

스타트-업 가이드 NS-RX231

e²studio 실행

목차

1 프로젝트 생성.....	2
2 소스 변경.....	15
2.1 레지스터.....	15
2.2 소스코드.....	17
3 프로젝트 라이팅.....	19
4 프로젝트 실행.....	22
5 Workbench6 튜닝.....	23

1 프로젝트 생성

CTSU(Capacitive Touch Sensor Unit) API를 이용하여 간단한 프로젝트를 만들어 주는 Workbench6의 First Step Guide기능으로 소스를 만들어 보겠습니다.

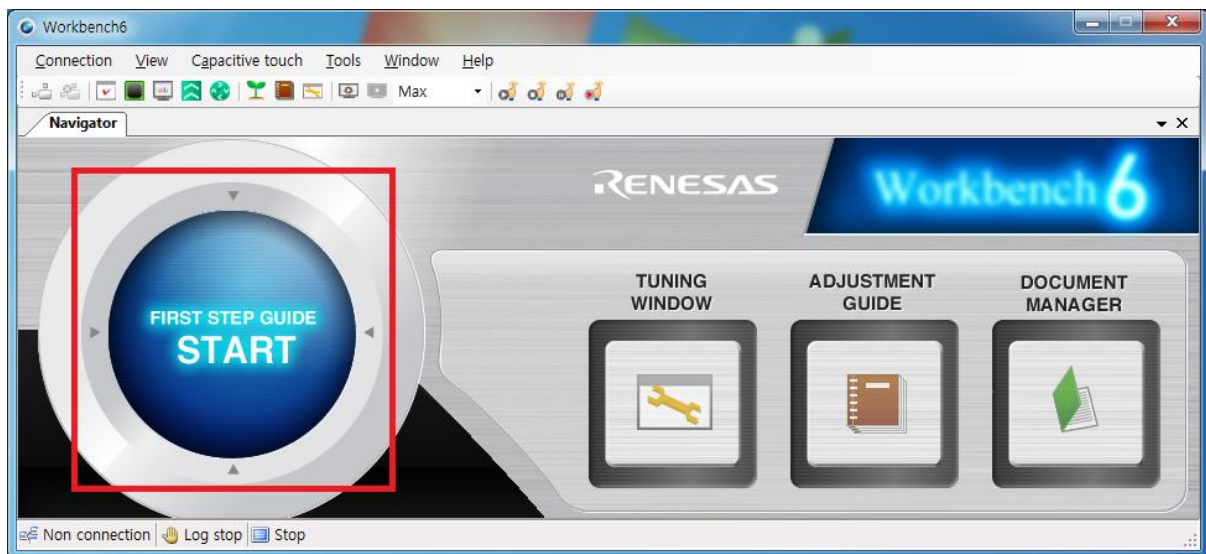
본문의 앞서, Workbench6의 기능을 다룬 영상이 있으니 참고 바랍니다.

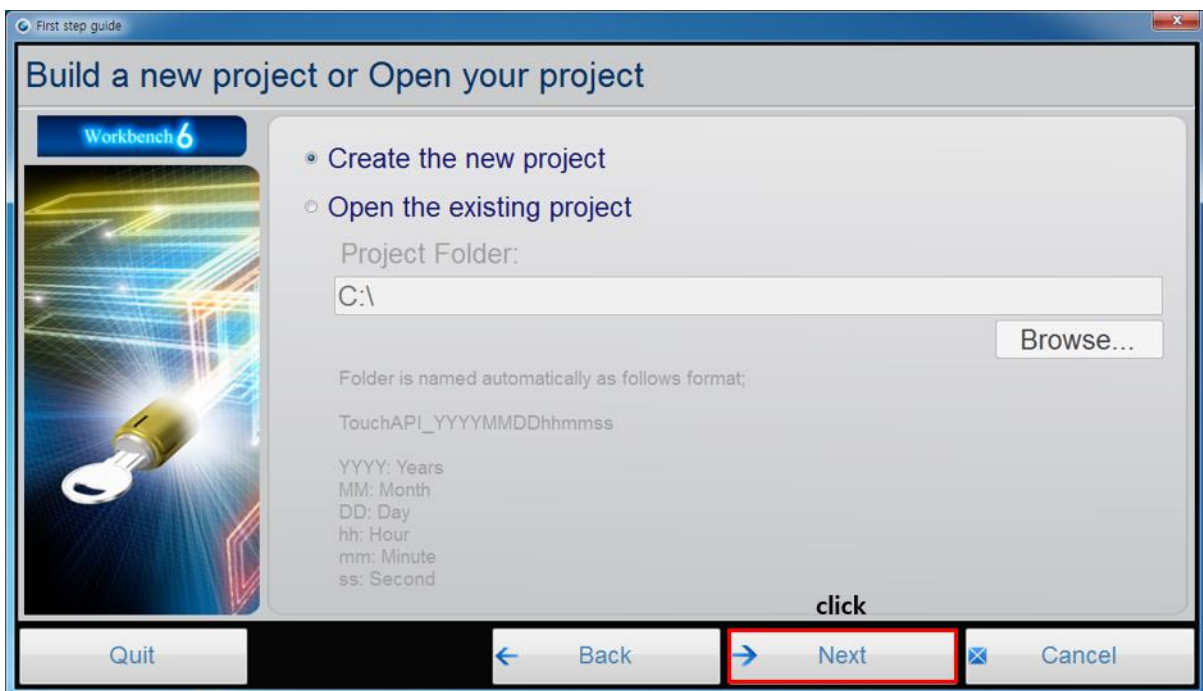
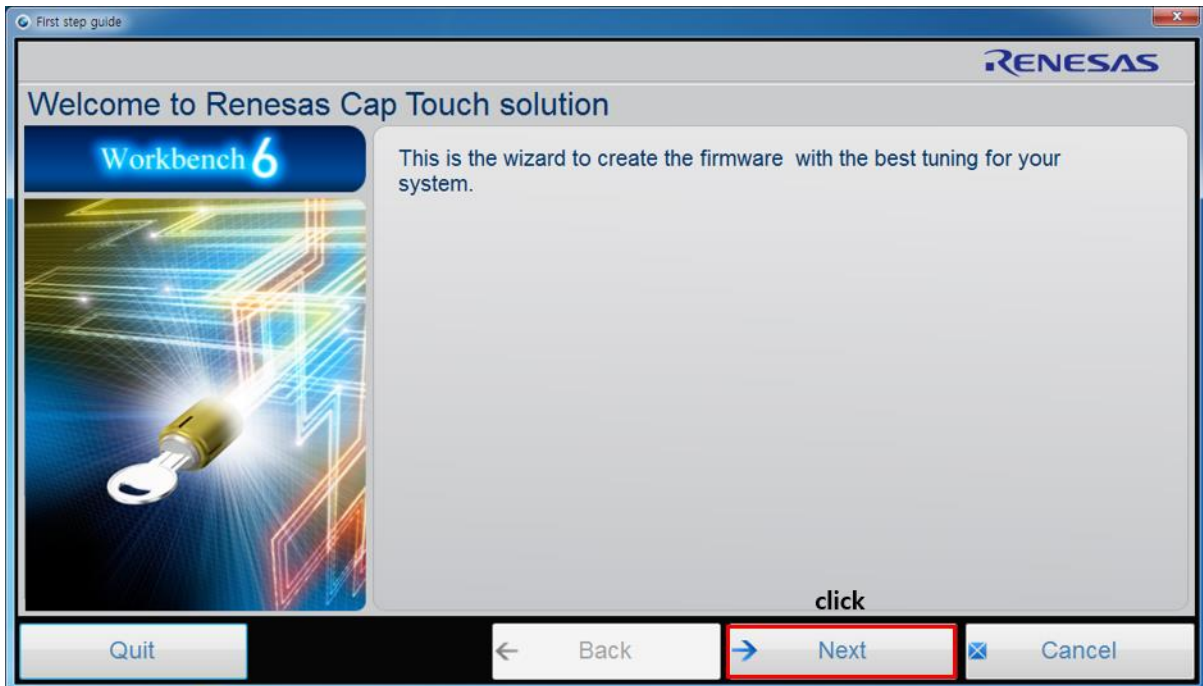
Integrated Development Environment for Renesas Capacitive Touch Workbench6(ver1.03)

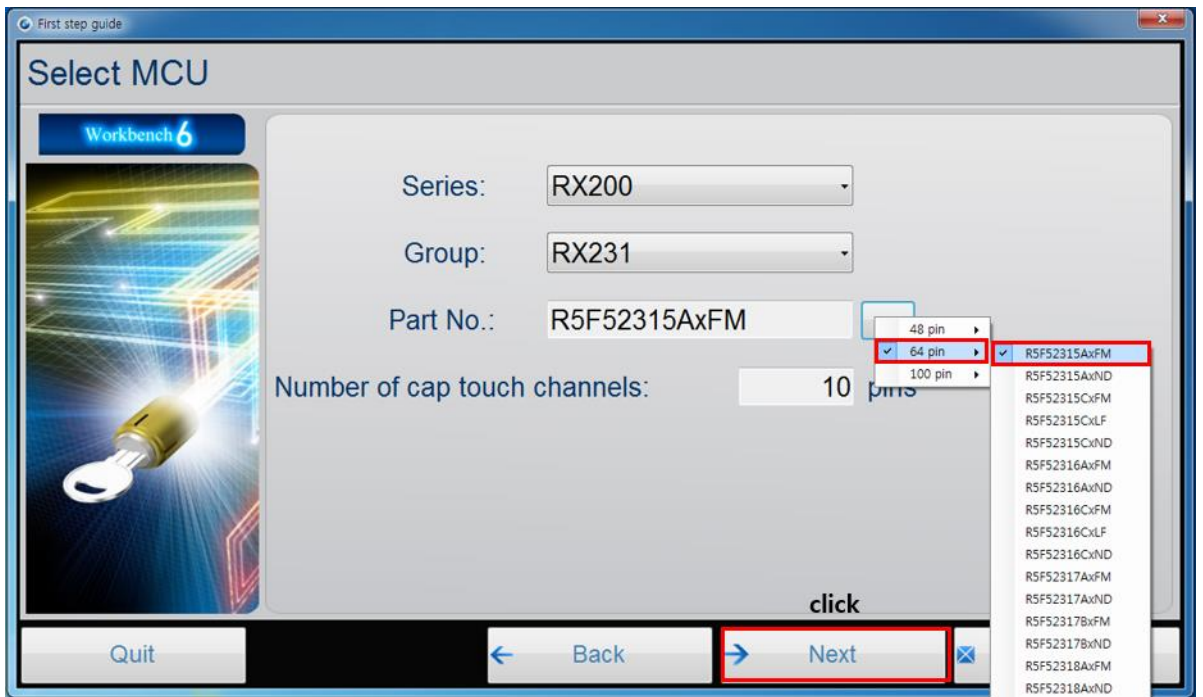
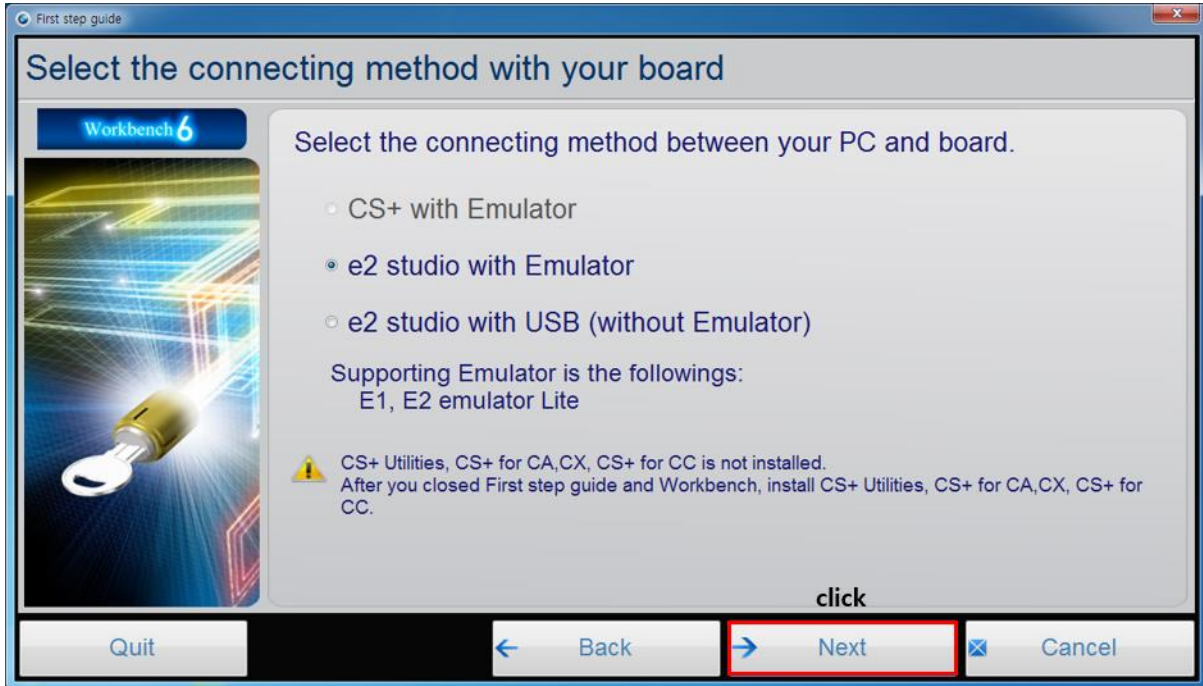
<https://www.youtube.com/watch?v=8oq5iyGyaMw> ①

