SiAL

Getting started for NS-RX231

e²studio execution

Contents

1 Create the Project

Let's create the source with the First Step Guide function of Workbench6 which makes a simple project using CTSU (Capacitive Touch Sensor Unit) API.

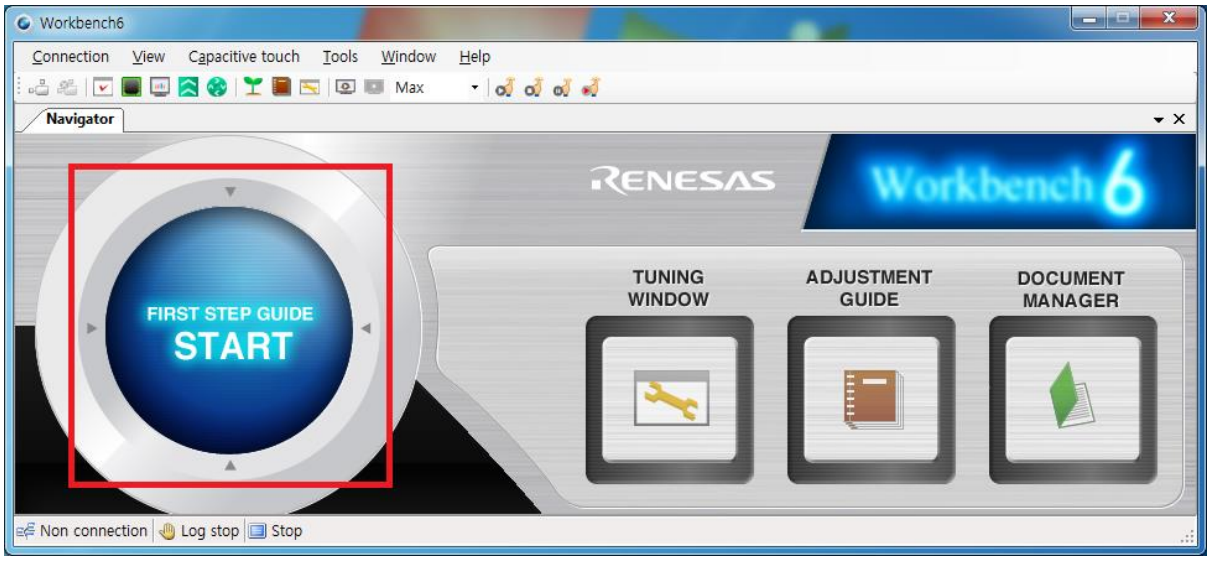
Before the main text, please refer to the video which deals with the function of Workbench6.

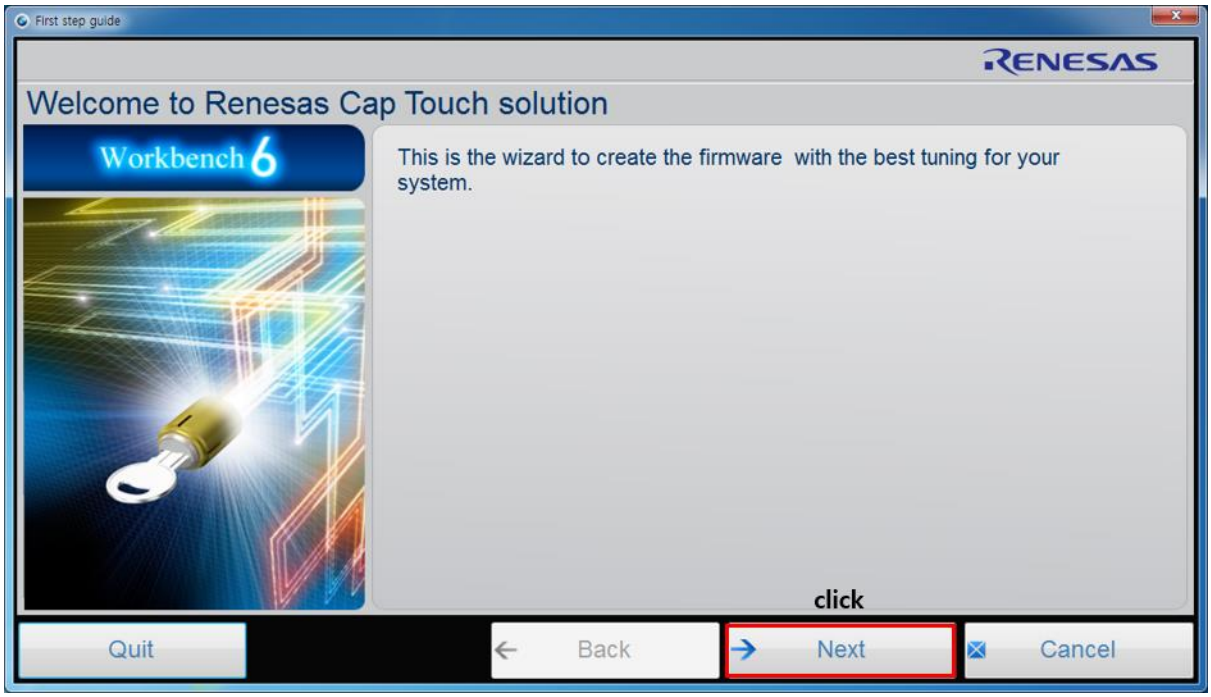
Integrated Development Environment for Renesas Capacitive Touch Workbench6(ver1.03)

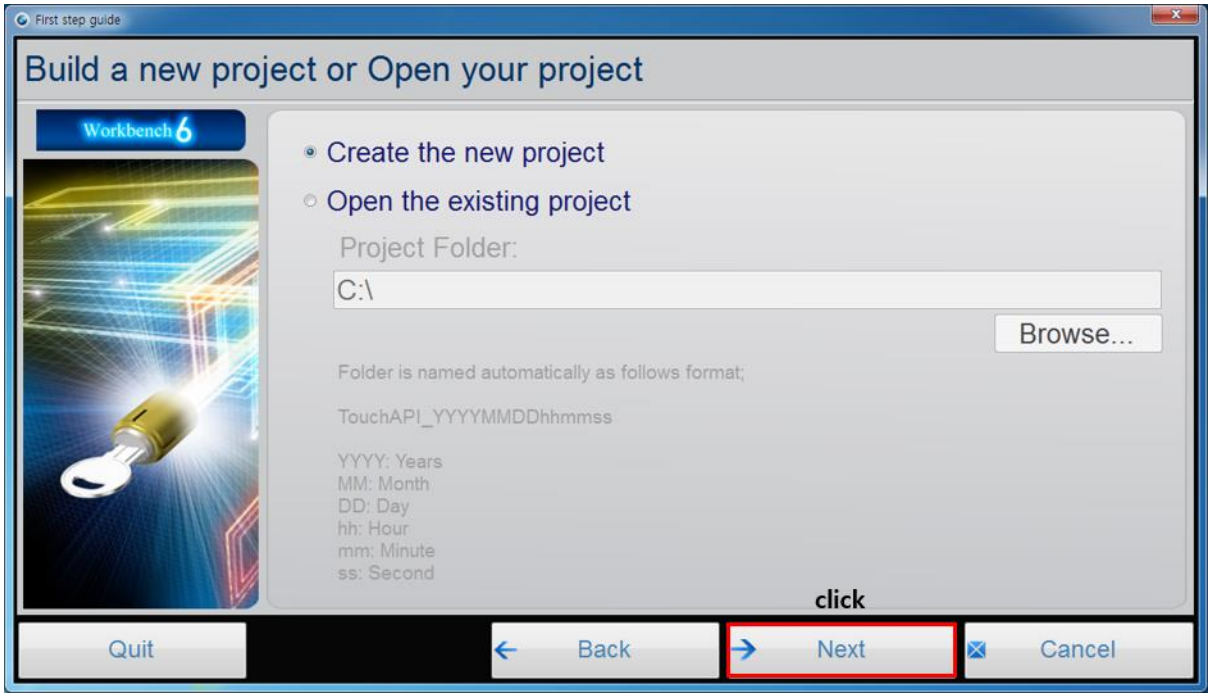
https://www.youtube.com/watch?v=8oq5iyGyaMw ①

