TC011	PASS
BB-TC012	PASS
BB-TC013	PASS
BB-TC014	PASS
BB-TC015	PASS
BB-TC016	PASS
BB-TC017	PASS
BB-TC021	PASS
BB-TC022	PASS
BB-TC023	PASS
BB-TC024	PASS
BB-TC031	PASS
BB-TC032	PASS
BB-TC033	PASS
BB-TC034	PASS
BB-TC041	PASS
BB-TC042	PASS

▼ 테스트용 카테고리

전체 테스트는 4개 카테고리로 분류해볼 수 있다.

▼ 기능 테스트

사용자가 기대하는 기능을 정확하게 수행하는가?

예시)

- 온도 요청 → 12 bit 분해능에서 값 반환 여부
- 분해능 변경 요청 → 데이터 변환 로직이 맞는지?

- stop 요청 → 온도변환 대기 시간에서 즉시 중단하는지?

▼ 에러처리 테스트

실제 발생 가능한 오류를 유도하면 설계된 대로 상태 전이가 발생하는가?

예시)

- **Open Circuit** 만들기 → BUS_NOT_RESPONDING → RECOVERY → 정상 복귀
- **CRC mismatch** 반복 발생 → MALFUNC 에러 전이
- **1-Wire Timeout** 유도 → TEMPSENS_STATE_ERROR 전이
- **NO_DEVICE** 강제 → BUSY 상태에서 ERROR로 전이

▼ 인터페이스 사용성 테스트

사용자가 오용했을 때도 드라이버가 안전하게 거부하는가?

예시)

- 초기화 오용 → 완전한 실패
- 잘못된 파라미터 거부
- 바쁜상태에서의 명령 거부
- 유효범위 밖의 resolution 입력

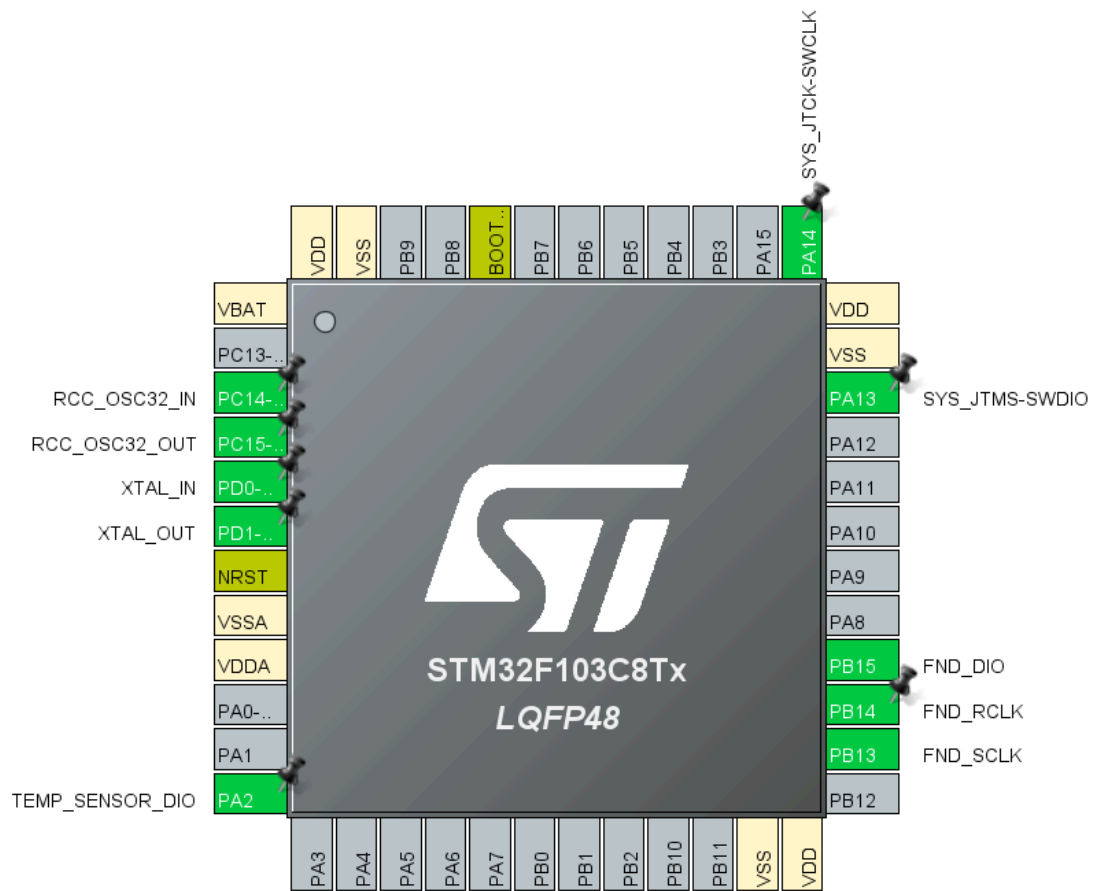
▼ 통합 테스트

상위 계층과의 전체 플로우가 문제 없이 동작하는가?

예시)

- 스케줄러가 1ms 타이머 기반으로 실행되는 동안 TempSensor_100ms 동작 검증
- OneWire 상태 변화에 따라 TempSensor 상태머신이 의도대로 전이하는지
- 연속 명령 시 sequence가 깨지지 않는지

▼ 테스트 환경



▼ TIMER

CLK 주파수: 72MHz

▼ TIM01

Prescaler	72 - 1
Counter	up
Period	1000
NVIC	update Interrupt

1MHz의 clk로 1000번을 세면 1ms 의 인터럽트가 발생
스케줄러에서 사용

▼ TIM02

Prescaler	72 - 1
Counter	up
Period	0xffff

NVIC	X
------	---

oneWire에서 사용

▼ SPI

Mode : Tx only master

NSS : X

BaudRate : 2.25Mbit/s

dataSize : 8bit

1stBit : MSB

CPOL : High

CPHA : 2 Edge

▼ GPIO

- TEMP_SENSOR_DIO
 - SPD : high
 - PullUp/Down R : X
 - mode : openDrain
- FND_RCLK
 - SPD : high
 - PullUp/Down R : pullup
 - mode : push pull

▼ main.c

```
int main(void) {

    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU Configuration-----
    -----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Sy
```

```

stick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_TIM2_Init();
    MX_TIM1_Init();
    /* USER CODE BEGIN 2 */

    /* TIM1 → Scheduler 1ms interrupt */
    __HAL_TIM_CLEAR_IT(&htim1, TIM_IT_UPDATE);
    HAL_TIM_Base_Start_IT(&htim1);
    HAL_NVIC_EnableIRQ(TIM1_UP_TIM10_IRQn);

    TaskSch_init();

    /* TIM2 → oneWire 1us timer */
    HAL_TIM_Base_Start(&htim2);
    oneWire_stMachine_init(FND_DIO_GPIO_Port, FND_DIO_Pin, &htim
2);
    tempSens_stMachine_init();

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1) {