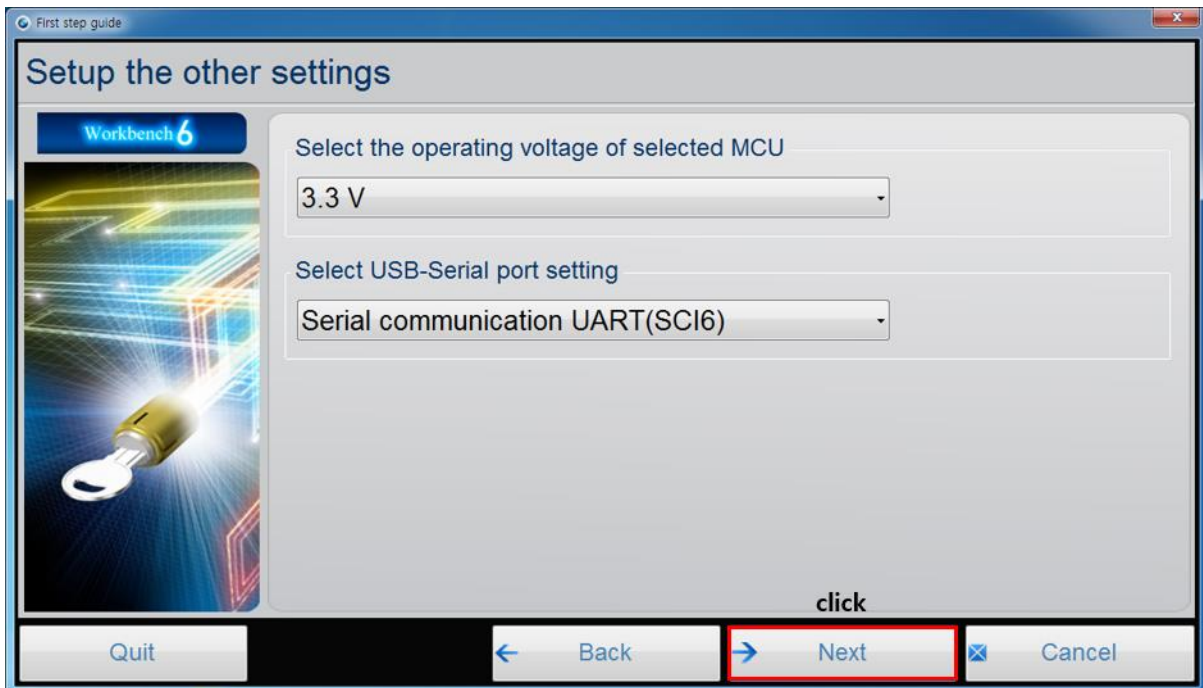
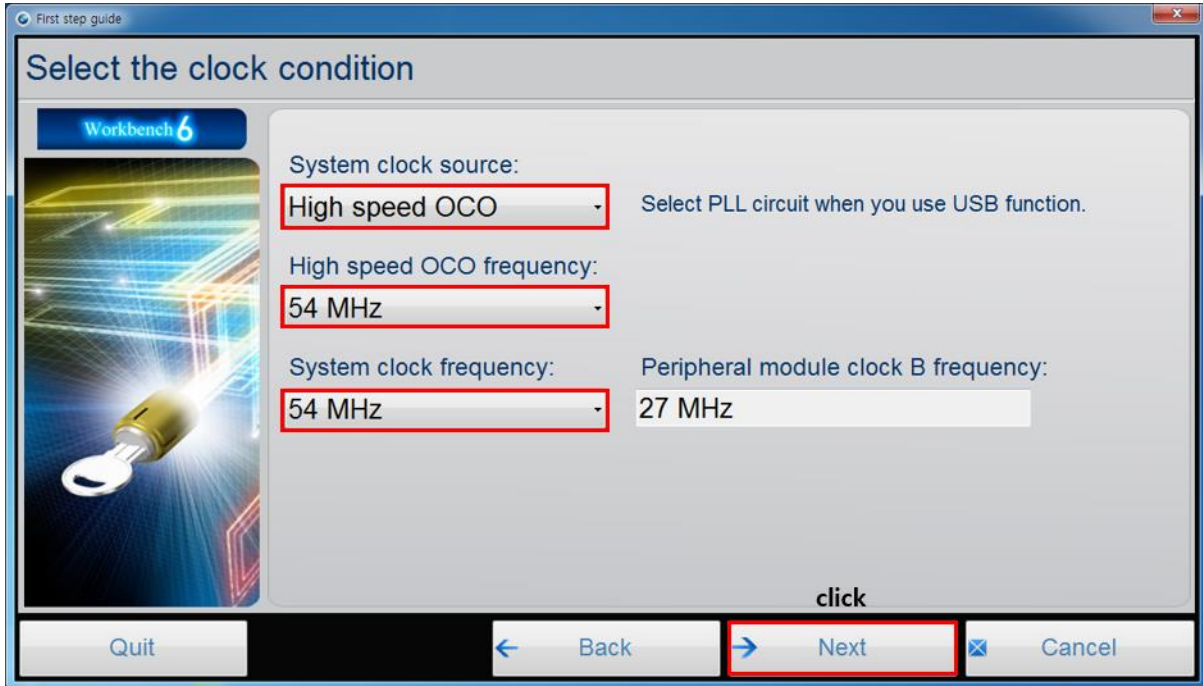
<https://www.youtube.com/watch?v=tdIpiyTrKVM> ②

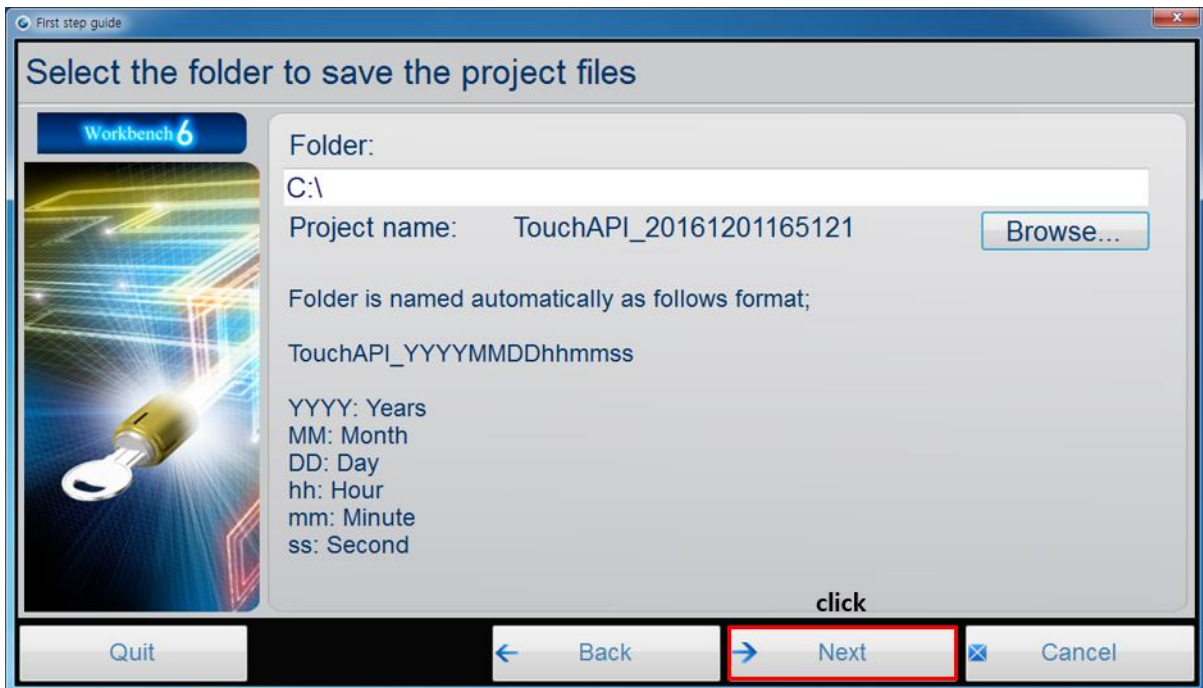
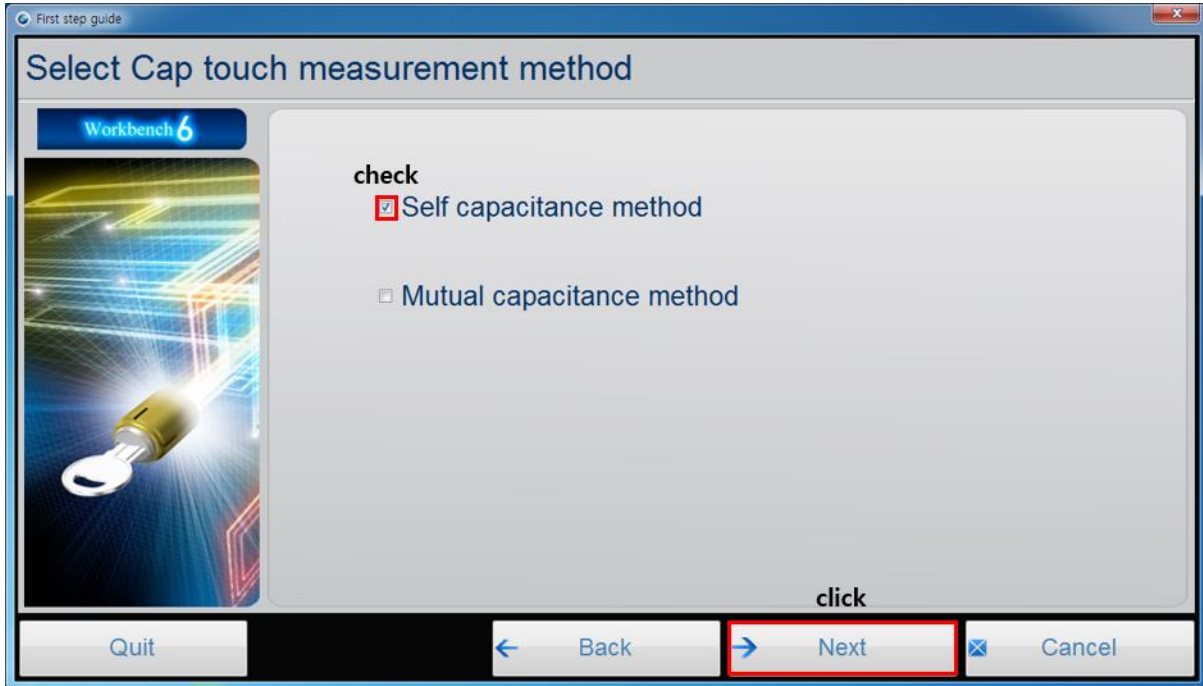
<https://www.youtube.com/watch?v=2nQ9OcP5Cgg> ③