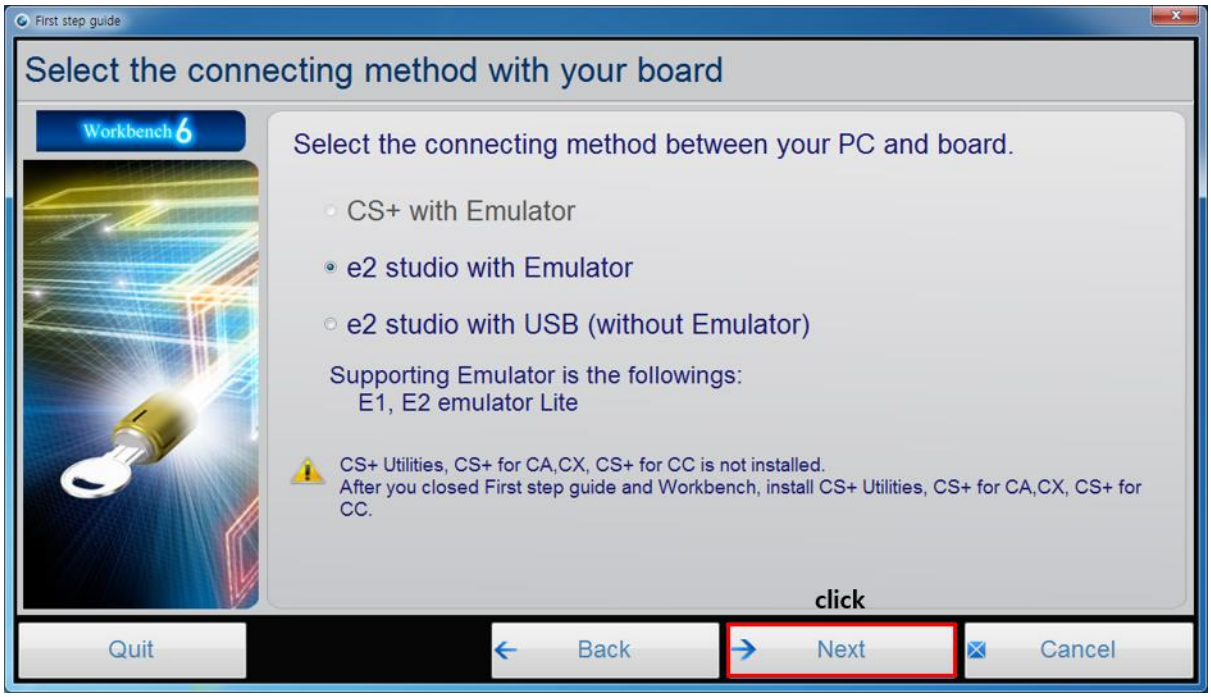
https://www.youtube.com/watch?v=tdIpiyTrKVM ②

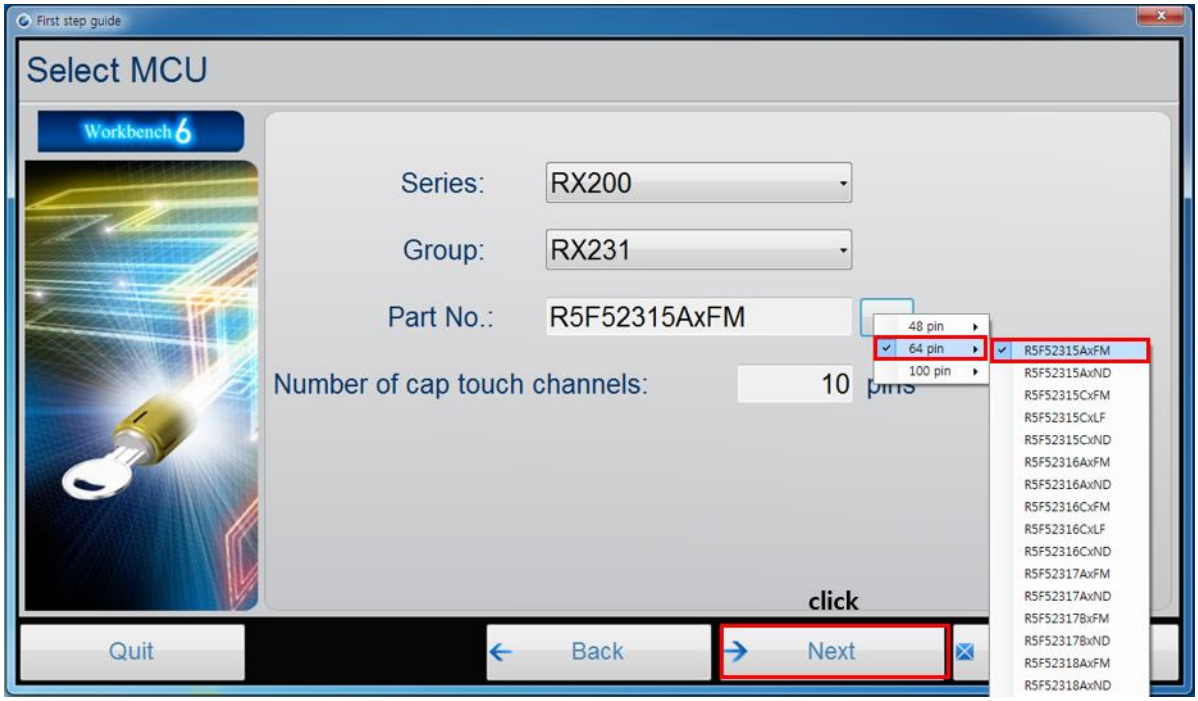
https://www.youtube.com/watch?v=2nQ9OcP5Cgg ③