```

```

        /* USER CODE END WHILE */
        TaskSch_execTask();
        /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}

```

▼ TaskSch_userTask.c

```

void userTask02(void) // 5ms
{
#ifdef TASKSCH_USE_MEASUERING
    TaskSch_getCounter(&vTaskInfos[TASKSCH_ID_02].typTemplInfo.startTick);
#endif

    oneWire_process_5ms(); // 5ms 태스크에서 oneWire 상태머신 호출

    vTaskInfos[TASKSCH_ID_02].taskCnt++;

#ifdef TASKSCH_USE_MEASUERING
    TaskSch_getCounter(&vTaskInfos[TASKSCH_ID_02].typTemplInfo.endTick);
    TaskSch_recordExecTime(&vTaskInfos[TASKSCH_ID_02]);
#endif
}

void userTask03(void) // 100ms
{
#ifdef TASKSCH_USE_MEASUERING
    TaskSch_getCounter(&vTaskInfos[TASKSCH_ID_03].typTemplInfo.startTick);
#endif

    tempSensor_100ms(); // 100ms 태스크에서 온도센서 상태머신 호출

    vTaskInfos[TASKSCH_ID_03].taskCnt++;

#ifdef TASKSCH_USE_MEASUERING

```

```

    TaskSch_getCounter(&vTaskInfos[TASKSCH_ID_03].typTemplInfo.e
ndTick);
    TaskSch_recordExecTime(&vTaskInfos[TASKSCH_ID_03]);
#endif
}

```

▼ testCase

▼ 인터페이스 사용성 테스트 (BB-TC01)

초기화

▼ BB-TC011

목적 : 정상적인 절차로 초기화를 진행했을 때 두 객체의 상태가 IDLE로 천이되는가?

절차 :

- `oneWire_stMachine_init()` 에 필요한 파라미터를 대입해서 호출
- `tempSens_stMachine_init()` 를 호출

판정 기준 :

- `oneWire_obj.currState = IDLE` 일 것
- `tempSens_obj.currState = IDLE` 일 것

테스트 결과 :

-

▼ BB-TC012

목적 : oneWire의 초기화가 실패했을 때 tempSenser 초기화가 이루어지지 않는가?

절차 :

- `oneWire_stMachine_init()` 의 파라미터 중에서 하나를 NULL값을 대입해 호출
- `tempSens_stMachine_init()` 를 호출

판정 기준 :

- `oneWire_obj.currState = ERROR` 일 것

- `oneWire_obj.errCode = OW_ERR_CANT_INITIALIZE` 일 것
- `tempSens_obj.currState = IDLE` 일 것
- `tempSens_obj.errCode = TEMPSSENS_ERR_CANT_INITIALIZE` 일 것

테스트 결과 : **PASS**

코드 수정 → oneWire 초기화 함수의 타이머 주소 입력부에 NULL 값을 대입

```
/* TIM2 → oneWire 1us timer */
HAL_TIM_Base_Start(&htim2);
oneWire_stMachine_init(FND_DIO_GPIO_Port, FND_DIO_Pin,
NULL);
tempSens_stMachine_init();
```

Expression	Type	Value
X vTempSensor_obj.errCode	typTempSens_errCode	TEMPSSENS_ERR_NOT_INITIALIZED
X vTempSensor_obj.currState	typTempSens_st	TEMPSSENS_STATE_ERROR
X vOneWire_obj.errCode	typErrCodeOW	OW_ERR_NOT_INITIALIZED
X vOneWire_obj.currState	typStatesOW	OW_STATE_ERROR
+ Add new expression		

▼ BB-TC013

목적 : 모든 초기화가 끝난 상황에서 tempSenser의 재초기화가 일어났을 때 여러상태로 전환되는가?

절차 :

- `oneWire_stMachine_init()` 에 필요한 파라미터를 대입해서 호출
- `tempSens_stMachine_init()` 를 호출
- `oneWire_obj` 와 `tempSens_obj` 의 `currState = IDLE` 을 확인
- `tempSens_stMachine_init()` 를 다시 호출

판정 기준 :

- `oneWire_obj` 와 `tempSens_obj` 의 `currState = ERROR`
- `oneWire_obj` 와 `tempSens_obj` 의 `errCode = TEMPSSENS_ERR_ALREADY_INIT`

테스트 결과 : **PASS**

초기화 성공

Expression	Type	Value
<code>X= vTempSensor_obj.errCode</code>	<code>typTempSens_errCode</code>	<code>TEMPSENS_ERR_OKAY</code>
<code>X= vTempSensor_obj.currState</code>	<code>typTempSens_st</code>	<code>TEMPSENS_STATE_IDLE</code>
<code>X= vOneWire_obj.errCode</code>	<code>typErrCodeOW</code>	<code>OW_ERR_OKAY</code>
<code>X= vOneWire_obj.currState</code>	<code>typStatesOW</code>	<code>OW_STATE_IDLE</code>
<code>+ Add new expression</code>		

1s 태스크에서 `oneWire_reInit_flag` 를 대입함으로써, 재초기화를 진행하게끔 코드를 수정

```
if(oneWire_reInit_flag == true)
{
    oneWire_reInit_flag = false;
    tempSens_stMachine_init();
}
```

Expression	Type	Value
<code>X= vTempSensor_obj.errCode</code>	<code>typTempSens_errCode</code>	<code>TEMPSENS_ERR_ALREADY_INIT</code>
<code>X= vTempSensor_obj.currState</code>	<code>typTempSens_st</code>	<code>TEMPSENS_STATE_ERROR</code>
<code>X= vOneWire_obj.errCode</code>	<code>typErrCodeOW</code>	<code>OW_ERR_OKAY</code>
<code>X= vOneWire_obj.currState</code>	<code>typStatesOW</code>	<code>OW_STATE_IDLE</code>
<code>X= oneWire_reInit_flag</code>	<code>_Bool</code>	<code>false</code>
<code>+ Add new expression</code>		

`tempSens`를 재초기화 하였기 때문에 그 하위계층인 `oneWire`는 정상상태고 `tempSens` 계층만 오류가 발생함을 확인가능

▼ BB-TC014

목적 : 사용자가 실수로 초기화 함수를 호출하지 않고 상태머신을 호출하였을 때 에러상태로 천이되는가?