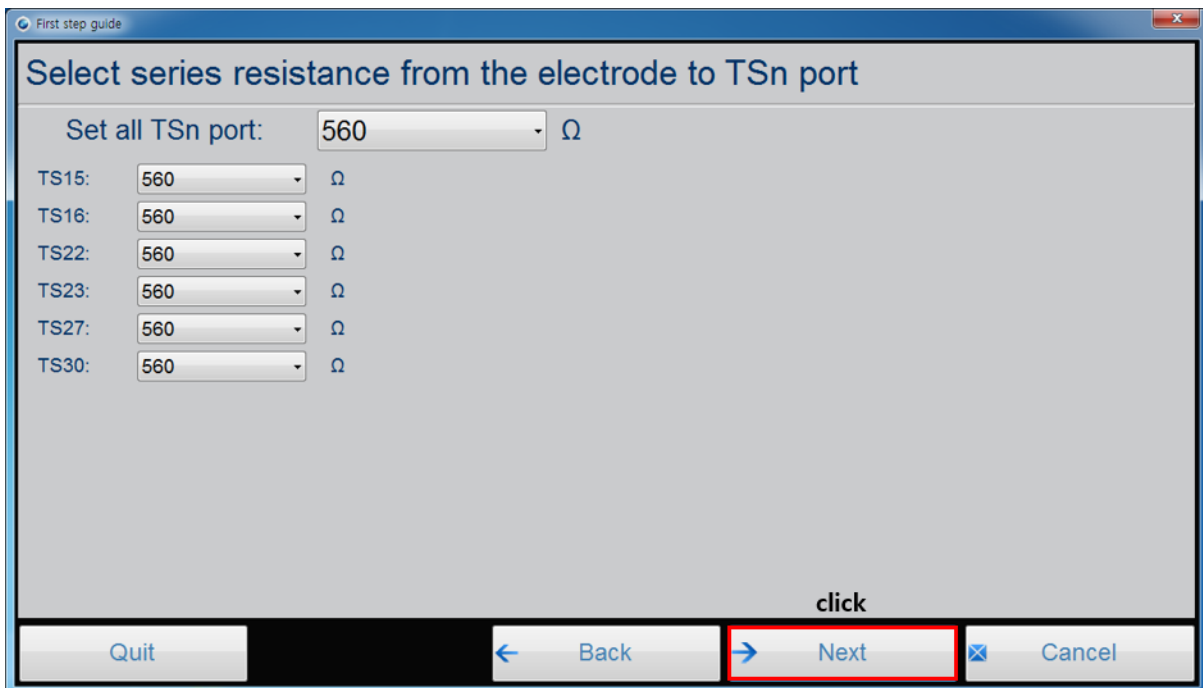
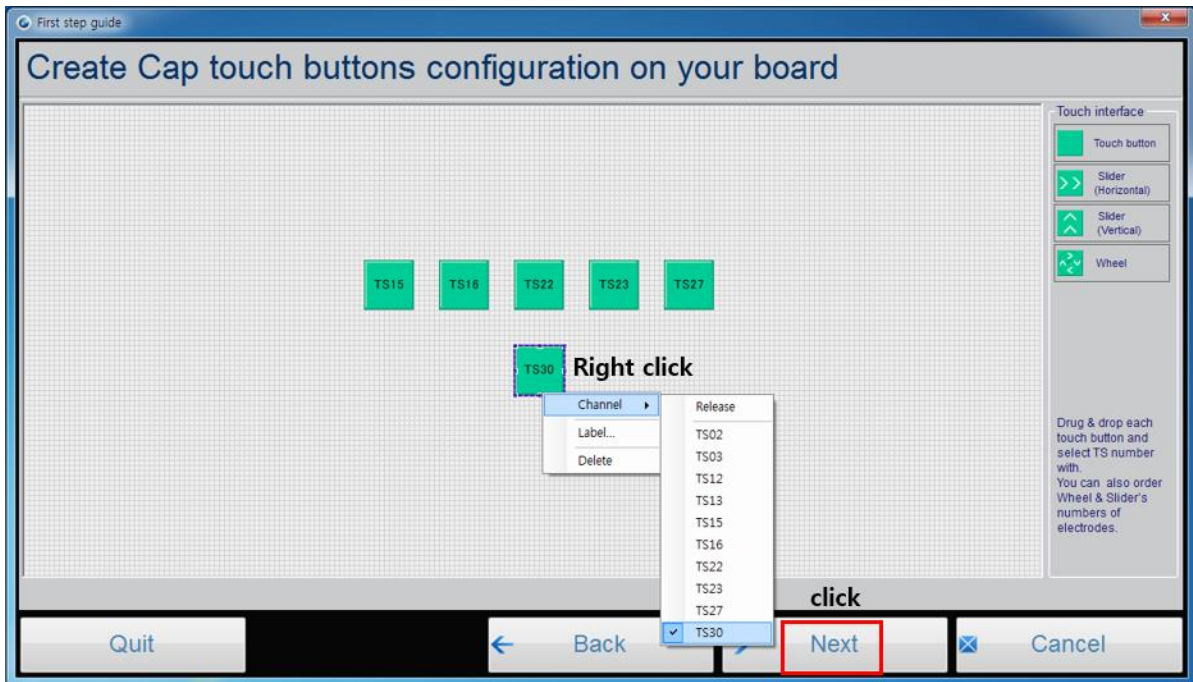


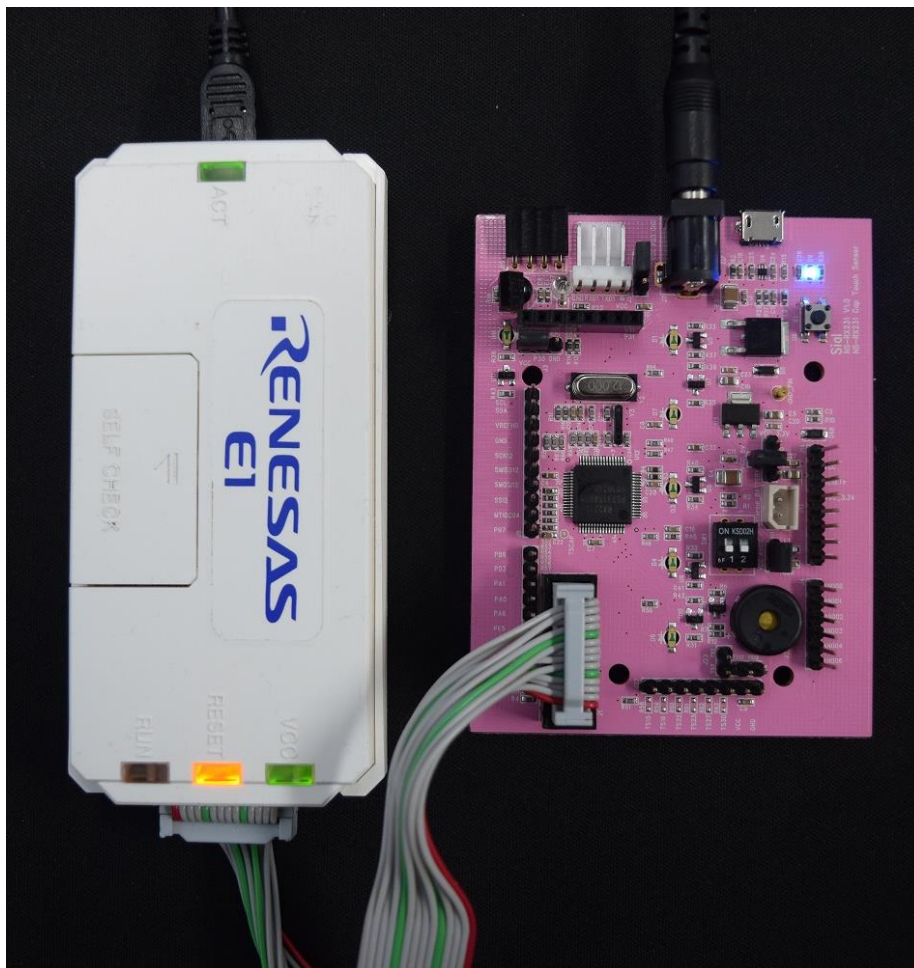
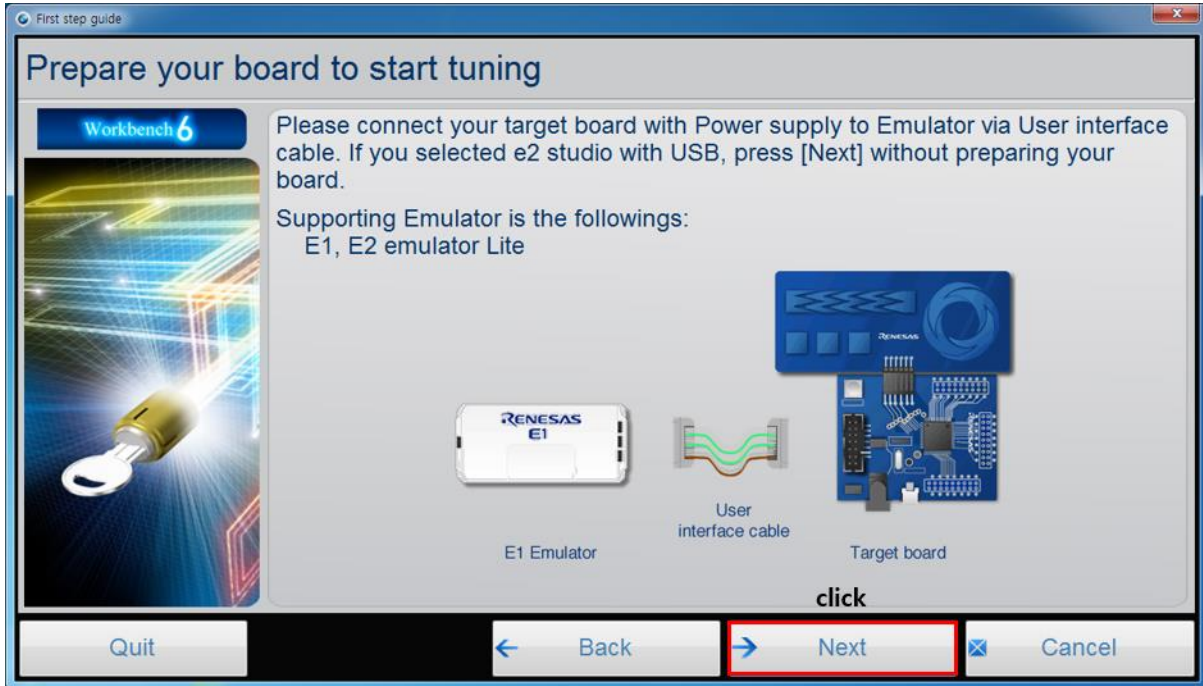


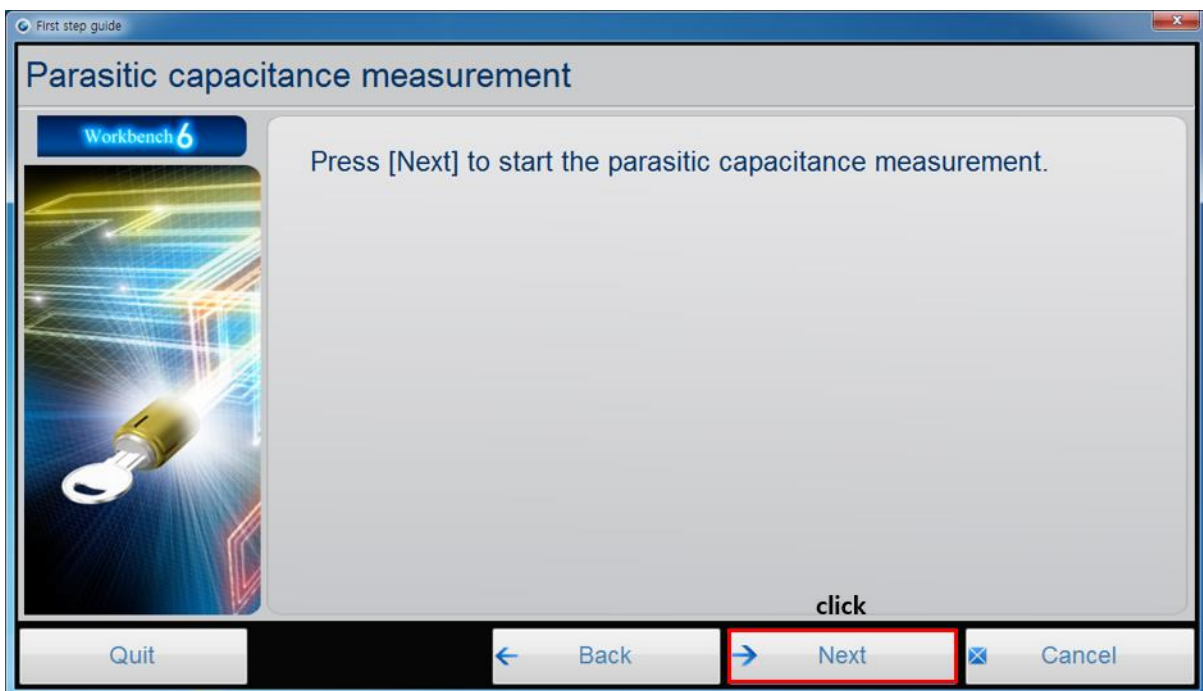
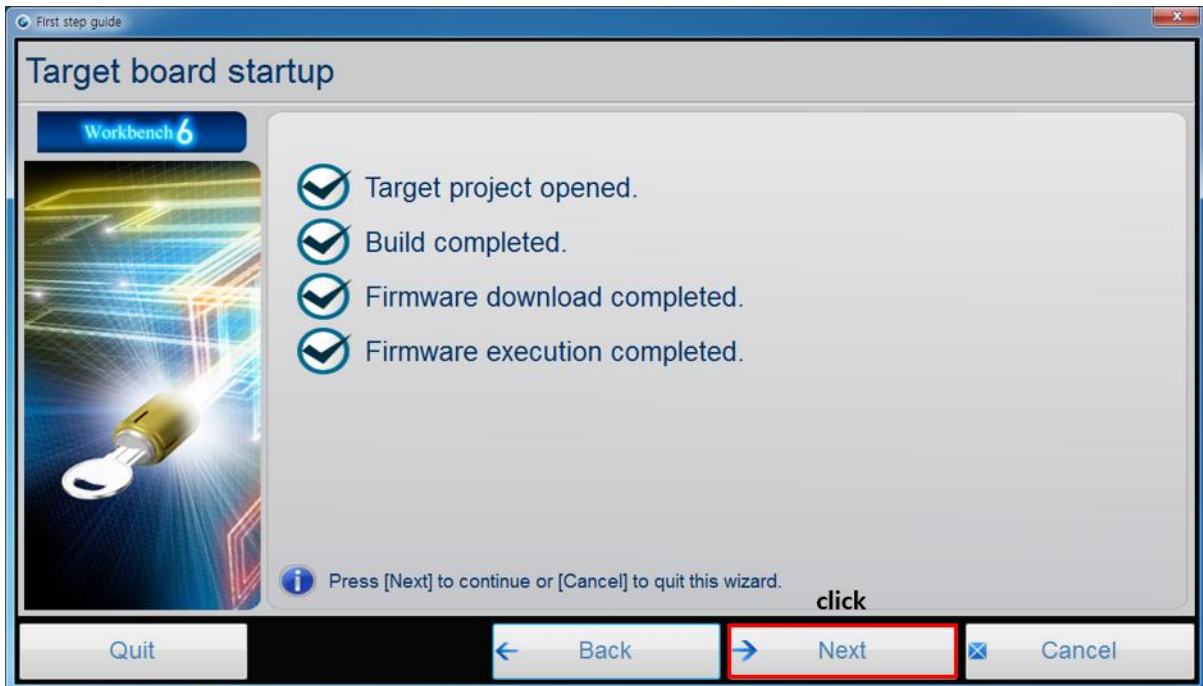