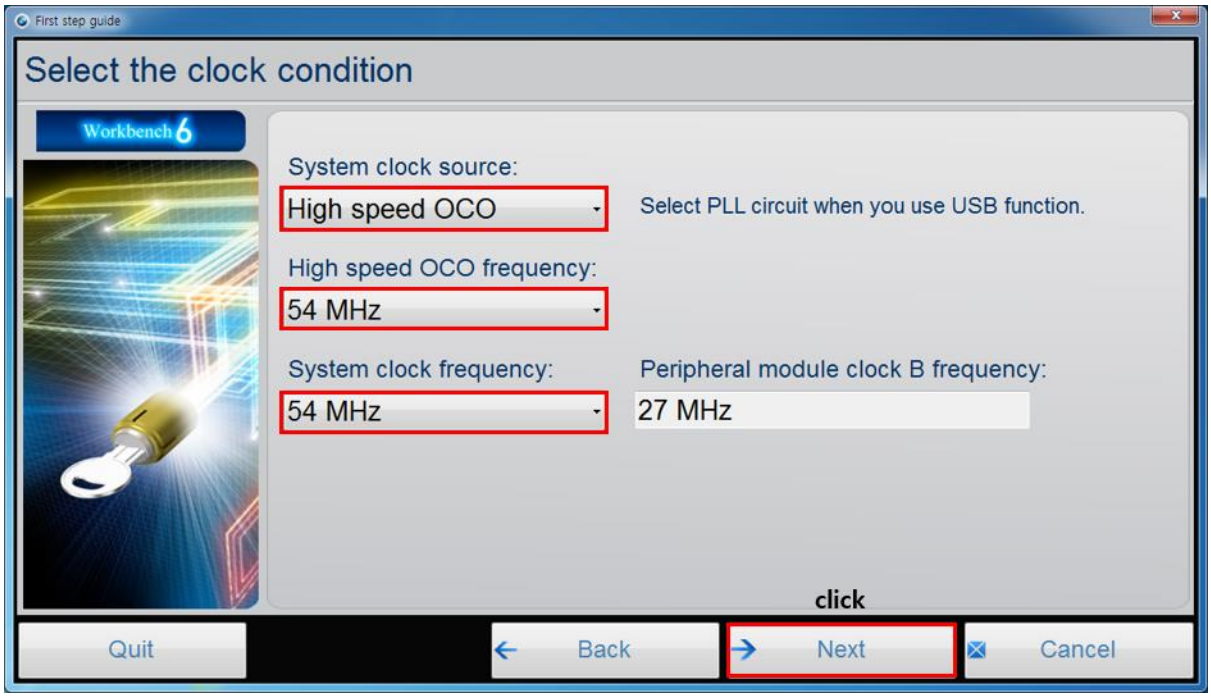


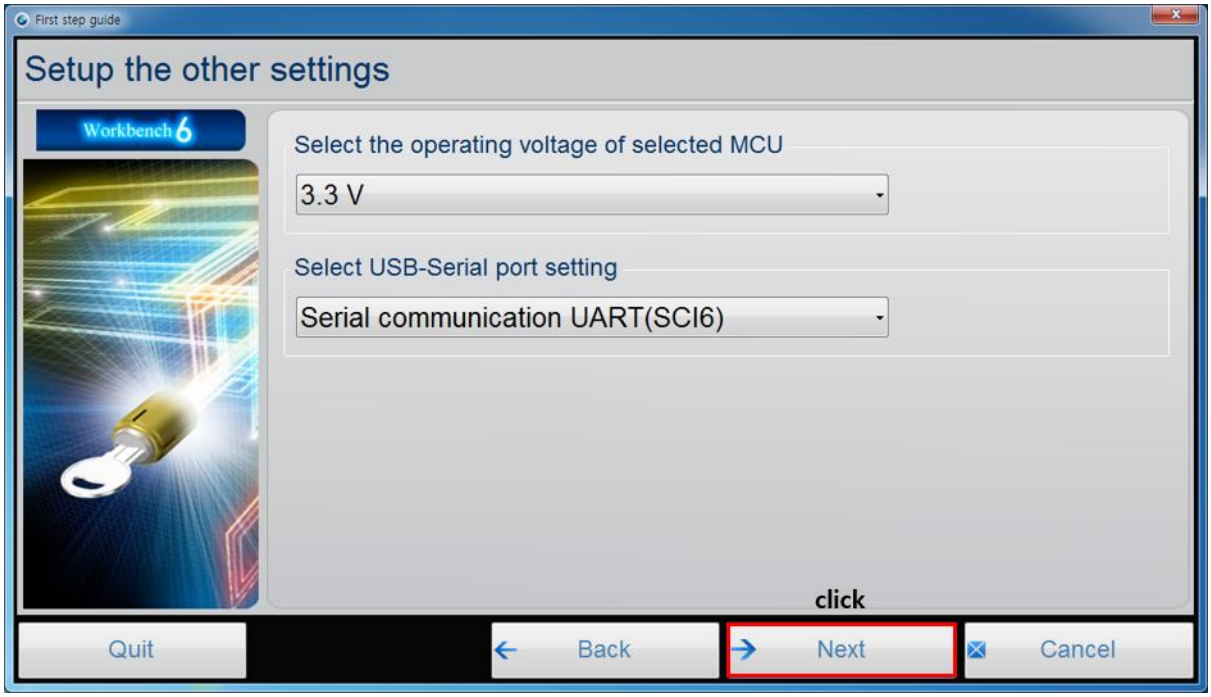

First step guide
RENESAS
Welcome to Renesas Cap Touch solution
Workbench 6
This is the wizard to create the firmware with the best tuning for your system.
click
Quit
Back
Next
Cancel