절차 : **PASS**

- 초기화 함수를 아예 호출하지 않고 상태머신 주기적으로 호출
- `tempSensor_reqCommand(REQ_DATA)` 요청

판정 기준 :

- `oneWire_obj` 와 `tempSens_obj` 의 `currState = ERROR` 를 확인
- `oneWire_obj` 와 `tempSens_obj` 의 `errCode = NOT_INITIALIZED` 를 확인
- `tempSensor_reqCommand` 반환값 `false` 확인

테스트 결과 :

main함수의 온도센서 객체 초기화 하는 부분을 주석처리하였음

```
/* TIM2 → oneWire 1us timer */
HAL_TIM_Base_Start(&htim2);
//oneWire_stMachine_init(FND_DIO_GPIO_Port, FND_DIO_Pi
n, &htim2);
//tempSens_stMachine_init();
```

Expression	Type	Value
<code>x= vTempSensor_obj.errCode</code>	<code>typTempSens_errCode</code>	<code>TEMPSENS_ERR_NOT_INITIALIZED</code>
<code>x= vTempSensor_obj.currState</code>	<code>typTempSens_st</code>	<code>TEMPSENS_STATE_ERROR</code>
<code>x= vOneWire_obj.errCode</code>	<code>typErrCodeOW</code>	<code>OW_ERR_NOT_INITIALIZED</code>
<code>x= vOneWire_obj.currState</code>	<code>typStatesOW</code>	<code>OW_STATE_ERROR</code>
<code>x= oneWire_relnit_flag</code>	<code>_Bool</code>	<code>false</code>
<code>+ Add new expression</code>		

요청과정

▼ BB-TC015

목적 : 상태가 PENDING/EXEC 일 때 작업 요청이 거부 되는가?

절차 :

- `oneWire_obj` 와 `tempSens_obj` 의 `currState = IDLE` 를 확인
- `tempSensor_reqCommand(REQ_DATA)` 요청
- 다음 100ms 틱에서 `tempSensor_reqCommand(REQ_DATA)` 요청

판정 기준 :

- `tempSensor_reqCommand(REQ_DATA)` 반환값 `false` 확인

테스트 결과 :

▼ BB-TC016

목적 : 파라미터 없이 SET_RESOL 요청시 에러 상태로 천이되는가?

절차 :

- `oneWire_obj` 와 `tempSens_obj` 의 `currState = IDLE` 를 확인
- `tempSens_obj.currCmd.cmd = SET_RESOL`
- `SET_RESOL(param=NULL)`

판정 기준 :

- 요청 거부
- `errCode == TEMPSENS_ERR_INVALID_CMD`
- `tempSens_obj.currCmd.cmd = TEMPSENS_CMD_WAIT` 변화 없음

테스트 결과 :

•

▼ BB-TC017

목적 : 잘못된 파라미터로 SET_RESOL 요청시 에러 상태로 천이되는가?

절차 :

- `oneWire_obj` 와 `tempSens_obj` 의 `currState = IDLE` 를 확인
- `tempSens_obj.currCmd.cmd = SET_RESOL`
- `SET_RESOL(param=8)`
- `SET_RESOL(param=13)`

판정 기준 :

- 요청 거부
- `errCode == TEMPSENS_ERR_INVALID_CMD`
- `tempSens_obj.currCmd.cmd = TEMPSENS_CMD_WAIT` 변화 없음

테스트 결과 :

•

▼ 기능 테스트 (BB-TC02)

▼ BB-TC021

목표 :

SET_RESOL 명령이 정상 처리되고 센서의 resolution byte가 기대값으로 변경되는가?

절차 :

- tempSensor가 초기화 완료될 것
- 1sec Task의 첫 틱에서 `tempSensor_reqCommand(SET_RESOL, param=9)`
- 다음 틱에서 `tempSensor_reqCommand(REQ_DATA, NULL)`
- 다음 tick에서 `tempSensor_getResolution()` 확인

판정 기준 :

- 다음 REQ_DATA 결과로 해당 분해능으로 변환됨
- `errCode = OKAY`

테스트 결과 :

▼ 테스트 코드

```
void TempSensor_Dummy_TC02_1(void) // 1sec task에서 돌리기
{

    static bool tc02_1_start_setResol_flag = false;
    static bool tc02_1_failed_setResol_flag = false;
    static bool tc02_1_start_readResol_flag = false;
    static bool tc02_1_failed_readResol_flag = false;
    static uint8_t tc02_1_test_setResol = 9;
    static uint8_t tc02_1_test_readResol = 0;
    static bool tc02_1_result_setResol_flag = false;
    static bool tc02_1_result_readResol_flag = false;

    if(tc02_1_start_setResol_flag == false)
    {
        tc02_1_result_setResol_flag = tempSensor_reqCommand(TEMPSENS_CMD_SET_RESOL,&tc02_1_test_setResol,&tc02_1_failed_setResol_flag);
        tc02_1_start_setResol_flag = true;
        /* 기대값
        * tc02_1_start_setResol_flag = true
```

```

        * tc02_1_result_setResol_flag = true
        * tc02_1_failed_setResol_flag = false
        */
    }

    else if(tc02_1_start_setResol_flag == true && tc02_1_start_readResol_flag == false)
    {
        tc02_1_result_readResol_flag = tempSensor_reqCommand(TEMPSENS_CMD_REQ_DATA,NULL,&tc02_1_failed_readResol_flag);
        tc02_1_start_readResol_flag = true;
        /* 기대값
        * tc02_1_start_readResol_flag = true
        * tc02_1_result_readResol_flag = true
        * tc02_1_failed_readResol_flag = false
        */
    }

    else if(tc02_1_start_setResol_flag == true && tc02_1_start_readResol_flag == true)
    {
        tc02_1_test_readResol = tempSensor_getResolution();
        /* 기대값
        * tc02_1_test_readResol = 9
        */
    }
}

```

H= Variables × Breakpoints Expressions Registers Live Expressions SFRs		
Name	Type	Value
X= tc02_1_start_setResol_flag	_Bool	true
X= tc02_1_failed_setResol_flag	_Bool	false
X= tc02_1_start_readResol_flag	_Bool	true
X= tc02_1_failed_readResol_flag	_Bool	false
X= tc02_1_test_setResol	uint8_t	9 '9'
X= tc02_1_test_readResol	uint8_t	9 '9'
X= tc02_1_test_errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY
X= tc02_1_result_setResol_flag	_Bool	true
X= tc02_1_result_readResol_flag	_Bool	true

▼ BB-TC022

목표 : 사용자가 요청한 온도 읽기 명령이 정상 실행되고 올바른 데이터가 반환되는가?