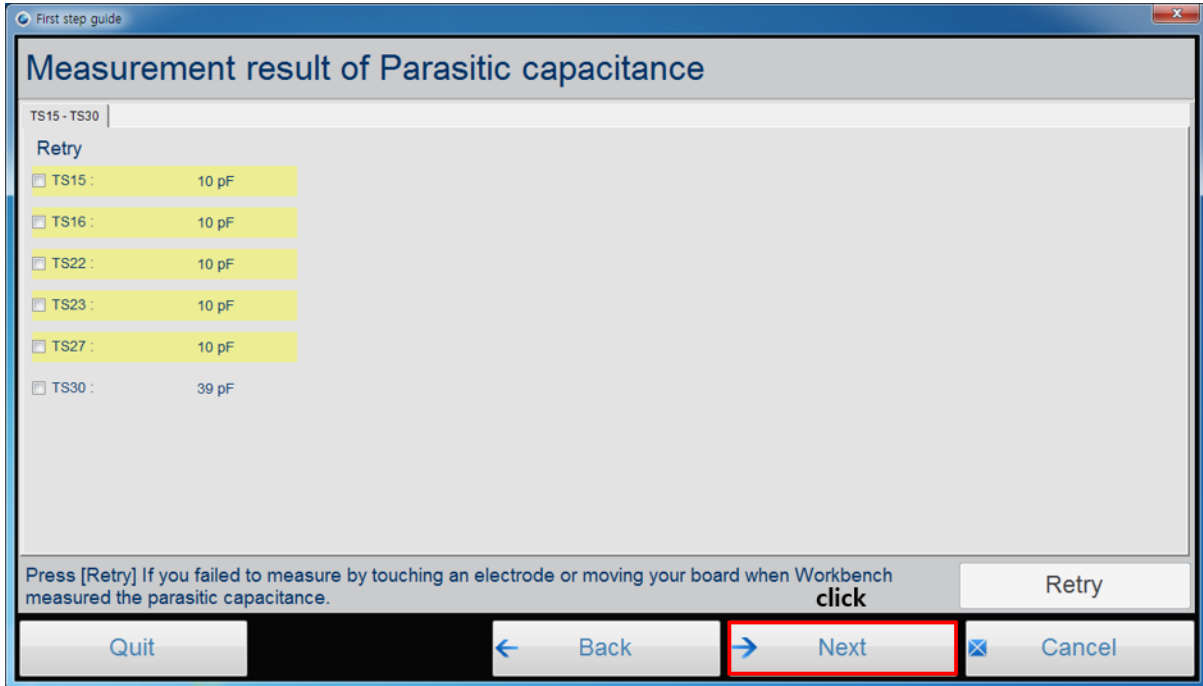




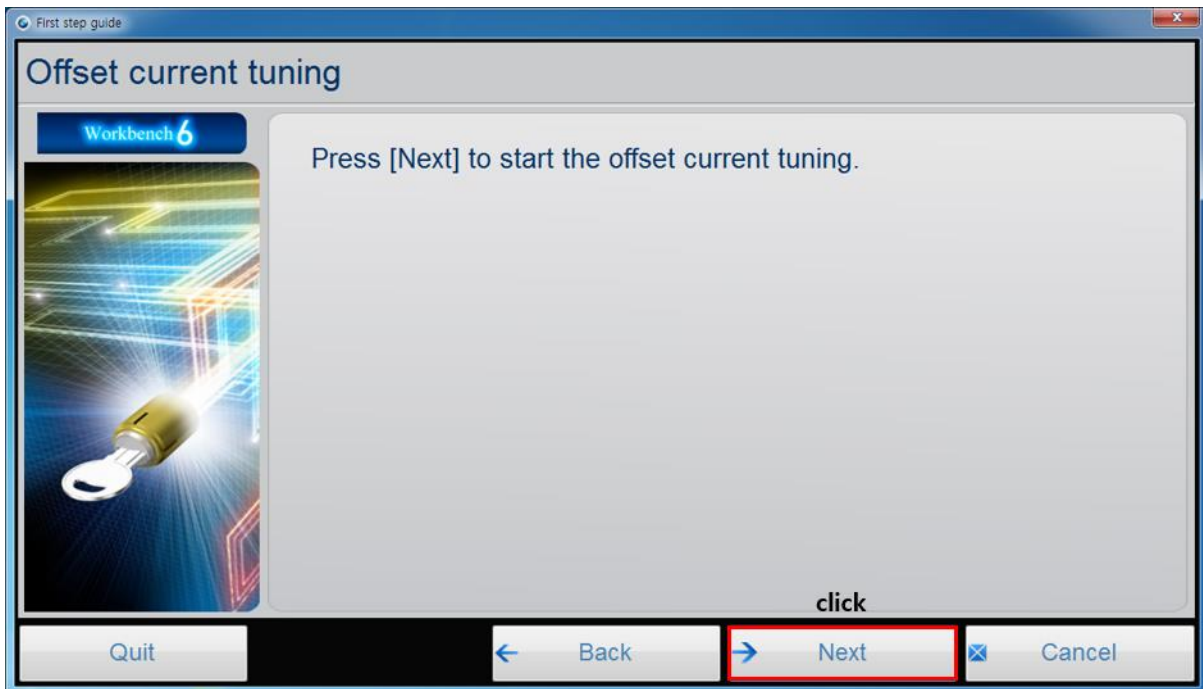


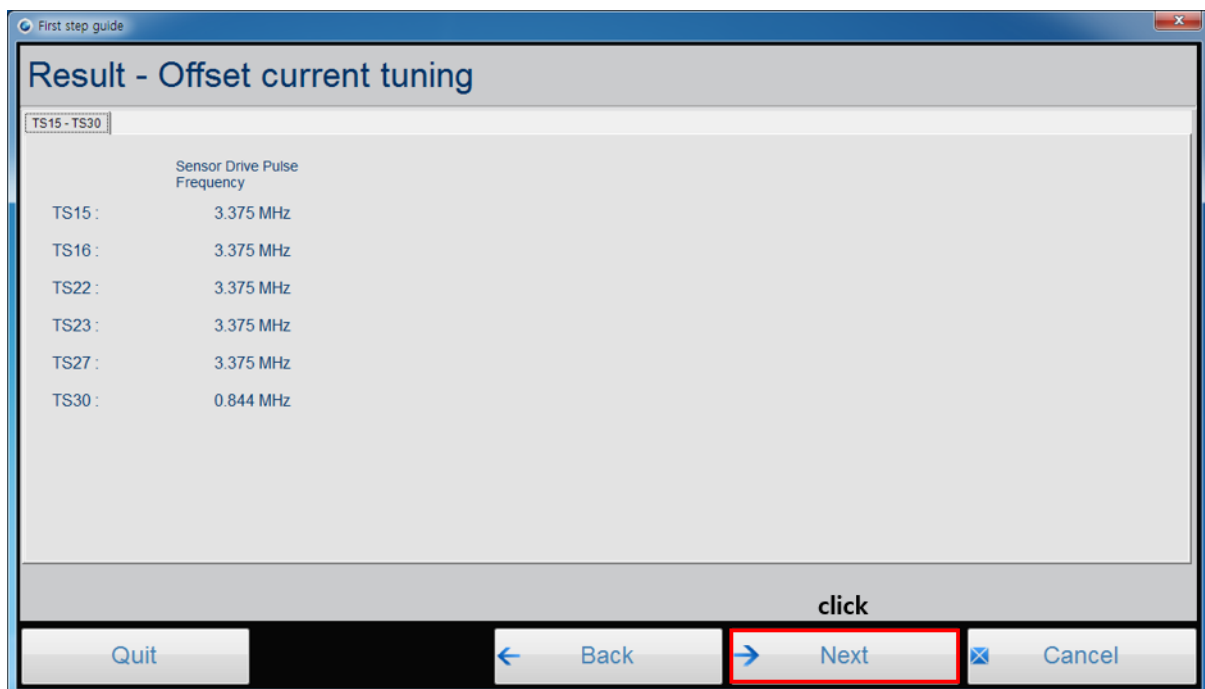
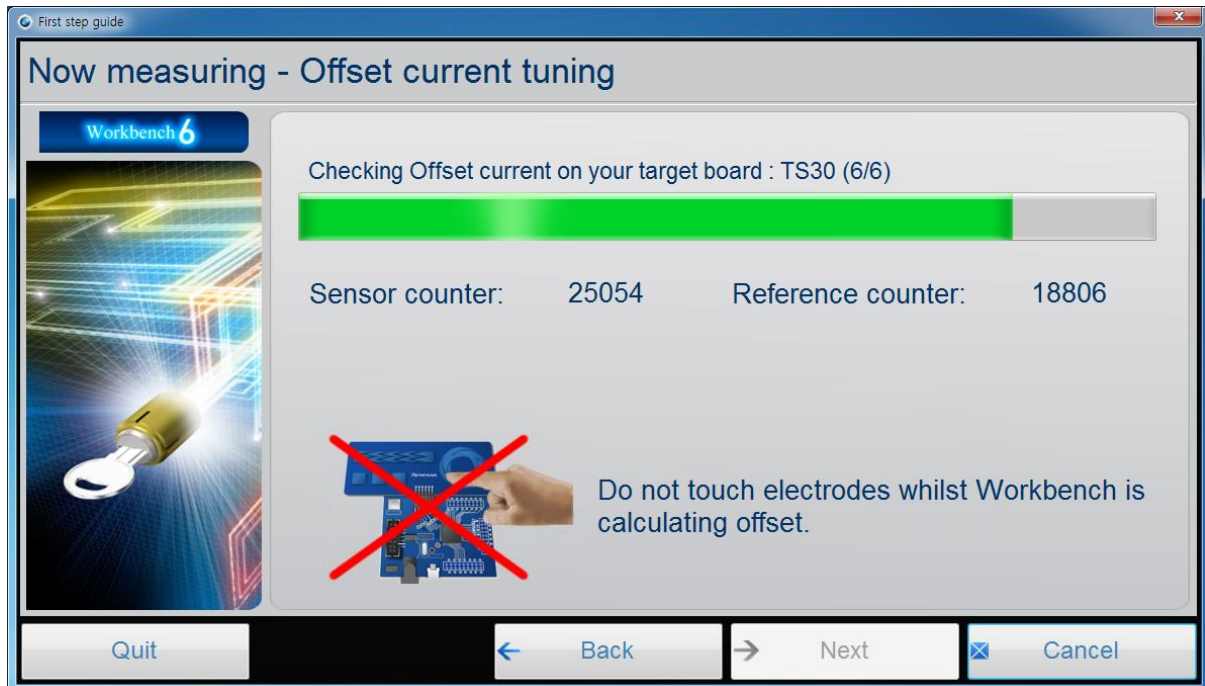