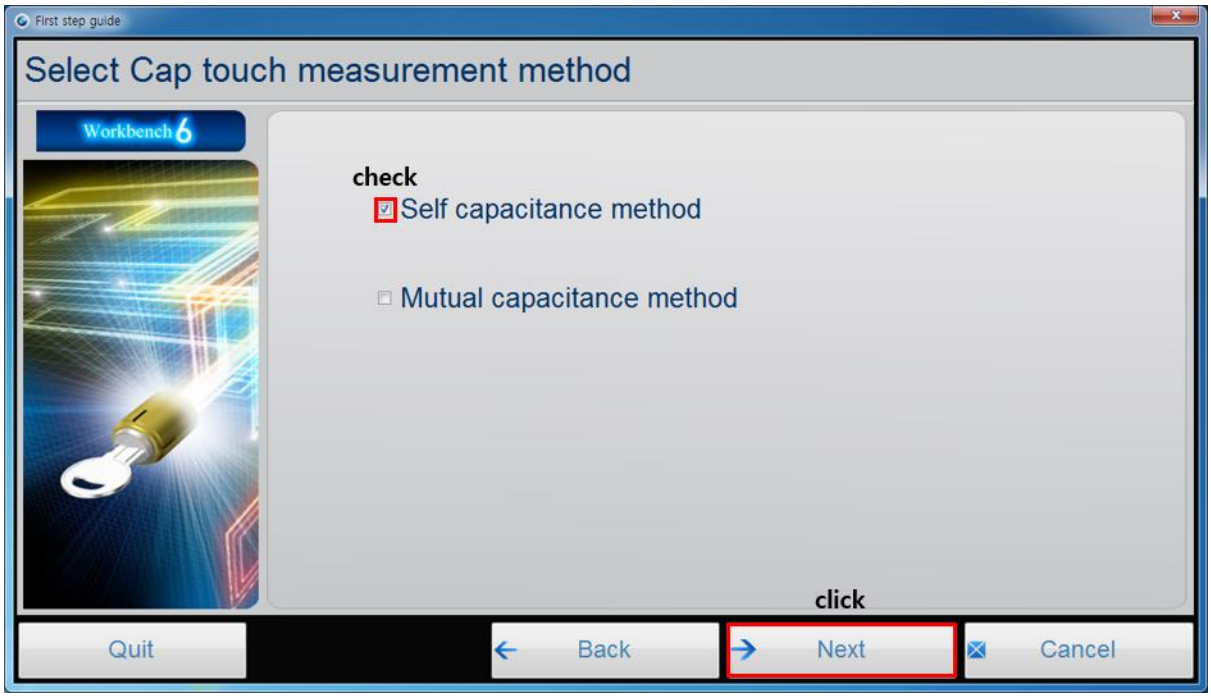

First step guide
Build a new project or Open your project
Workbench 6
Create the new project
Open the existing project
Project Folder:
C:\
Browse...
Folder is named automatically as follows format;
TouchAPI_YYYYMMDDhhmmss
YYYY: Years
MM: Month
DD: Day
hh: Hour
mm: Minute
ss: Second
click
Quit
Back
Next
Cancel

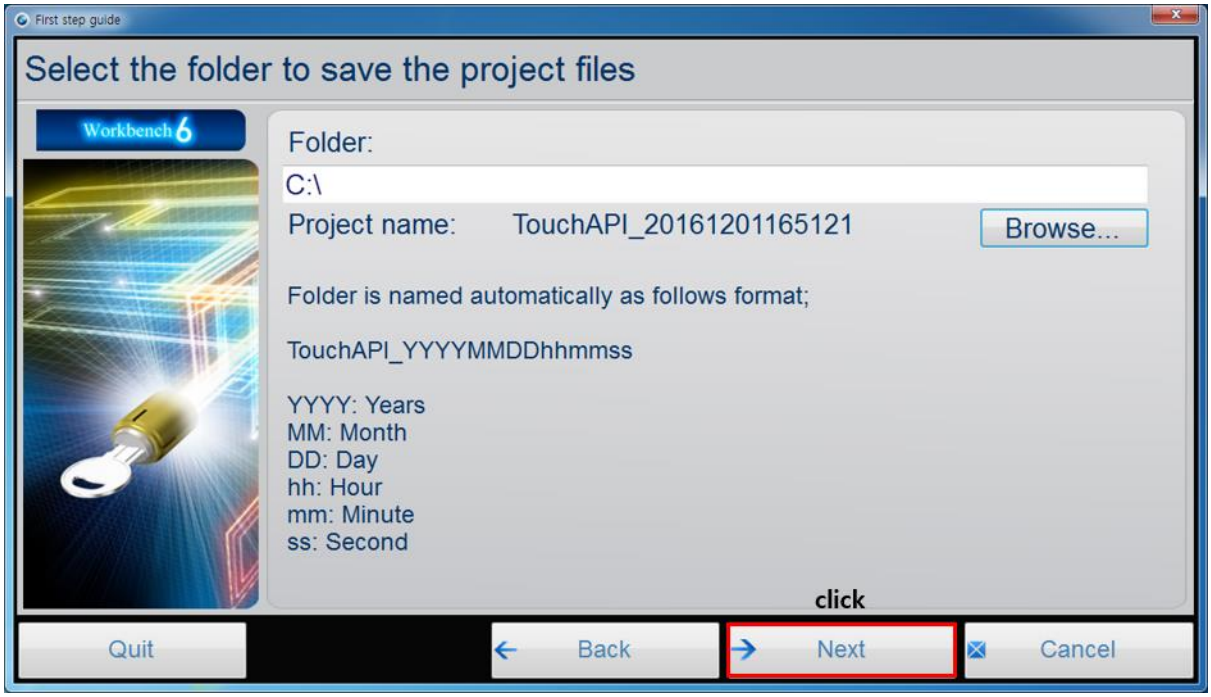

First step guide
Select the connecting method with your board
Workbench 6
Select the connecting method between your PC and board.
CS+ with Emulator
e2 studio with Emulator
e2 studio with USB (without Emulator)
Supporting Emulator is the followings:
E1, E2 emulator Lite
CS+ Utilities, CS+ for CA,CX, CS+ for CC is not installed.
After you closed First step guide and Workbench, install CS+ Utilities, CS+ for CA,CX, CS+ for CC.
click
Quit
Back
Next
Cancel

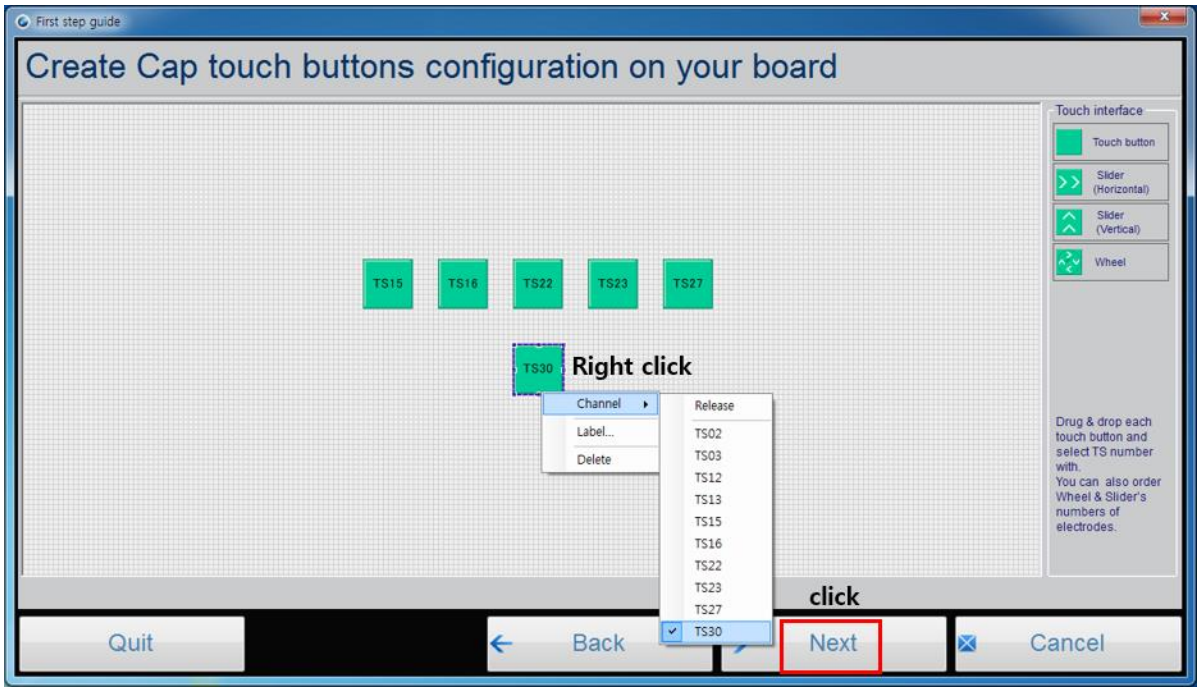

First step guide
Select MCU
Workbench 6
Series:
RX200
Group:
RX231
Part No.:
R5F52315AxFM
Number of cap touch channels:
10
pins
48 pin
64 pin
100 pin
R5F52315AxFM
R5F52315AxND
R5F52315CxFM
R5F52315CxLF
R5F52315CxND
R5F52316AxFM
R5F52316AxND
R5F52316CxFM
R5F52316CxLF
R5F52316CxND
R5F52317AxFM
R5F52317AxND
R5F52317BxFM
R5F52317BxND
R5F52318AxFM
R5F52318AxND
click
Quit
Back
Next