절차 :

- tempSensor가 초기화 완료될 것
- 1sec Task의 첫 틱에서 `tempSensor_reqCommand(REQ_DATA)` 요청 (이전 분해능 9bit)
- 다음 틱에서 `tempSensor_getTemper_celcius()` 확인

판정 기준 :

- 온도 값을 정상적으로 읽을 것
- 분해능이 9bit 인지 확인할 것
- `errCode = OKAY`

테스트 결과 :

▼ 테스트 코드

```
void TempSensor_Dummy_TC02_2(void)
{
    static bool tc02_2_start_readTemp_flag = false;
    static bool tc02_2_failed_readTemp_flag = false;
    static bool tc02_2_result_readTemp_flag = false;
    static float tc02_2_test_readTemp = 0.0;
    static uint8_t tc02_2_test_readResol = 0;
    static typTempSens_errCode tc02_2_test_errCode = TEMPSENS_ERR_OKAY;

    if (tc02_2_start_readTemp_flag == false)
    {
        tc02_2_result_readTemp_flag = tempSensor_reqCommand(TEMPSENS_CMD_SET_RESOL,NULL,&tc02_2_failed_readTemp_flag);
        tc02_2_start_readTemp_flag = true;
        /* 기대값
        * tc02_2_start_readTemp_flag = true
        * tc02_2_result_readTemp_flag = true
```

```

        * tc02_2_failed_readTemp_flag = false
        */
    }
    else
    {
        tc02_2_test_readTemp = tempSensor_getTemper_cel
cius();
        tc02_2_test_readResol = tempSensor_getResolution
();
        tc02_2_test_errCode = tempSensor_getErrCode();
        /* 기대값
        * tc02_2_test_readTemp = ???
        * tc02_2_test_readResol = 12
        * tc02_2_test_errCode = TEMPSENS_ERR_OKAY
        */
    }
}

```

Name	Type	Value
X= tc02_2_start_readTemp_flag	_Bool	true
X= tc02_2_failed_readTemp_flag	_Bool	false
X= tc02_2_result_readTemp_flag	_Bool	true
X= tc02_2_test_readTemp	float	22.5
X= tc02_2_test_readResol	uint8_t	9 '9'
X= tc02_2_test_errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY

▼ BB-TC023

목표 : 센서 EEPROM 저장/복구 루틴이 정상작동하는가?

절차 :

- SET_RESOL(11)
- SAVE_RESOL
- SET_RESOL(9)
- RECOV_PREV_RESOL
- `getResolution()` 확인

판정 기준 :

- 최종 resolution이 11이어야 함
- `errCode = OKAY`

테스트 결과 :

▼ 테스트 코드

```
void TempSensor_Dummy_TC02_3(void)
{
    static uint8_t tc02_3_testSeq = 0;

    static bool tc02_3_result_setResol_flag = false;
    static bool tc02_3_failed_setResol_flag = false;
    static uint8_t tc02_3_test_setResol = 11;

    static bool tc02_3_result_saveResol_flag = false;
    static bool tc02_3_failed_saveResol_flag = false;

    static bool tc02_3_result_setResol2_flag = false;
    static bool tc02_3_failed_setResol2_flag = false;
    static uint8_t tc02_3_test_setResol2 = 9;

    static bool tc02_3_result_recallResol_flag = false;
    static bool tc02_3_failed_recallResol_flag = false;

    static bool tc02_3_result_readResol_flag = false;
    static bool tc02_3_failed_readResol_flag = false;

    static uint8_t tc02_3_test_readResol = 0;
    static typTempSens_errCode tc02_3_test_errCode = TEMPSENS_ERR_OKAY;

    switch(tc02_3_testSeq)
    {
    case 0:
        tc02_3_result_setResol_flag = tempSensor_reqComm
        and(
            TEMPSENS_CMD_SET_RESOL, &tc02_3_test_set
            Resol, &tc02_3_failed_setResol_flag);
```

```

tc02_3_testSeq++;
/* 기대값
 * tc02_3_result_setResol_flag = true
 * tc02_3_failed_setResol_flag = false
 */
break;
case 1:
tc02_3_result_saveResol_flag = tempSensor_reqCom
mand(
    TEMPSENS_CMD_SAVE_RESOL, NULL, &tc02_3_
failed_saveResol_flag);
tc02_3_testSeq++;
/* 기대값
 * tc02_3_result_saveResol_flag = true
 * tc02_3_failed_saveResol_flag = false
 */
break;
case 2:
tc02_3_result_setResol2_flag = tempSensor_reqCom
mand(
    TEMPSENS_CMD_SET_RESOL, &tc02_3_test_set
Resol2, &tc02_3_failed_setResol2_flag);
tc02_3_testSeq++;
/* 기대값
 * tc02_3_result_setResol2_flag = true
 * tc02_3_failed_setResol2_flag = false
 */
break;
case 3:
tc02_3_result_recallResol_flag = tempSensor_reqCo
mmand(
    TEMPSENS_CMD_RECOV_PREV_RESOL, NULL, &t
c02_3_failed_recallResol_flag);
tc02_3_testSeq++;
/* 기대값
 * tc02_3_result_recallResol_flag = true
 * tc02_3_failed_recallResol_flag = false
 */

```

```

        break;
    case 4:
        tc02_3_result_readResol_flag = tempSensor_reqCom
mand(
            TEMPSENS_CMD_REQ_DATA, NULL, &tc02_3_fail
ed_readResol_flag);
        tc02_3_testSeq++;
        /* 기대값
        * tc02_3_result_readResol_flag = true
        * tc02_3_failed_readResol_flag = false
        */
        break;
    case 5:
        tc02_3_test_errCode = tempSensor_getErrCode();
        tc02_3_test_readResol = tempSensor_getResolution
();
        /* 기대값
        * tc02_3_test_errCode = okay
        * tc02_3_test_readResol = 11
        */
        tc02_3_testSeq++;
        break;
    default:
        break;
    }
}

```

Name	Type	Value
X= tc02_3_testSeq	uint8_t	6 'W006'
X= tc02_3_result_setResol_flag	_Bool	true
X= tc02_3_failed_setResol_flag	_Bool	false
X= tc02_3_test_setResol	uint8_t	11 '#v'
X= tc02_3_result_saveResol_flag	_Bool	true
X= tc02_3_failed_saveResol_flag	_Bool	false
X= tc02_3_result_setResol2_flag	_Bool	true
X= tc02_3_failed_setResol2_flag	_Bool	false
X= tc02_3_test_setResol2	uint8_t	9 '#t'
X= tc02_3_result_recallResol_flag	_Bool	true
X= tc02_3_failed_recallResol_flag	_Bool	false
X= tc02_3_result_readResol_flag	_Bool	true
X= tc02_3_failed_readResol_flag	_Bool	false
X= tc02_3_test_readResol	uint8_t	11 '#v'
X= tc02_3_test_errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY

▼ BB-TC024

목표 : 진행 중이던 어떤 명령이라도 STOP 명령이 오면 즉시 중단되는가?