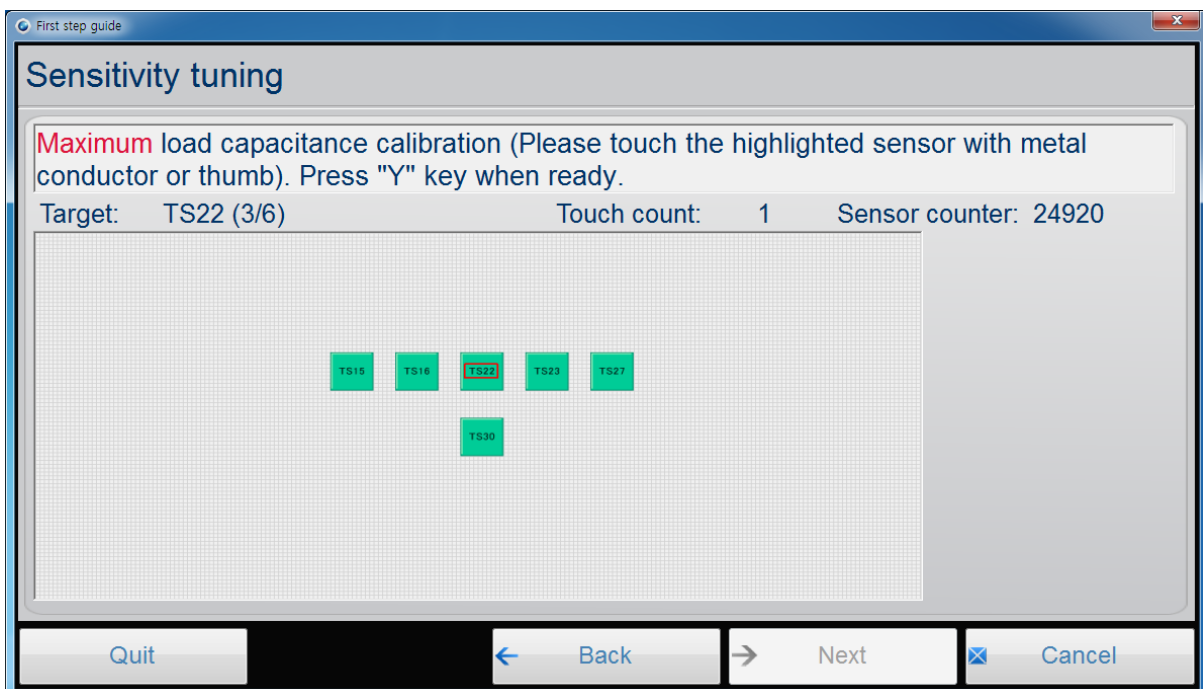
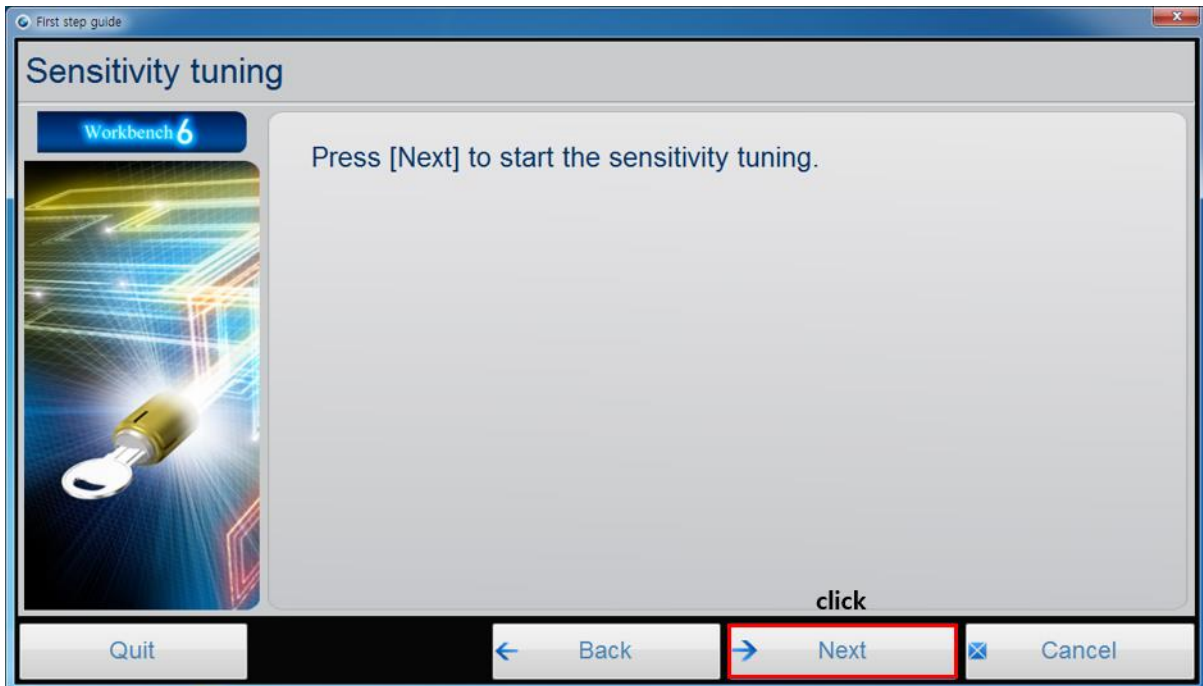


이 값을 다를 수 있습니다





이 값을 다를 수 있습니다



First step guide

Sensitivity tuning

Normal load capacitance calibration (Please touch the highlighted sensor with your finger).
Press "Y" key when ready.

Target: TS22 (3/6) Touch count: 3 Sensor counter: 23253

Quit Back Next Cancel

First step guide

Tuning result of sensitivity

Sensor counter value(without the sum number of the measurement)

TS15 - TS30

- Capacitance change between normal touch and max touch
- Capacitance change between no touch and normal touch
- Capacitance value without touch
- The touch/no touch finger threshold level

Offset TS15 TS16 TS22 TS23 TS27 TS30

Retry

TS15 TS22 TS27
 TS16 TS23 TS30


Check your board or retry tuning process when check box is ON because the tuning might fail. Press [Next] to **click** code.

Quit Back Next Cancel


First step guide

Target board startup

Workbench 6



- Target project opened.
- Build completed.
- Firmware download completed.
- Firmware execution completed.

 Press [Finish] to quit this wizard.

Quit ← Back → Finish ✕ Cancel

2 소스 변경

2.1 레지스터

먼저 LED제어를 위해 레지스터부터 보자. 문서는 RX231 User's Manual:Hardware를 참고했습니다.

21.3.1 Port Direction Register (PDR)

Address(es): PORT0.PDR 0008 C000h, PORT1.PDR 0008 C001h, PORT2.PDR 0008 C002h, PORT3.PDR 0008 C003h, PORT4.PDR 0008 C004h, PORT5.PDR 0008 C005h, PORTA.PDR 0008 C00Ah, PORTB.PDR 0008 C00Bh, PORTC.PDR 0008 C00Ch, PORTD.PDR 0008 C00Dh, PORTE.PDR 0008 C00Eh, PORTH.PDR 0008 C011h, PORTJ.PDR 0008 C012h

b7	b6	b5	b4	b3	b2	b1	b0
B7	B6	B5	B4	B3	B2	B1	B0

Value after reset: 0 0 0 0 0 0 0 0

Bit	Symbol	Bit Name	Description	R/W
b0	B0	Pm0 I/O Select	0: Input (Functions as an input pin.) 1: Output (Functions as an output pin.)	R/W
b1	B1	Pm1 I/O Select		R/W
b2	B2	Pm2 I/O Select		R/W
b3	B3	Pm3 I/O Select		R/W
b4	B4	Pm4 I/O Select		R/W
b5	B5	Pm5 I/O Select		R/W
b6	B6	Pm6 I/O Select		R/W
b7	B7	Pm7 I/O Select		R/W

m = 0 to 5, A to E, H, J

Port Direction Register(PDR)는 포트 핀의 입출력의 설정을 담당합니다. 기본값은 0이며, 0으로 설정하면 입력, 1로 설정하면 출력으로 설정되게 됩니다.

21.3.2 Port Output Data Register (PODR)

Address(es): PORT0.PODR 0008 C020h, PORT1.PODR 0008 C021h, PORT2.PODR 0008 C022h, PORT3.PODR 0008 C023h, PORT4.PODR 0008 C024h, PORT5.PODR 0008 C025h, PORTA.PODR 0008 C02Ah, PORTB.PODR 0008 C02Bh, PORTC.PODR 0008 C02Ch, PORTD.PODR 0008 C02Dh, PORTE.PODR 0008 C02Eh, PORTH.PODR 0008 C031h, PORTJ.PODR 0008 C032h

b7	b6	b5	b4	b3	b2	b1	b0
B7	B6	B5	B4	B3	B2	B1	B0

Value after reset: 0 0 0 0 0 0 0 0

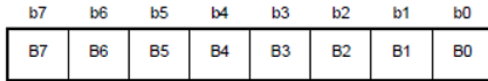
Bit	Symbol	Bit Name	Description	R/W
b0	B0	Pm0 Output Data Store	Holds output data.	R/W
b1	B1	Pm1 Output Data Store		R/W
b2	B2	Pm2 Output Data Store		R/W
b3	B3	Pm3 Output Data Store		R/W
b4	B4	Pm4 Output Data Store		R/W
b5	B5	Pm5 Output Data Store		R/W
b6	B6	Pm6 Output Data Store		R/W
b7	B7	Pm7 Output Data Store		R/W

m = 0 to 5, A to E, H, J

Port Output Data Register(PODR)는 포트 핀의 출력 데이터를 담을 수 있는 레지스터입니다. PMR, PDR레지스터의 설정이 맞게 되어있다면, PODR이 1일 때에는 계속 HIGH의 출력을 내게 됩니다.

21.3.3 Port Input Data Register (PIDR)

Address(es): PORT0.PIDR 0008 C040h, PORT1.PIDR 0008 C041h, PORT2.PIDR 0008 C042h, PORT3.PIDR 0008 C043h, PORT4.PIDR 0008 C044h, PORT5.PIDR 0008 C045h, PORTA.PIDR 0008 C04Ah, PORTB.PIDR 0008 C04Bh, PORTC.PIDR 0008 C04Ch, PORTD.PIDR 0008 C04Dh, PORTE.PIDR 0008 C04Eh, PORTH.PIDR 0008 C051h, PORTJ.PIDR 0008 C052h



Value after reset: x x x x x x x x

x: Undefined

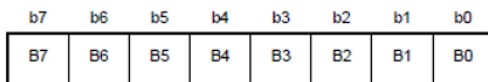
Bit	Symbol	Bit Name	Description	R/W
b0	B0	Pm0	Indicates individual pin states of the corresponding port.	R
b1	B1	Pm1		R
b2	B2	Pm2		R
b3	B3	Pm3		R
b4	B4	Pm4		R
b5	B5	Pm5		R
b6	B6	Pm6		R
b7	B7	Pm7		R

m = 0 to 5, A to E, H, J

Port Input Data Register(PIDR)는 포트 핀의 입력 데이터가 들어오는 레지스터입니다. 읽기만 가능하고 쓰기만 가능한 레지스터이고, PDR, PMR레지스터의 설정과는 상관없이 핀의 현재 데이터를 읽어올 수 있습니다.