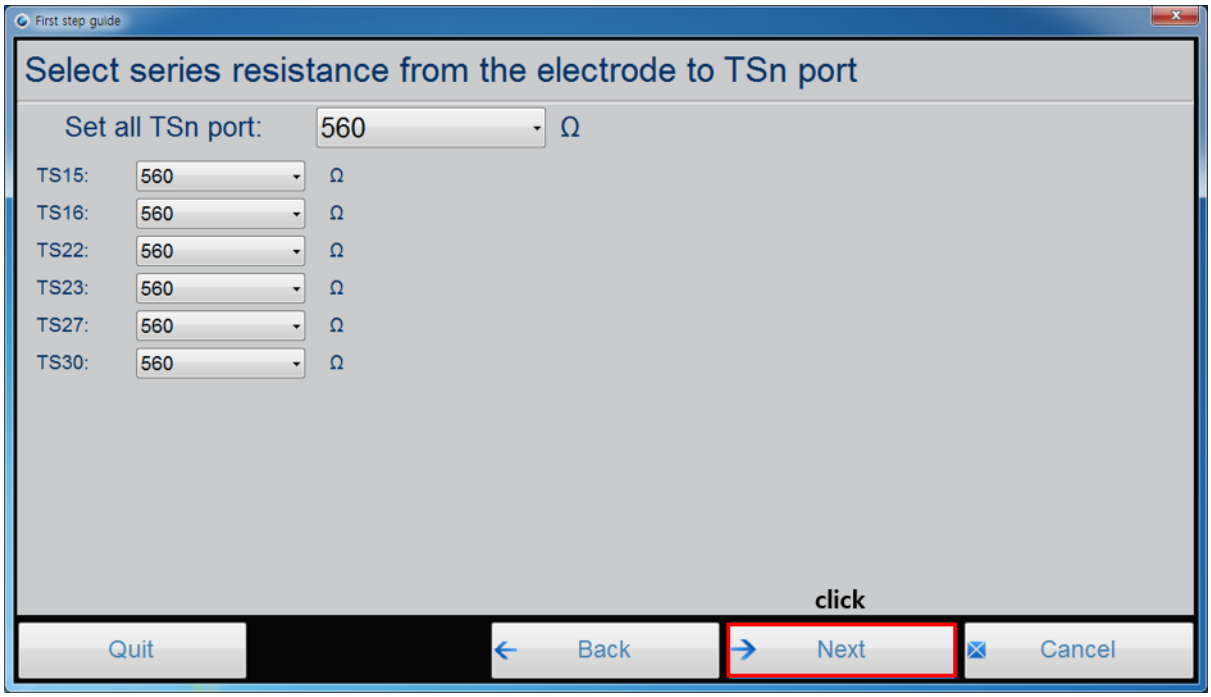

First step guide
Select the clock condition
Workbench 6
System clock source:
High speed OCO
Select PLL circuit when you use USB function.
High speed OCO frequency:
54 MHz
System clock frequency:
54 MHz
Peripheral module clock B frequency:
27 MHz
click
Quit
Back
Next
Cancel

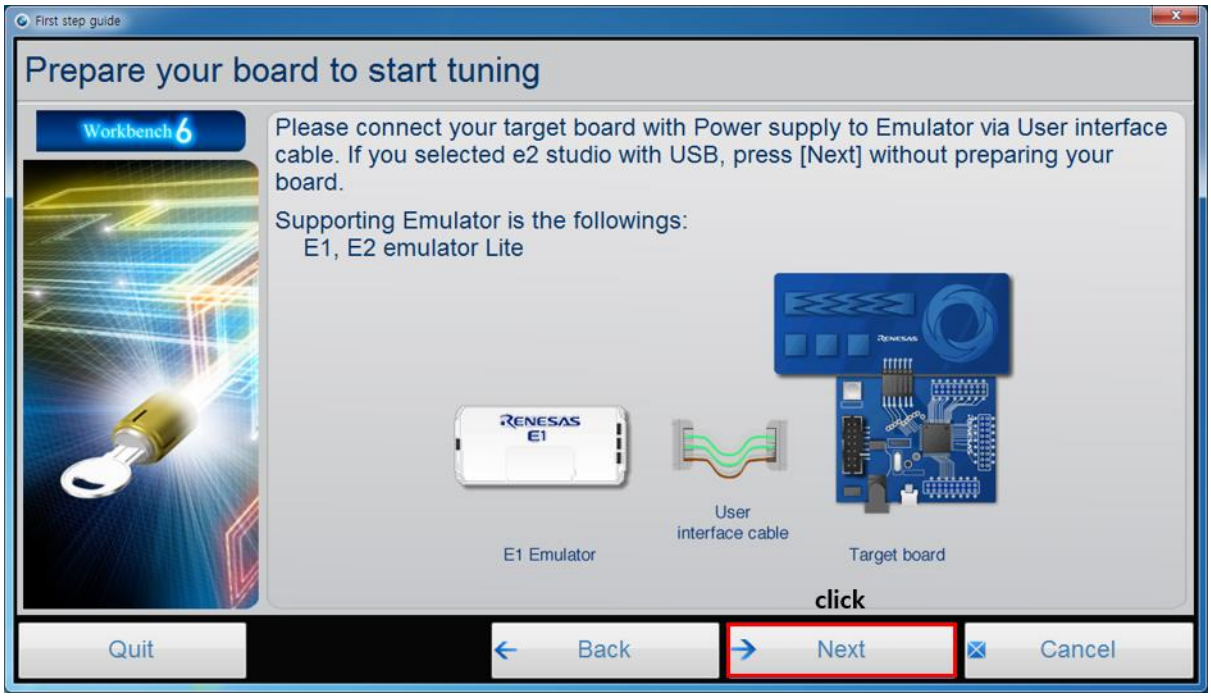

First step guide
Setup the other settings
Workbench 6
Select the operating voltage of selected MCU
3.3 V
Select USB-Serial port setting
Serial communication UART(SCI6)
click
Quit
Back
Next
Cancel