절차 :

- REQ_DATA 요청
- 약 300ms 후 STOP 명령 요청
- 다음 100ms tick에서 IDLE 복귀 여부 확인

판정 기준 :

- `tempSens_isReady_forReq() = true`
- `oneWire_obj.cmdNum = 0 or 1`
- `tempSensor_obj.errCode = OKAY`

테스트 결과 :

▼ 테스트 코드

```
void TempSensor_Dummy_TC02_4(void) // 100ms task
{
    static uint16_t tc02_4_seq = 0;
    static bool tc02_4_result_readTemp_flag = false;
    static bool tc02_4_failed_readTemp_flag = false;
    static bool tc02_4_result_stopProc_flag = false;
    static bool tc02_4_failed_stopProc_flag = false;

    static typTempSens_errCode tc02_4_test_errCode = TEMPSENS_ERR_OKAY;
    static bool tc02_4_test_ready = false;

    switch(tc02_4_seq)
    {
        case 0:
            tc02_4_result_readTemp_flag = tempSensor_reqCommand(
                TEMPSENS_CMD_REQ_DATA, NULL, &tc02_4_failed_readTemp_flag);
            tc02_4_seq++;
            break;
```

```

case 3:
    tc02_4_result_stopProc_flag= tempSensor_reqComm
and(
    TEMPSENS_CMD_STOP_ALL_PROCESS,NULL,&t
c02_4_failed_stopProc_flag);
    tc02_4_seq++;
    break;
case 4:
    tc02_4_test_errCode = tempSensor_getErrCode();
    tc02_4_test_ready = tempSens_isReady_forReq();
    tc02_4_seq++;
    break;
default:
    tc02_4_seq++;
    break;
}
}

```

H= Variables × Breakpoints Expressions Registers Live Expressions SFRs		
Name	Type	Value
X= tc02_4_seq	uint16_t	4
X= tc02_4_result_readTemp_flag	_Bool	true
X= tc02_4_failed_readTemp_flag	_Bool	false
X= tc02_4_result_stopProc_flag	_Bool	true
X= tc02_4_failed_stopProc_flag	_Bool	false
X= tc02_4_test_errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY
X= tc02_4_test_ready	_Bool	true

Expression	Type	Value
vTempSensor_obj	typTempSensor_obj	{...}
scratchpad	uint8_t [9]	[9]
temperCelcius	float	0
sensorResol	uint8_t	0 'W000'
resol8Bit	uint8_t	0 'W000'
scratchpadFilled	_Bool	false
lastData_IsValid_flag	_Bool	true
stopProc	_Bool	false
errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY
mismatch_dataCnt	uint8_t	0 'W000'
currState	typTempSens_st	TEMPSENS_STATE_IDLE
prevState	typTempSens_st	TEMPSENS_STATE_IDLE
OWcurrState	typStatesOW	OW_STATE_RDY_CHK
OWerrCode	typErrCodeOW	OW_ERR_OKAY
initialized	_Bool	true
stateCnt	uint32_t	326
currCmd	typTempSensCMD	{...}
currCode	typTempSens_provCMDcodes	0
vOneWire_obj	typOneWire	{...}
GPIOx	GPIO_TypeDef *	0x40010800
DIO_Pin	uint16_t	4
htimX	TIM_HandleTypeDef *	0x2000021c <htim2>
initialized	_Bool	true
errCode	typErrCodeOW	OW_ERR_OKAY
currState	typStatesOW	OW_STATE_IDLE
prevState	typStatesOW	OW_STATE_RDY_CHK
stateCnt	uint32_t	6520
rcvrFailCount	uint8_t	0 'W000'
rdyChk_Cnt_5ms	uint8_t	54 '6'
rdyChk_TO_5ms	uint8_t	200 'W310'
rdyChk_cmpltFlag	_Bool	false
rdyChk_timeOut_flag	_Bool	false
stopFlag	_Bool	true
cmdQ	typOneWireCMD [30]	[30]
cmdNext	typOneWireCMD	{...}
cmdNum	uint8_t	13 'Wr'

(oneWire의 RDY_CHK 중, stop시그널을 받고 상태가 강제로 천이되는 모습)

(tempSensor 객체는 다음틱에서 곧바로 다음 명령을 받을 수 있게 pending 상태를 벗어나지만, oneWire 객체는 RDY_CHK가 아니면 즉시 상태를 빠져나오지 않고, 기존 작업을 처리한 후 IDLE의 entry 액션에서 플러싱을 함)

▼ 에러 처리 테스트 (BB-TC03)

▼ BB-TC031

목표 : CRC 미스매치 한계 달성시, MALFUNC 에러 발생

절차 :

- tempSensor 정상 초기화된 상태
- 1초 주기로 `tempSensor_reqCommand(REQ_DATA)` 요청

- tempSensor 주기에서 더미 코드로 scratchpad 8 bytes = 0xFF (CRC mismatch 유도)
- 5번째 mismatch 시 errCode 확인

판정 기준 :

- tempSensor_obj.mismatch_dataCnt = 5
- tempSensor_obj.lastData_isValid_flag = false
- tempSensor_obj.errCode == TEMPSSENS_ERR_MALFUNC
- tempSensor_obj.currState == ERROR

테스트 결과 :

▼ 테스트 코드

```
void TempSensor_Dummy_TC03_1(uint8_t* target) // Temp
Sensor.c 의 postProc의 상단에서 호출시킨다
{
    *target = 0xff;
}
```

호출위치

```
static void tempSensor_rawData_postProc(void) // 센서에서 9바이트를 모두 보내오면 필요한 값들을 변환해서 객체에 저장
{
    uint16_t tmp = 0;
    int8_t minus = 0, integer = 0;
    float temperVal = 0;
    uint8_t calculated_CRCVal = tempSensor_CRC8(vTempSensor_obj.scratchpad, 8); // 1. CRC 계산
    uint8_t tmpResol = 0;

    TempSensor_Dummy_TC03_1(&vTempSensor_obj.scratchpad[TEMPSSENS_CRC_POS]); // todo : for test

    if (calculated_CRCVal != vTempSensor_obj.scratchpad[TEMPSSENS_CRC_POS]) // 2. CRC 체크 -> 카운트 증가
    {
        vTempSensor_obj.lastData_IsValid_flag = false;
        vTempSensor_obj.mismatch_dataCnt++;

        if (vTempSensor_obj.mismatch_dataCnt >= TEMPSSENS_DATA_INVALID_CNT_THRSLD) // 3. 연속으로 5번 미스매치시 오류로 처리
        {
            vTempSensor_obj.errCode = TEMPSSENS_ERR_MALFUNC;
        }
    }
    else
    {

```

Expression	Type	Value	Address
vTempSensor_obj	typTempSensor_obj	{...}	0x200001a4
> scratchpad	uint8_t [9]	[9]	0x200001a4
temperCelcius	float	0	0x200001ad
sensorResol	uint8_t	0 'W000'	0x200001b1
resol8Bit	uint8_t	0 'W000'	0x200001b2
scratchpadFilled	_Bool	true	0x200001b3
lastData_IsValid_flag	_Bool	false	0x200001b4
stopProc	_Bool	false	0x200001b5
errCode	typTempSens_errCode	TEMPSENS_ERR_MALFUNC	0x200001b6
mismatch_dataCnt	uint8_t	5 'W005'	0x200001b7
currState	typTempSens_st	TEMPSENS_STATE_IDLE	0x200001b8
prevState	typTempSens_st	TEMPSENS_STATE_IDLE	0x200001b9
OWcurrState	typStatesOW	OW_STATE_IDLE	0x200001ba
OWerrCode	typErrCodeOW	OW_ERR_OKAY	0x200001bb
initialized	_Bool	true	0x200001bc
stateCnt	uint32_t	50	0x200001bd
> currCmd	typTempSens_CMD	{...}	0x200001c1
currCode	typTempSens_provCMDcodes	0	0x200001ca