21.3.4 Port Mode Register (PMR)

Address(es): PORT0.PMR 0008 C060h, PORT1.PMR 0008 C061h, PORT2.PMR 0008 C062h, PORT3.PMR 0008 C063h, PORT4.PMR 0008 C064h, PORT5.PMR 0008 C065h, PORTA.PMR 0008 C06Ah, PORTB.PMR 0008 C06Bh, PORTC.PMR 0008 C06Ch, PORTD.PMR 0008 C06Dh, PORTE.PMR 0008 C06Eh, PORTH.PMR 0008 C071h, PORTJ.PMR 0008 C072h



Value after reset: 0 0 0 0 0 0 0 0

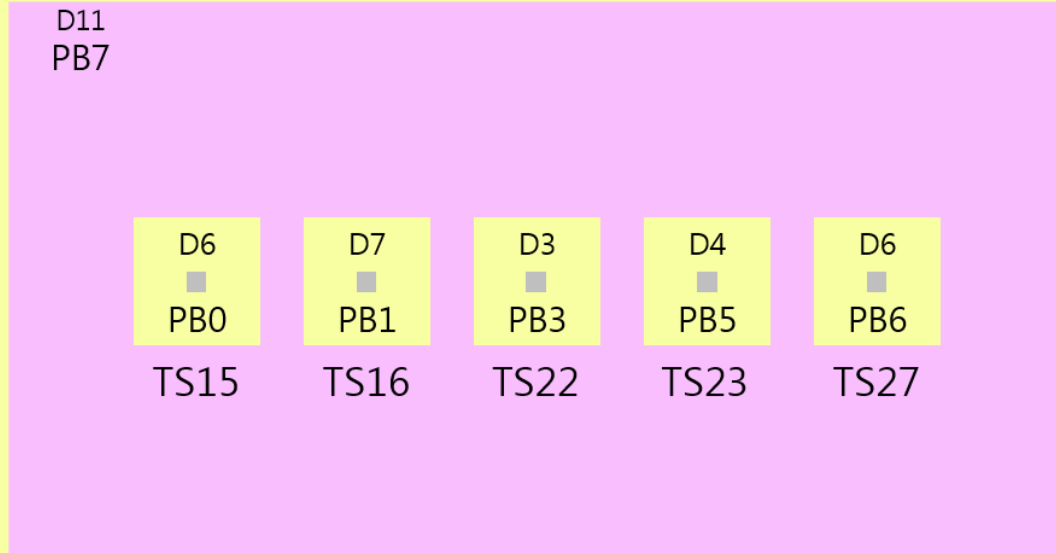
Bit	Symbol	Bit Name	Description	R/W
b0	B0	Pm0 Pin Mode Control	0: Use pin as general I/O port. 1: Use pin as I/O port for peripheral functions.	R/W
b1	B1	Pm1 Pin Mode Control		R/W
b2	B2	Pm2 Pin Mode Control		R/W
b3	B3	Pm3 Pin Mode Control		R/W
b4	B4	Pm4 Pin Mode Control		R/W
b5	B5	Pm5 Pin Mode Control		R/W
b6	B6	Pm6 Pin Mode Control		R/W
b7	B7	Pm7 Pin Mode Control		R/W

m = 0 to 5, A to E, H, J

Port Mode Register(PMR)는 기본값은 0이며, 1로 설정하면 인터럽트나 통신과 같은 기능을 위해 핀을 사용하게 되고, 0으로 설정하면 PODR 레지스터를 통해 제어를 할 수 있게 됩니다.

2.2 소스코드

TS30 ■



```

if ( _0_SUCCESS == R_Set_Cap_Touch_Result_Create( method ) )
{
    ts_result = R_Get_Cap_Touch_Result( method );

    if (1 == (ts_result.button[0] >> 15)) //TS15
    {
        PORTB.PODR.BYTE = 0x01; //PB0 0000 0001
    }
    else if(1 == (ts_result.button[1] >> 0)) //TS16
    {
        PORTB.PODR.BYTE = 0x02; //PB1 0000 0010
    }
    else if(1 == (ts_result.button[1] >> 6)) //TS22
    {
        PORTB.PODR.BYTE = 0x08; //PB3 0000 1000
    }
    else if(1 == (ts_result.button[1] >> 7)) //TS23
    {
        PORTB.PODR.BYTE = 0x20; //PB5 0010 0000
    }
    else if(1 == (ts_result.button[1] >> 11)) //TS27
    {
        PORTB.PODR.BYTE = 0x40; //PB6 0100 0000
    }
    else if(1 == (ts_result.button[1] >> 14)) //TS30
    {
        PORTB.PODR.BYTE = 0x80; //PB7 1000 0000
    }
}

```

ts_result는 button, slider, wheel로 이루어져 있습니다. 그 중 button만 설명합니다.

ts_result.button은 [0~2] 값을 가지고 있고, 값의 출력은 아래의 수식과 같습니다.

$$ts_result.button[n'] = 2^{TSn-16 \times n'}$$

ts_result.button[0]	TS0 ~ TS15
ts_result.button[1]	TS16 ~ TS31
ts_result.button[2]	TS32 ~ TS47

TS15번만 터치인식이 되면 button[0]은 2^{15} 인 32768값이 나오게 됩니다.

```

if (1 == (ts_result.button[0] >> 15)) //TS15
{
    PORTB.PODR.BYTE
}
else if(1 == (ts_res
{
    PORTB.PODR.BYTE
}

```

Expression	Type	Value
ts_result.button	uint16_t [3]	0x5d0 <ts_result>
(*) ts_result.button[0]	uint16_t	32768
(*) ts_result.button[1]	uint16_t	0
(*) ts_result.button[2]	uint16_t	0

TS27번만 터치가 인식됐을 경우에는, button[1]이 2^{11} 인 2048값이 나오게 됩니다.

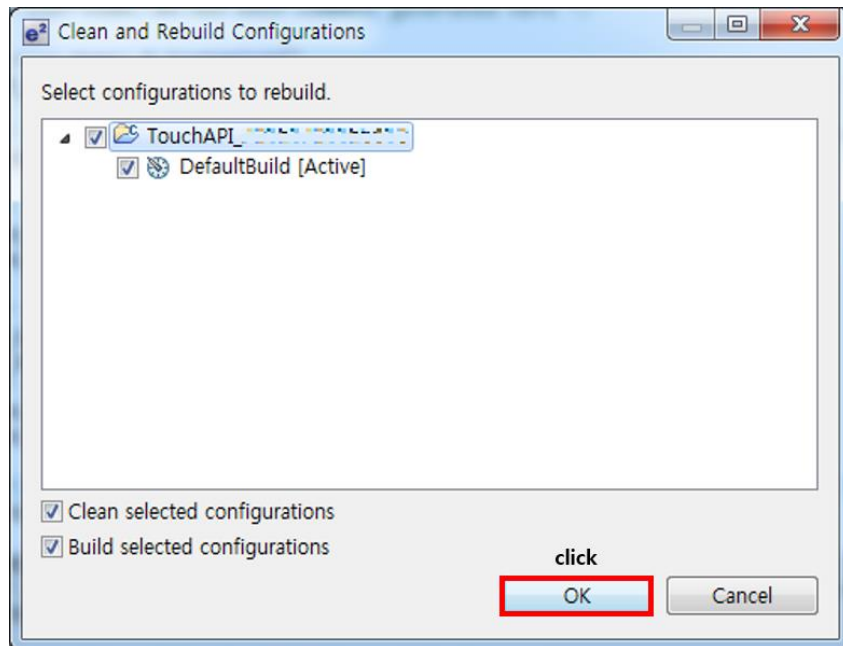
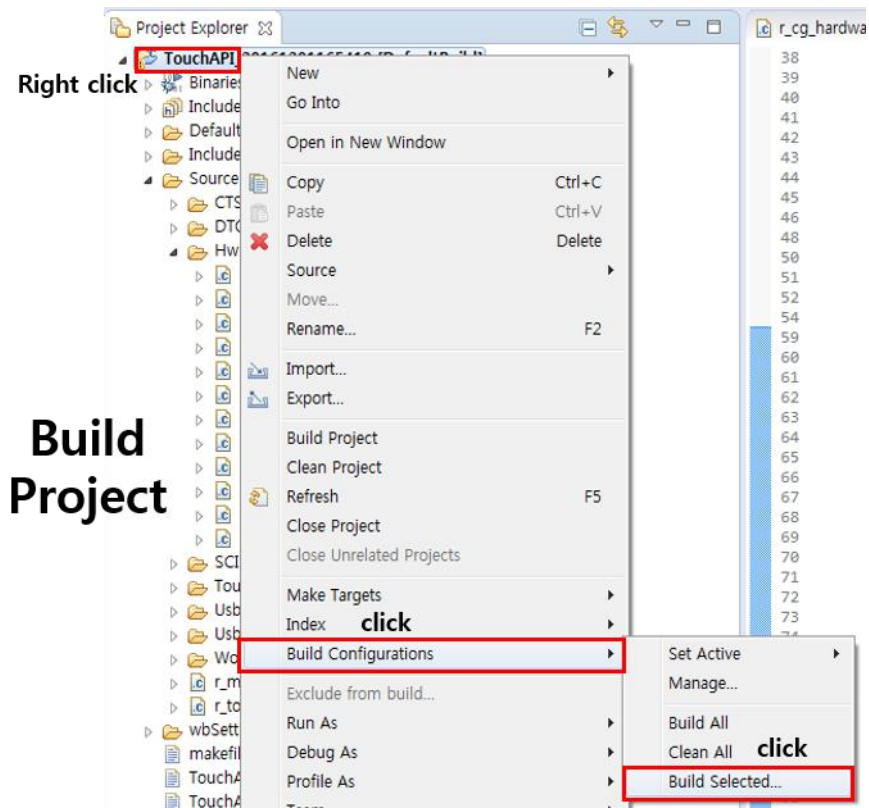
```

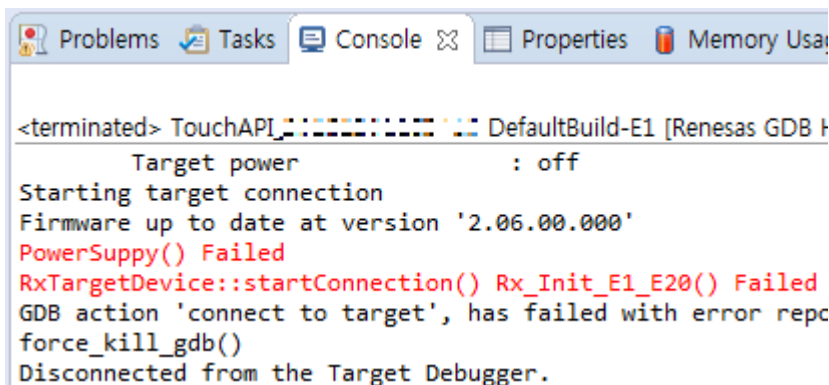
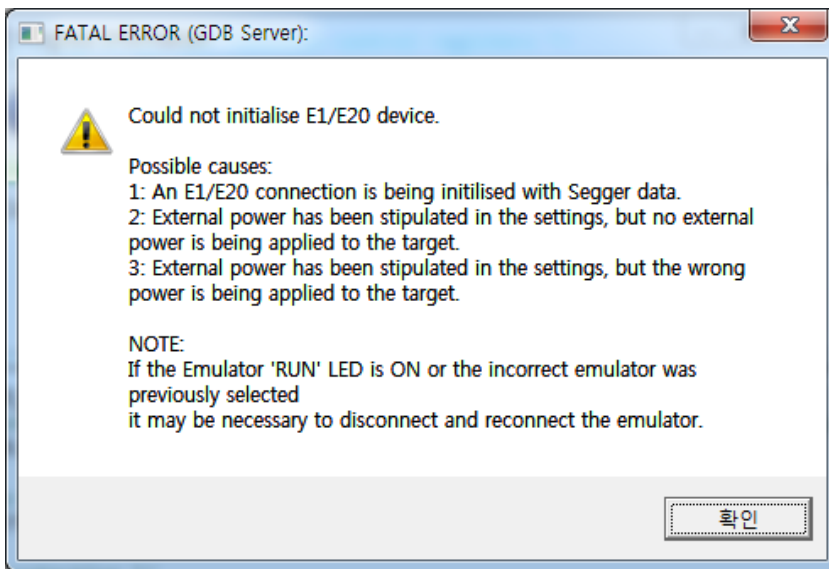
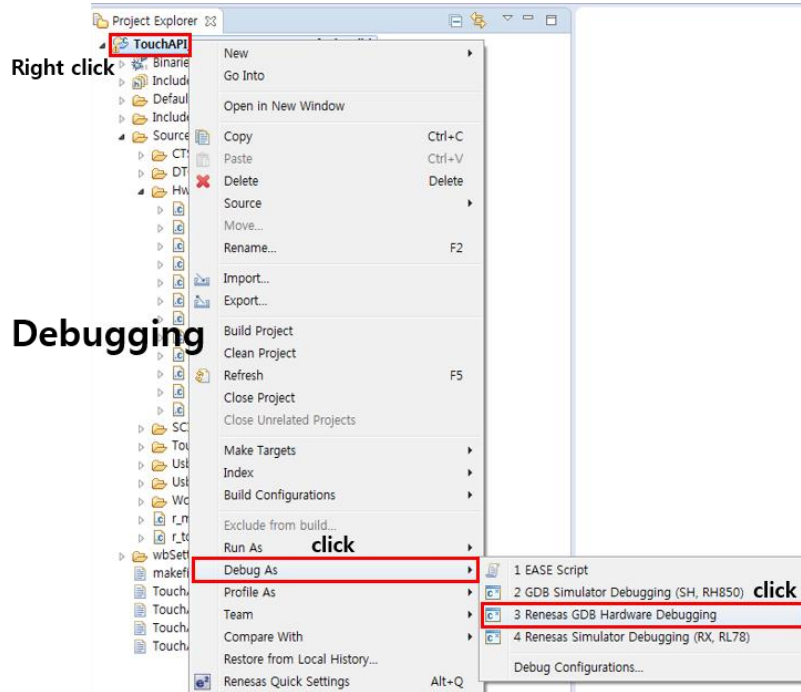
else if(1 == (ts_result.button[1] >> 11)) //TS27
{
    PORTB.PODR.BYTE = 0x
}
else if(1 == (ts_result.
{
    PORTB.PODR.BYTE = 0x
}