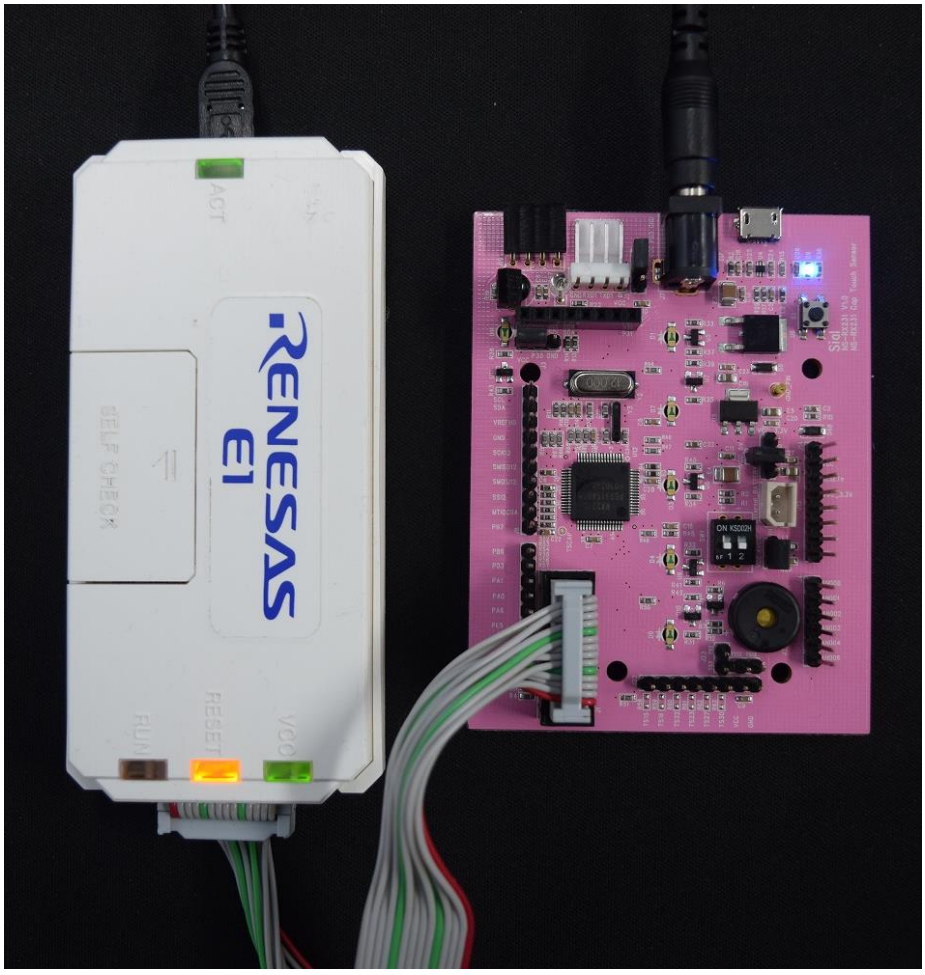

First step guide
Select Cap touch measurement method
Workbench 6
check
Self capacitance method
Mutual capacitance method
click
Quit
Back
Next
Cancel

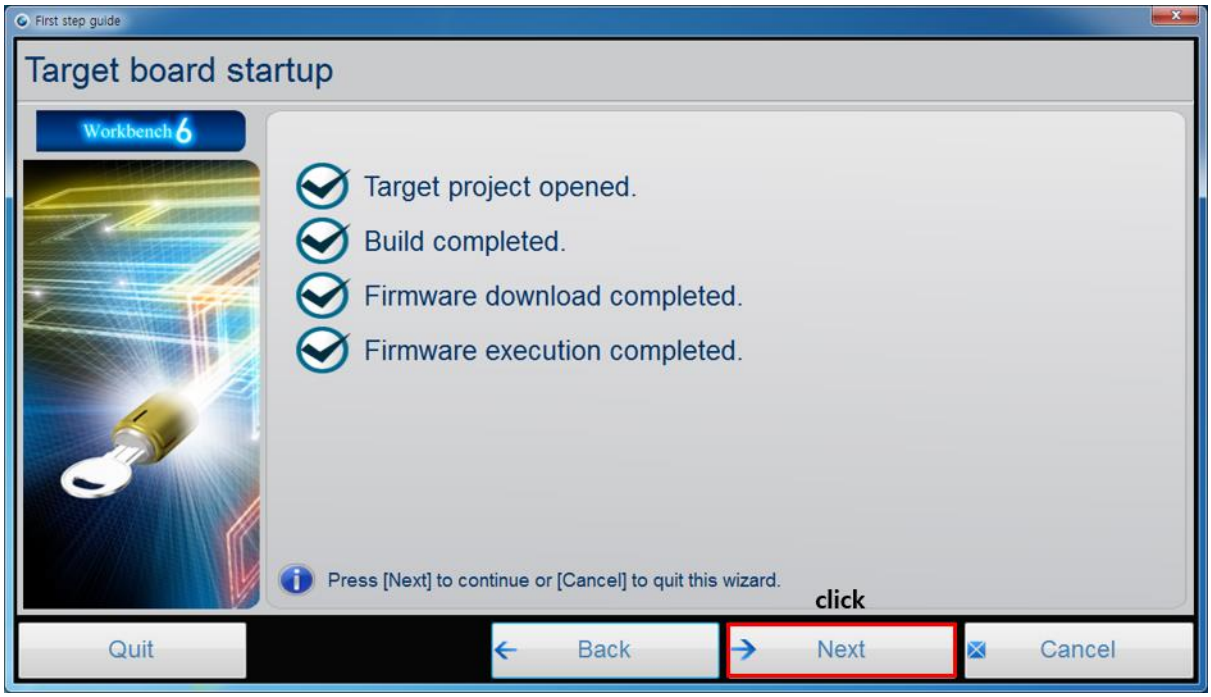

First step guide
Select the folder to save the project files
Workbench 6
Folder:
C:\
Project name: TouchAPI_20161201165121
Browse...
Folder is named automatically as follows format;
TouchAPI_YYYYMMDDhhmmss
YYYY: Years
MM: Month
DD: Day
hh: Hour
mm: Minute
ss: Second
click
Quit
Back
Next
Cancel