▼ BB-TC032

목표 : oneWire timeout 에러 확인

절차 :

- RDY_CHK oneWire에게 전달하는 대기시간을 오류가 발생하도록 하드 코딩
 - oneWire에 전달되는 정상적인 대기시간 파라미터 값 대신 매우 짧은 시간으로 변경해서 타임아웃 유도 → 5 tick (25ms)
- tempSensor 정상 초기화된 상태
- tempSensor_reqCommand(REQ_DATA) 요청
- EXEC → PENDING 상태에서 timeout발생
- 다음 tick의 상태를 확인

판정 기준 :

- tempSensor_obj.currState == ERROR
- tempSensor_obj.errCode == TEMPSSENS_ERR_TIMEOUT
- oneWire_obj.currState = ERROR
- oneWire_obj.errCode = OW_ERR_TIMEOUT
- oneWire_obj.rdyChk_TO_5ms = 5
- oneWire_obj.rdyChk_Cnt_5ms = 7

테스트 결과 :

▼ 테스트 코드

```

void TempSensor_Dummy_TC03_2(uint8_t** param) // O
W.c 의 rdy_chk 상태의 entry 액션에서 호출시킨다
{
    static uint8_t dummy_5msTick = 5;

    *param = &dummy_5msTick;
}

```

```

545     case OW_STATE_RDY_CHK:
546
547         TempSensor_Dummy_TC03_2(&vOneWire_obj.cmdNext.param); // todo : for test
548
549         vOneWire_obj.rdyChk_Cnt_5ms = 0;
550
551         if (vOneWire_obj.cmdNext.param == NULL) // 처음이라 센서에 얼마의 분해능이 저장되었는지 모름
552         {
553             vOneWire_obj.rdyChk_TO_5ms = OW_1SEC_TO_500MS_TICK;
554         }
555         else // 분해능이 얼마인지 알고있음 -> 계산가능
556         {
557             vOneWire_obj.rdyChk_TO_5ms = *(vOneWire_obj.cmdNext.param);
558         }
559
560         vOneWire_obj.rdyChk_timeOut_flag = false;
561         vOneWire_obj.rdyChk_cmpltFlag = false;
562         break;

```

(OW.c의 entryAction 함수 내부)

vOneWire_obj	typOneWire	{...}	0x200000f4
> GPIOx	GPIO_TypeDef *	0x40010800	0x200000f4
> DIO_Pin	uint16_t	4	0x200000f8
> htimX	TIM_HandleTypeDef *	0x2000021c <htim2>	0x200000fa
> initialized	_Bool	true	0x200000fe
> errCode	typErrCodeOW	OW_ERR_OKAY	0x200000ff
> currState	typStatesOW	OW_STATE_IDLE	0x20000100
> prevState	typStatesOW	OW_STATE_IDLE	0x20000101
> stateCnt	uint32_t	400	0x20000102
> rcvrFailCount	uint8_t	0 'W000'	0x20000106
> rdyChk_Cnt_5ms	uint8_t	5 'W005'	0x20000107
> rdyChk_TO_5ms	uint8_t	5 'W005'	0x20000108
> rdyChk_cmpltFlag	_Bool	true	0x20000109
> rdyChk_timeOut_flag	_Bool	false	0x2000010a
> stopFlag	_Bool	false	0x2000010b
> cmdQ	typOneWireCMD [30]	[30]	0x2000010c
> cmdNext	typOneWireCMD	{...}	0x200001a2
> cmdNum	uint8_t	0 'W0'	0x200001a7
> htim1->Instance.CNT	volatile uint32_t	935	0x40012c24
+ Add new expression			

Expression	Type	Value	Address
vTempSensor_obj	typTempSensor_obj	{...}	0x200001a8
scratchpad	uint8_t [9]	[9]	0x200001a8
temperCelcius	float	0	0x200001b1
sensorResol	uint8_t	0 'W000'	0x200001b5
resol8Bit	uint8_t	0 'W000'	0x200001b6
scratchpadFilled	_Bool	false	0x200001b7
lastData_IsValid_flag	_Bool	true	0x200001b8
stopProc	_Bool	false	0x200001b9
errCode	typTempSens_errCode	TEMPSENS_ERR_TIMEOUT	0x200001ba
mismatch_dataCnt	uint8_t	0 'W000'	0x200001bb
currState	typTempSens_st	TEMPSENS_STATE_ERROR	0x200001bc
prevState	typTempSens_st	TEMPSENS_STATE_ERROR	0x200001bd
OWcurrState	typStatesOW	OW_STATE_ERROR	0x200001be
OWerrCode	typErrCodeOW	OW_ERR_TIMEOUT	0x200001bf
initialized	_Bool	true	0x200001c0
stateCnt	uint32_t	19	0x200001c1
currCmd	typTempSensCMD	{...}	0x200001c5
currCode	typTempSens_provCMDcodes	0	0x200001ce
vOneWire_obj	typOneWire	{...}	0x200000f4

▼ BB-TC033

목표 : 센서 응답없음 상태에서 tempSensor 계층의 상태천이 확인

절차 :

- TempSensor_Config.h 파일에서
`OW_NO_RESPONDING_THRESHOLD_5msTick` 상수를 5 틱으로 변경
- HW적으로 dio 핀을 임시적으로 VCC에 접지
- `oneWire_obj.currState = RDY_CHK` 상태로 천이되는것을 확인
- dio 핀을 GND에 접지시킬 것
- `oneWire_obj.currState = IDLE` 상태로 천이되는것을 확인

판정 기준 :

- `oneWire_obj.rcvrFailCount < 5`
- 다시 선을 접지 시키면 `oneWire_obj.currState = IDLE` 일 것

테스트 결과 :