```

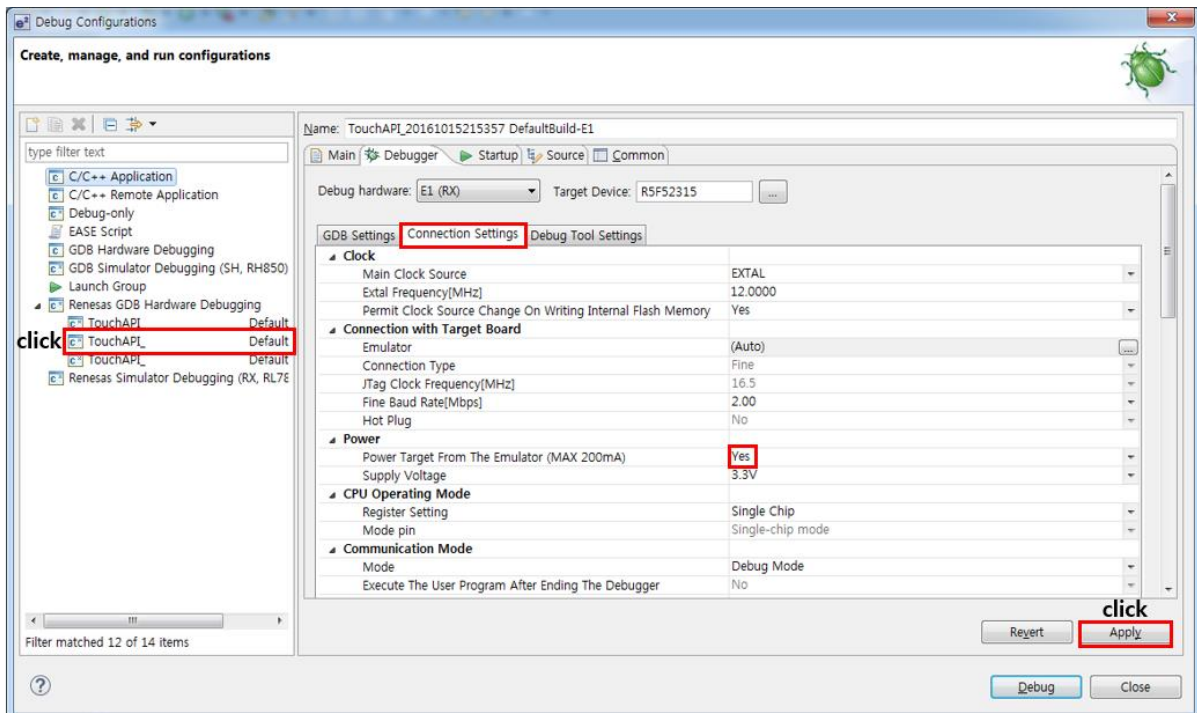
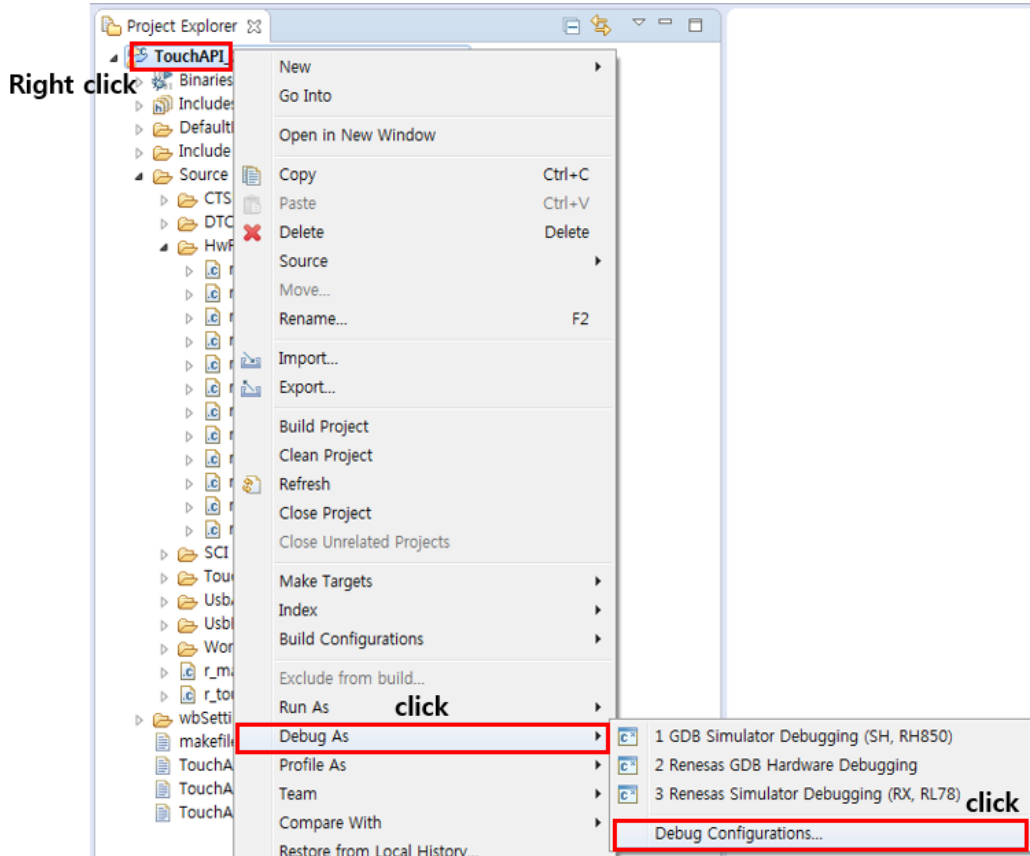
Expression	Type	Value
ts_result.button	uint16_t [3]	0x5d0 <ts_result>
(*) ts_result.button[0]	uint16_t	0
(*) ts_result.button[1]	uint16_t	2048
(*) ts_result.button[2]	uint16_t	0

3 프로젝트 라이팅



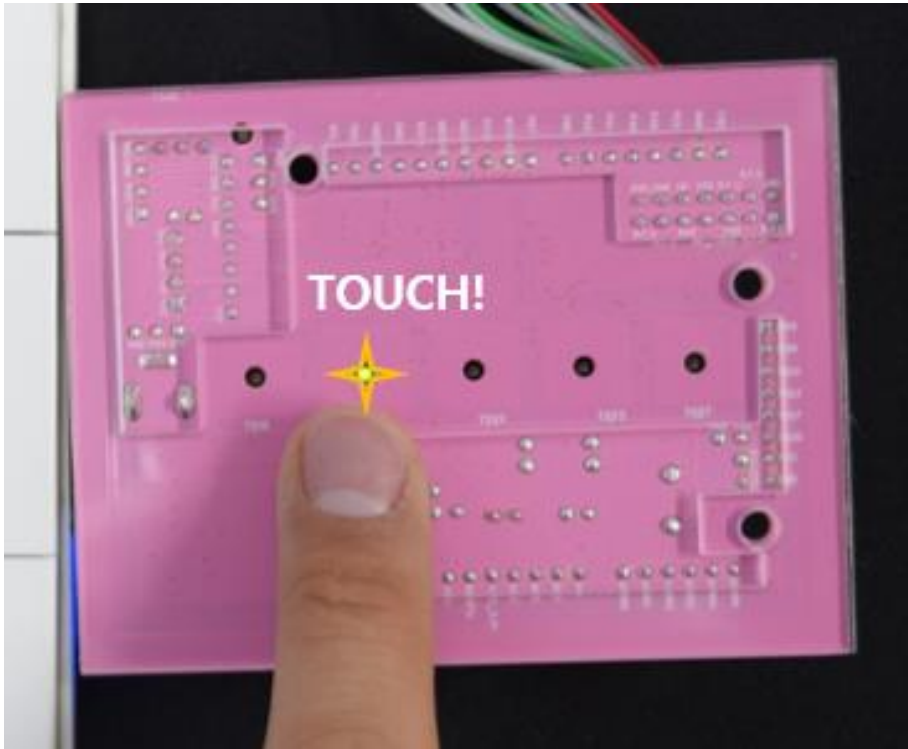


디버깅 진행 중에, 위 같은 에러 메시지가 뜬다면, 밑에 설정을 참고합니다.



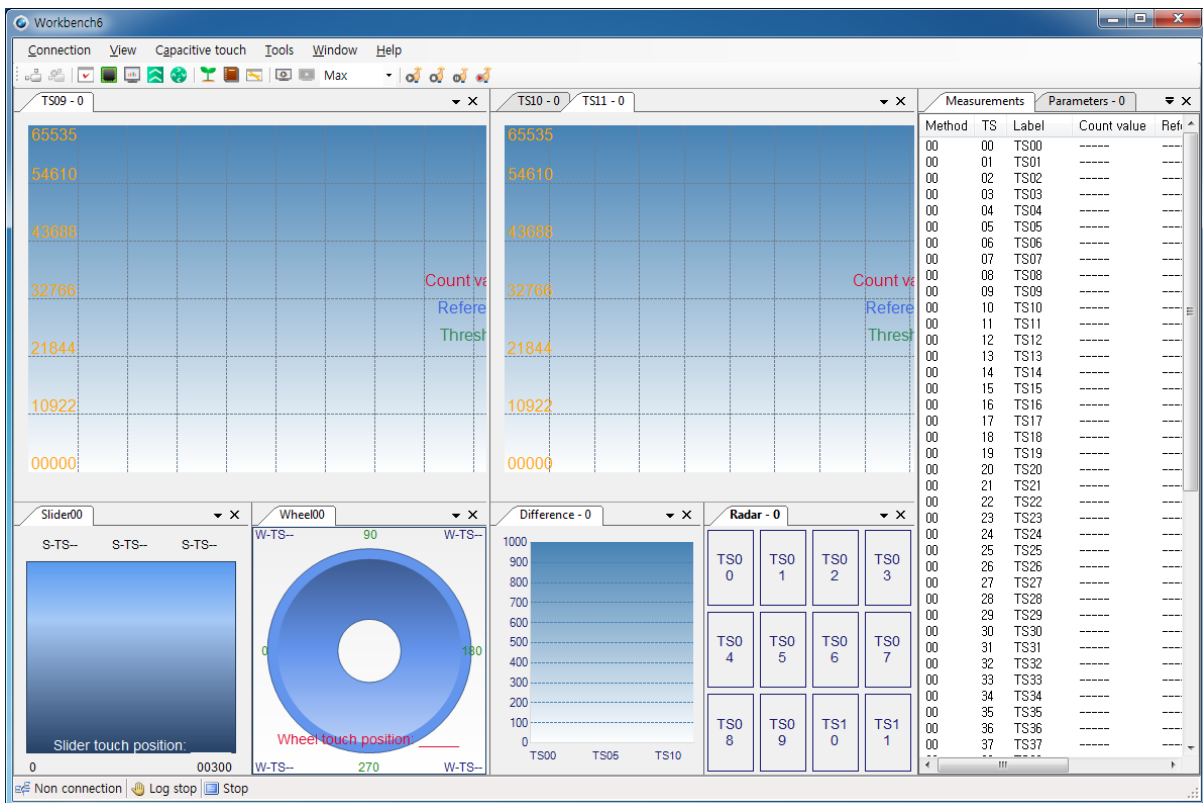
프로젝트를 만든 직후에는 위 사진과 같이 에뮬레이터를 통해 전원을 공급해주는 옵션이 NO로 되어있습니다. Yes로 변경하고 진행하면 추가전원(어댑터, 마이크로USB 등)의 필요 없이 진행할 수 있을 것입니다.