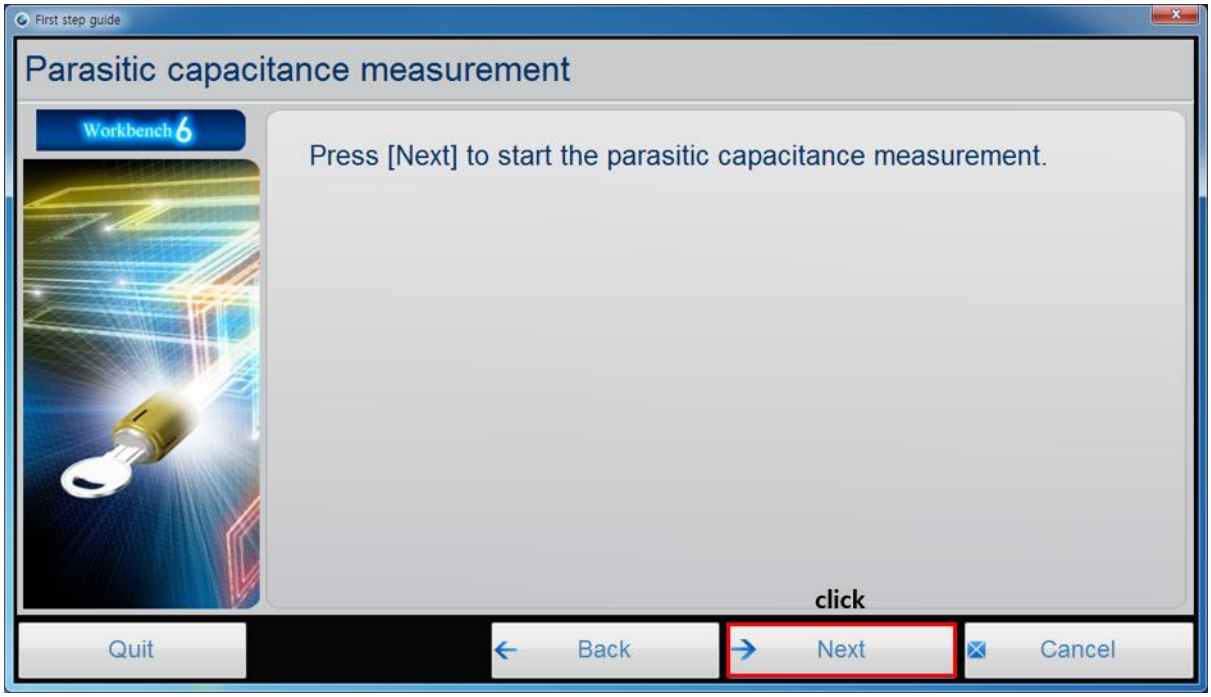

First step guide
Create Cap touch buttons configuration on your board
Touch interface
Touch button
Slider (Horizontal)
Slider (Vertical)
Wheel
TS15
TS16
TS22
TS23
TS27
TS30
Right click
Channel
Label...
Delete
Release
TS02
TS03
TS12
TS13
TS15
TS16
TS22
TS23
TS27
TS30
Drug & drop each touch button and select TS number with.
You can also order Wheel & Slider's numbers of electrodes.
click
Quit
Back
Next
Cancel

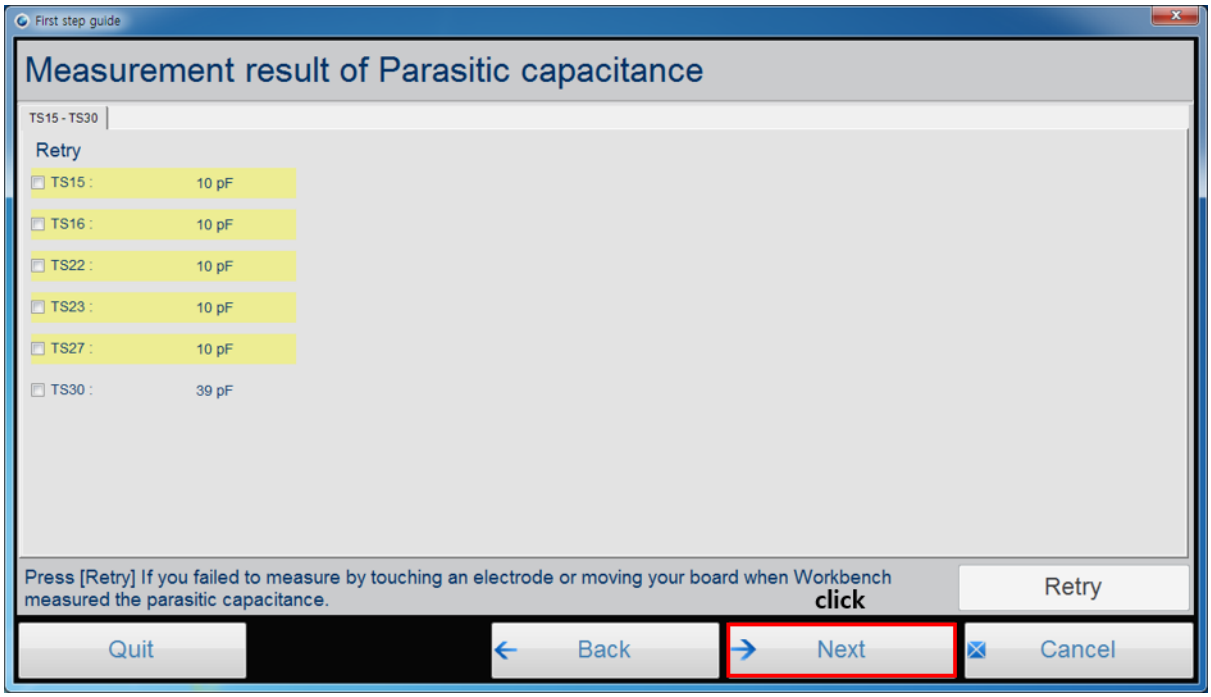

First step guide
Select series resistance from the electrode to TSn port
Set all TSn port:
560
Ω
TS15:
560
Ω
TS16:
560
Ω
TS22:
560
Ω
TS23:
560
Ω
TS27:
560
Ω
TS30:
560
Ω
click
Quit
Back
Next
Cancel


First step guide
Prepare your board to start tuning
Workbench 6
Please connect your target board with Power supply to Emulator via User interface cable. If you selected e2 studio with USB, press [Next] without preparing your board.
Supporting Emulator is the followings:
E1, E2 emulator Lite
RENESAS E1
E1 Emulator
User interface cable
Target board
click
Quit
Back
Next
Cancel