Expression	Type	Value	Address
vTempSensor_obj	typTempSensor_obj	{...}	0x200001a8
> scratchpad	uint8_t [9]	[9]	0x200001a8
tempCelsius	float	0	0x200001b1
sensorResol	uint8_t	0 'W000'	0x200001b5
resol8Bit	uint8_t	0 'W000'	0x200001b6
scratchpadFilled	_Bool	false	0x200001b7
lastData_IsValid_flag	_Bool	true	0x200001b8
stopProc	_Bool	false	0x200001b9
errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY	0x200001ba
mismatch_dataCnt	uint8_t	0 'W000'	0x200001bb
currState	typTempSens_st	TEMPSENS_STATE_IDLE	0x200001bc
prevState	typTempSens_st	TEMPSENS_STATE_IDLE	0x200001bd
OWcurrState	typStatesOW	OW_STATE_IDLE	0x200001be
OWerrCode	typErrCodeOW	OW_ERR_OKAY	0x200001bf
initialized	_Bool	true	0x200001c0
stateCnt	uint32_t	1	0x200001c1
> currCmd	typTempSensCMD	{...}	0x200001c5
currCode	typTempSens_provCMDcodes	0	0x200001ce
vOneWire_obj	typOneWire	{...}	0x200000f0
> GPIOx	GPIO_TypeDef *	0x40010800	0x200000f0
DIO_Pin	uint16_t	4	0x200000f4
> htimX	TIM_HandleTypeDef *	0x2000021c <-htim2>	0x200000f6
initialized	_Bool	true	0x200000fa
errCode	typErrCodeOW	OW_ERR_NO_RESPONSE	0x200000fb
currState	typStatesOW	OW_STATE_RECOVERY	0x200000fc
prevState	typStatesOW	OW_STATE_RECOVERY	0x200000fd
stateCnt	uint32_t	24	0x200000fe
rcvrFailCount	uint16_t	3	0x20000102
rdyChk_Cnt_5ms	uint8_t	0 'W000'	0x20000104
rdyChk_TO_5ms	uint8_t	0 'W000'	0x20000105
rdyChk_cmpltFlag	_Bool	false	0x20000106
rdyChk_timeOut_flag	_Bool	false	0x20000107
stopFlag	_Bool	false	0x20000108
> cmdQ	typOneWireCMD [30]	[30]	0x20000109
> cmdNext	typOneWireCMD	{...}	0x2000019f
cmdNum	uint8_t	0 'W000'	0x200001a4

(reset 명령을 보내고 VCC에 접지시켜서 3틱동안의 리커버리 기능 수행)

Expression	Type	Value	Address
vTempSensor_obj	typTempSensor_obj	{...}	0x200001a8
> scratchpad	uint8_t [9]	[9]	0x200001a8
tempCelsius	float	0	0x200001b1
sensorResol	uint8_t	0 'W000'	0x200001b5
resol8Bit	uint8_t	0 'W000'	0x200001b6
scratchpadFilled	_Bool	false	0x200001b7
lastData_IsValid_flag	_Bool	true	0x200001b8
stopProc	_Bool	false	0x200001b9
errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY	0x200001ba
mismatch_dataCnt	uint8_t	0 'W000'	0x200001bb
currState	typTempSens_st	TEMPSENS_STATE_EXEC	0x200001bc
prevState	typTempSens_st	TEMPSENS_STATE_EXEC	0x200001bd
OWcurrState	typStatesOW	OW_STATE_IDLE	0x200001be
OWerrCode	typErrCodeOW	OW_ERR_OKAY	0x200001bf
initialized	_Bool	true	0x200001c0
stateCnt	uint32_t	11	0x200001c1
> currCmd	typTempSensCMD	{...}	0x200001c5
currCode	typTempSens_provCMDcodes	0	0x200001ce
vOneWire_obj	typOneWire	{...}	0x200000f0
> GPIOx	GPIO_TypeDef *	0x40010800	0x200000f0
DIO_Pin	uint16_t	4	0x200000f4
> htimX	TIM_HandleTypeDef *	0x2000021c <-htim2>	0x200000f6
initialized	_Bool	true	0x200000fa
errCode	typErrCodeOW	OW_ERR_OKAY	0x200000fb
currState	typStatesOW	OW_STATE_RESET	0x200000fc
prevState	typStatesOW	OW_STATE_IDLE	0x200000fd
stateCnt	uint32_t	219	0x200000fe
rcvrFailCount	uint16_t	4	0x20000102
rdyChk_Cnt_5ms	uint8_t	0 'W000'	0x20000104
rdyChk_TO_5ms	uint8_t	0 'W000'	0x20000105
rdyChk_cmpltFlag	_Bool	false	0x20000106
rdyChk_timeOut_flag	_Bool	false	0x20000107
stopFlag	_Bool	false	0x20000108
> cmdQ	typOneWireCMD [30]	[30]	0x20000109
> cmdNext	typOneWireCMD	{...}	0x2000019f
cmdNum	uint8_t	16 'W020'	0x200001a4
tempSensor_dummy_userResol		Failed to evaluate expression	Failed to evaluate expression
tempSensor_dummy_seq_cnt		Failed to evaluate expression	Failed to evaluate expression
tempSensor_dummy_failed_flag		Failed to evaluate expression	Failed to evaluate expression
+ Add new expression			

(다시 리커버리 상태에서 GND에 접지시켜서 다시 idle 상태로 천이됨을 확인
인)

▼ BB-TC034

목표 : 센서 응답없음 상태에서 tempSensor 계층이 에러 상태로 천이 되는 것
을 확인

절차 :

- TempSensor_Config.h 파일에서
`OW_NO_RESPONDING_THRESHOLD_5msTick` 상수를 5 틱으로 변경
- HW적으로 dio 핀을 임시적으로 VCC에 접지
- 5틱 후 `tempSensor_obj.currState` 확인

판정 기준 :

- `oneWire_obj.rcvrFailCount ≥ 5`
- `oneWire_obj.currState = ERROR`
- `oneWire_obj.errCode = TEMPSSENS_ERR_NOT_RESPONDING`

테스트 결과 :

Expression	Type	Value	Address
vTempSensor_obj	typTempSensor_obj	{...}	0x200001a8
scratchpad	uint8_t [9]	{9}	0x200001a8
temperCelcius	float	0	0x200001b1
sensorResol	uint8_t	0 'W000'	0x200001b5
resol8Bit	uint8_t	0 'W000'	0x200001b6
scratchpadFilled	_Bool	false	0x200001b7
lastData_IsValid_flag	_Bool	true	0x200001b8
stopProc	_Bool	false	0x200001b9
errCode	typTempSens_errCode	TEMPSSENS_ERR_NO_RESPONSE	0x200001ba
mismatch_dataCnt	uint8_t	0 'W000'	0x200001bb
currState	typTempSens_st	TEMPSSENS_STATE_ERROR	0x200001bc
prevState	typTempSens_st	TEMPSSENS_STATE_ERROR	0x200001bd
OWcurrState	typStatesOW	OW_STATE_ERROR	0x200001be
OWerrCode	typErrCodeOW	OW_ERR_NOT_RESPONDING	0x200001bf
initialized	_Bool	true	0x200001c0
stateCnt	uint32_t	61	0x200001c1
currCmd	typTempSensCMD	{...}	0x200001c5
currCode	typTempSens_provCMDcodes	0	0x200001ce
vOneWire_obj	typOneWire	{...}	0x200000f0
GPIOx	GPIO_TypeDef *	0x40010800	0x200000f0
DIO_Pin	uint16_t	4	0x200000f4
htimX	TIM_HandleTypeDef *	0x2000021c <htim2>	0x200000f6
initialized	_Bool	true	0x200000fa
errCode	typErrCodeOW	OW_ERR_NOT_RESPONDING	0x200000fb
currState	typStatesOW	OW_STATE_ERROR	0x200000fc
prevState	typStatesOW	OW_STATE_ERROR	0x200000fd
stateCnt	uint32_t	1219	0x200000fe
rcvrFailCount	uint16_t	5	0x20000102
rdyChk_Cnt_5ms	uint8_t	0 'W000'	0x20000104
rdyChk_To_5ms	uint8_t	0 'W000'	0x20000105
rdyChk_cmpltFlag	_Bool	false	0x20000106
rdyChk_timeOut_flag	_Bool	false	0x20000107
stopFlag	_Bool	false	0x20000108
cmdQ	typOneWireCMD [30]	[30]	0x20000109
cmdNext	typOneWireCMD	{...}	0x2000019f
cmdNum	uint8_t	0 'W000'	0x200001a4
tempSensor_dummy_userResol		Failed to evaluate expression	Failed to evaluate expression
tempSensor_dummy_seq_cnt		Failed to evaluate expression	Failed to evaluate expression
tempSensor_dummy_failed_flag		Failed to evaluate expression	Failed to evaluate expression