4 프로젝트 실행

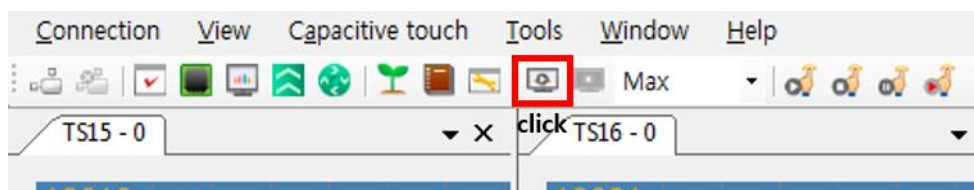
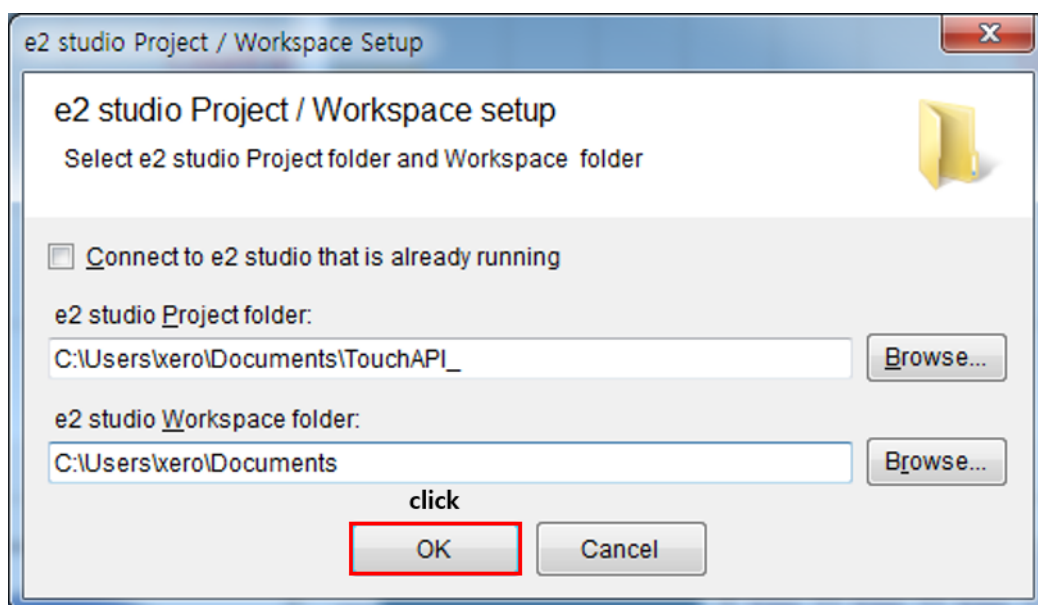
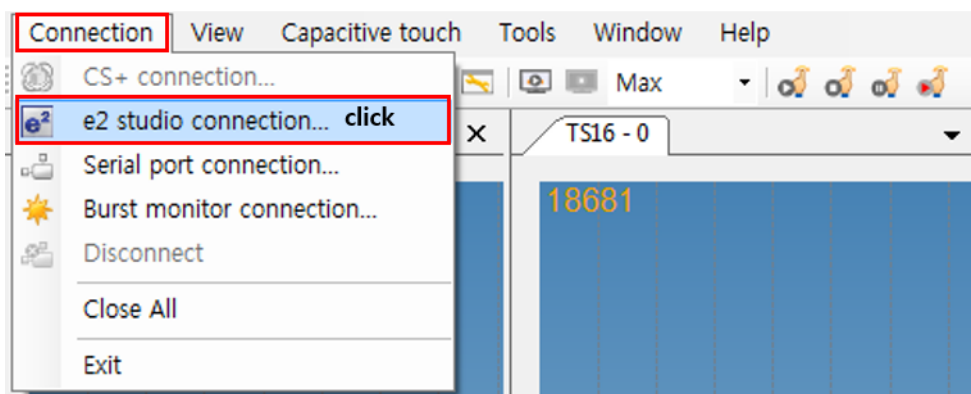


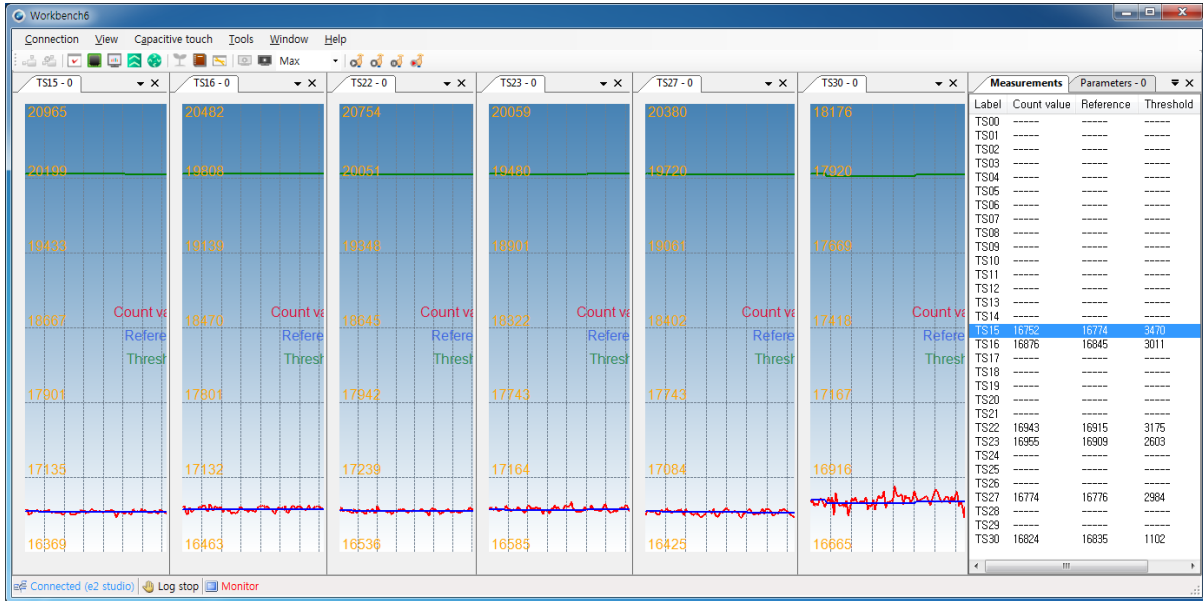
5개의 버튼 각각 눌러보면 해당 LED가 점등될 것이며, 주변의 근접센서인 TS30는 0.5~1cm정도 가까이 손가락을 대보면 LED가 점등될 것입니다.

5 Workbench6 튜닝



click





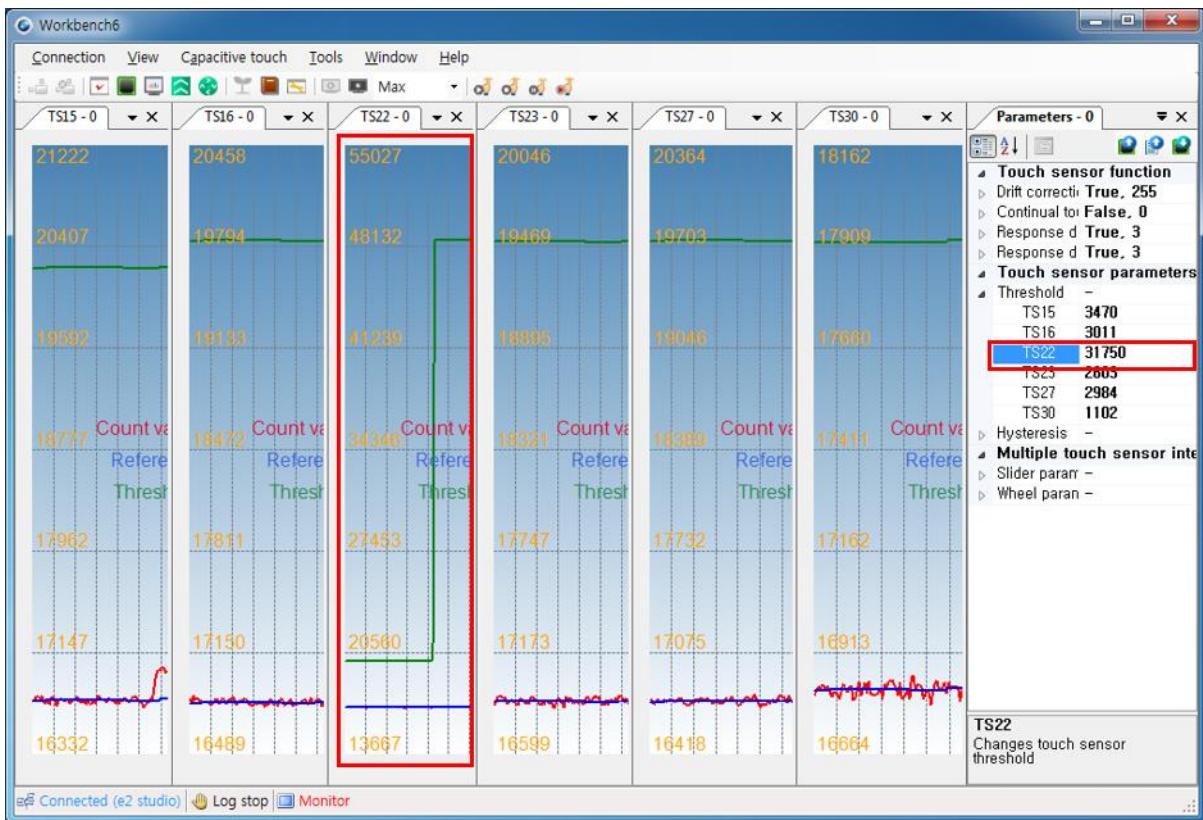
측정 데이터는 다를 수 있습니다.

Count Value : 계속 변하는 터치 센서 값

Reference Value : Count Value를 평균적으로 계산하여 일정한 수치로 나타낸 값

Threshold Value : 터치 인식 범위 값

$$\text{Touch?} = \text{Count Value} > \text{Reference Value} + \text{Threshold Value}$$



Threshold값은 다음과 같이 Parameters탭에서 변경이 가능합니다.

▶ Touch sensor function	
▶ Drift correction	True, 255
▶ Continual touch limiter	False, 0
▶ Response delay time to touch	True, 3
▶ Response delay time to non touch	True, 3

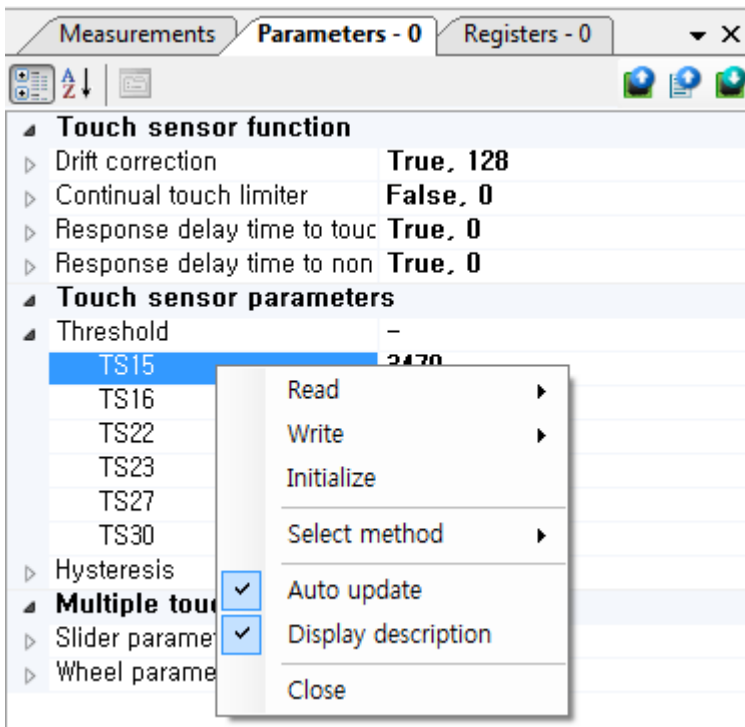
Touch sensor function에서는 Reference값의 계산속도, 터치응답시간 등을 지정해 줄 수 있습니다.

Reference값은 약 4초마다 Count value의 값을 평균적으로 계산해 일정한 수치로 만듭니다.

Reference값의 계산속도는 Drift Correction부분을 조절하면 빠르게 할 수 있습니다.

$$\approx \text{Drift Correction Value} * 0.016 [\text{Sec}] \text{ (범위 : 0 ~ 255)}$$

터치 응답시간은 기본적으로 3으로 설정되어 있지만, 0으로 설정하는 것을 추천드립니다.



Parameters탭에서 우클릭을 하게 되면 위와 같은 창이 뜨게 됩니다.

Read	Read From target system	연결된 보드에서 Parameter값을 읽어옵니다
	Read from parameter file	PC에 저장된 Parameter파일을 읽어옵니다
Write	Write to target system	연결된 보드에 저장합니다
	Write to parameter file	parameter파일에 저장합니다
	Write to Touch API	프로젝트에 저장합니다
Initialize		변경 전 초기값으로 재설정합니다