▼ 통합 테스트 (BB-TC04)

▼ BB-TC041

목표 : 주기적으로 호출되어야 하는 FND모듈과의 연계를 통해 FND가 끊임없이 온도를 표시하는지 확인

절차 :

- 1ms의 task에서 FND 모듈 호출
- 1s의 task에서 REQ_DATA
- 중간중간 온도센서에 열을 가해 변하는 것 확인
- 15분간 유지

판정 기준 :

- tempSensor는 ERROR 상태를 표시하지 않을 것
- FND 모듈이 단 한번이라도 한 칸만 출력되지 않을 것

테스트 결과 :



vTempSensor_obj	typTempSensor_obj	{...}	
> scratchpad	uint8_t [9]	[9]	
tempCelcius	float	19.875	
sensorResol	uint8_t	11 '#v'	
resol8Bit	uint8_t	0 '#000'	
scratchpadFilled	_Bool	true	
lastData_IsValid_flag	_Bool	true	
stopProc	_Bool	false	
errCode	typTempSens_errCode	TEMPSENS_ERR_OKAY	
mismatch_dataCnt	uint8_t	0 '#000'	
currState	typTempSens_st	TEMPSENS_STATE_PENDING	
prevState	typTempSens_st	TEMPSENS_STATE_PENDING	
OWcurrState	typStatesOW	OW_STATE_IDLE	
OWerrCode	typErrCodeOW	OW_ERR_OKAY	
initialized	_Bool	true	
stateCnt	uint32_t	9098	
> currCmd	typTempSensCMD	{...}	
currCode	typTempSens_provCMDcodes	0	
vOneWire_obj	typOneWire	{...}	
> GPIOx	GPIO_TypeDef *	0x40010800	
DIO_Pin	uint16_t	4	
> htimX	TIM_HandleTypeDef *	0x20000294 <htim2>	
initialized	_Bool	true	
errCode	typErrCodeOW	OW_ERR_OKAY	
currState	typStatesOW	OW_STATE_IDLE	
prevState	typStatesOW	OW_STATE_IDLE	
stateCnt	uint32_t	181969	
rcvrFailCount	uint16_t	0	
rdyChk_Cnt_5ms	uint8_t	102 'f'	
rdyChk_TO_5ms	uint8_t	200 '#310'	
rdyChk_cmpltFlag	_Bool	true	
rdyChk_timeOut_flag	_Bool	false	
stopFlag	_Bool	false	
> cmdQ	typOneWireCMD [30]	[30]	
> cmdNext	typOneWireCMD	{...}	
cmdNum	uint8_t	0 '#0'	

TaskSch_execClock	typExecTimer	{...}	0x2
day	uint8_t	0 '#000'	0x2
hour	uint8_t	0 '#000'	0x2
min	uint8_t	15 '#017'	0x2
sec	uint8_t	3 '#003'	0x2
mili_sec	uint16_t	301	0x2

▼ BB-TC042

목표 : 온도센서 드라이버가 스케줄러에 딜레이를 발생시키는지 확인

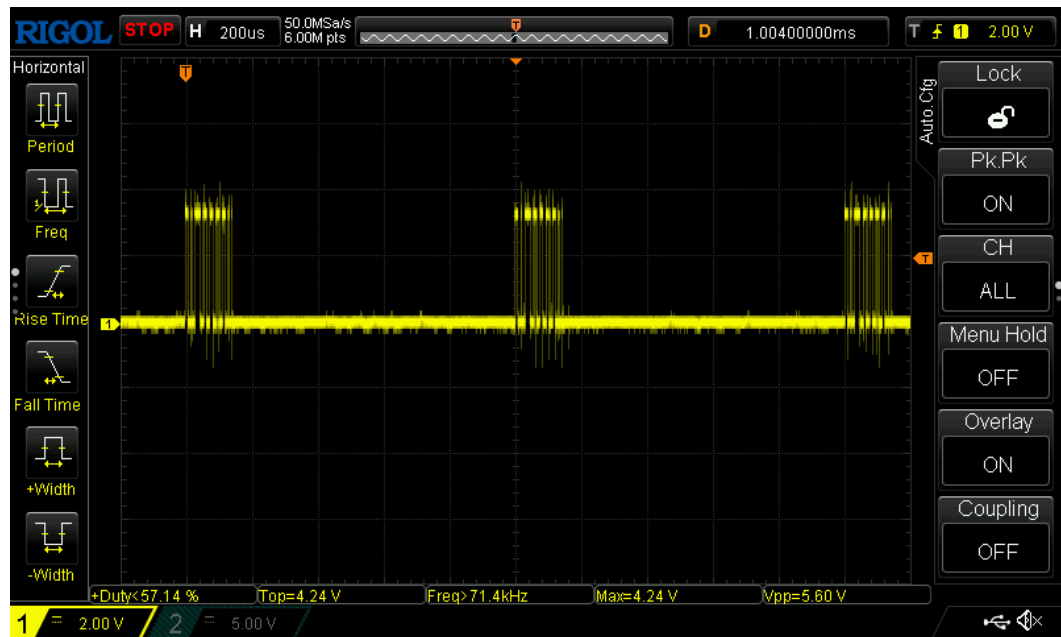
절차 :

- 1ms의 task에서 FND 모듈 호출
- 1s의 task에서 REQ_DATA
- 오실로스코프로 FND DIO단을 찍기

판정 기준 :

- FND 모듈의 통신 주기가 정확히 1ms를 유지할 것

테스트 결과 :



가로 한칸에 200us 이기 때문에 1ms Task로 온도정보가 올바르게 출력되는 것을 확인가능