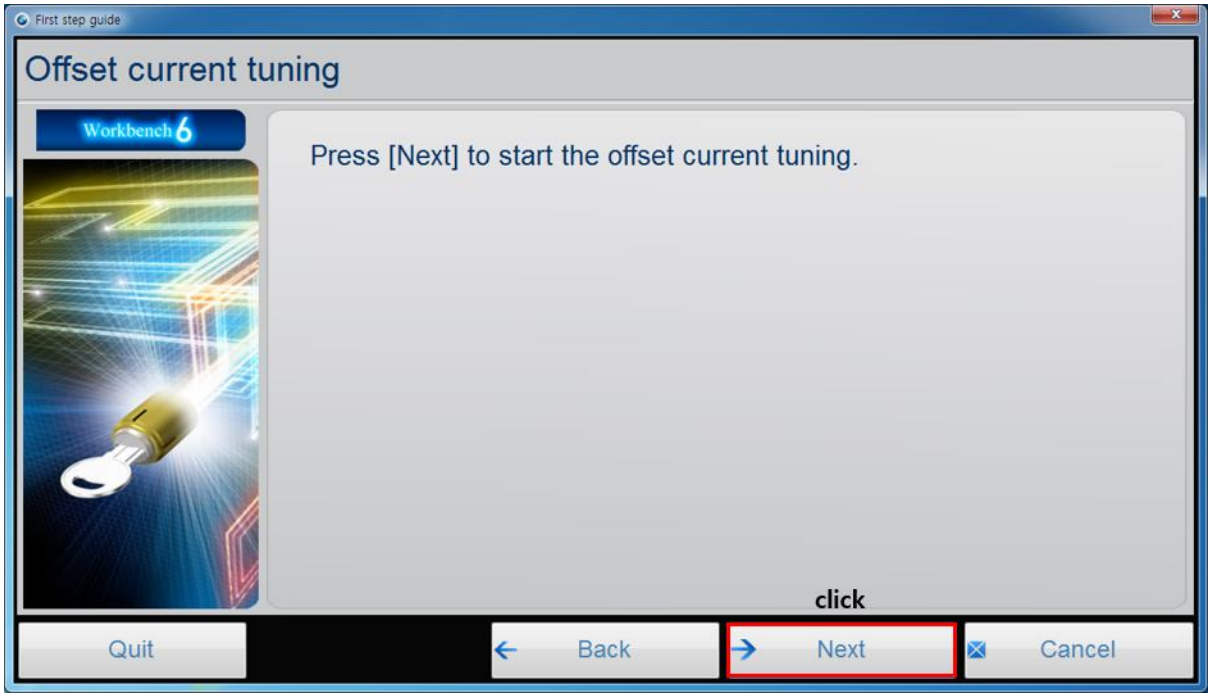

RENESAS
E1

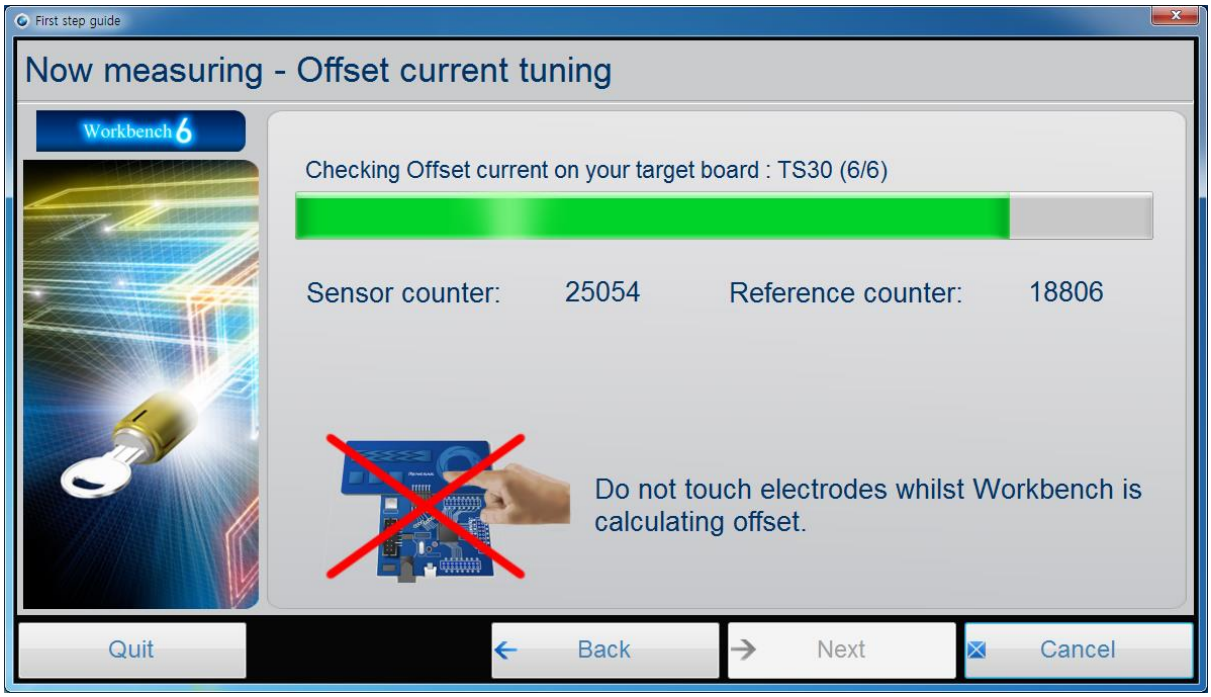

First step guide
Target board startup
Workbench 6
Target project opened.
Build completed.
Firmware download completed.
Firmware execution completed.
Press [Next] to continue or [Cancel] to quit this wizard.
click
Quit
Back
Next
Cancel

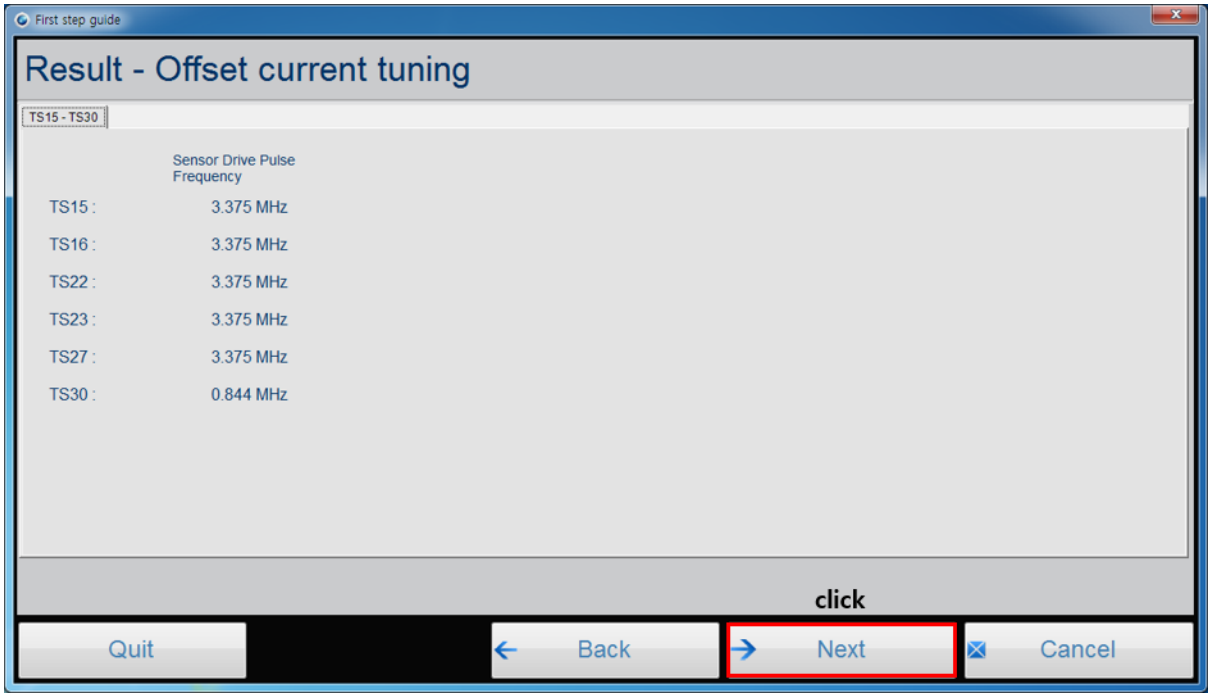

First step guide
Parasitic capacitance measurement
Workbench 6
Press [Next] to start the parasitic capacitance measurement.
click
Quit
Back
Next
Cancel



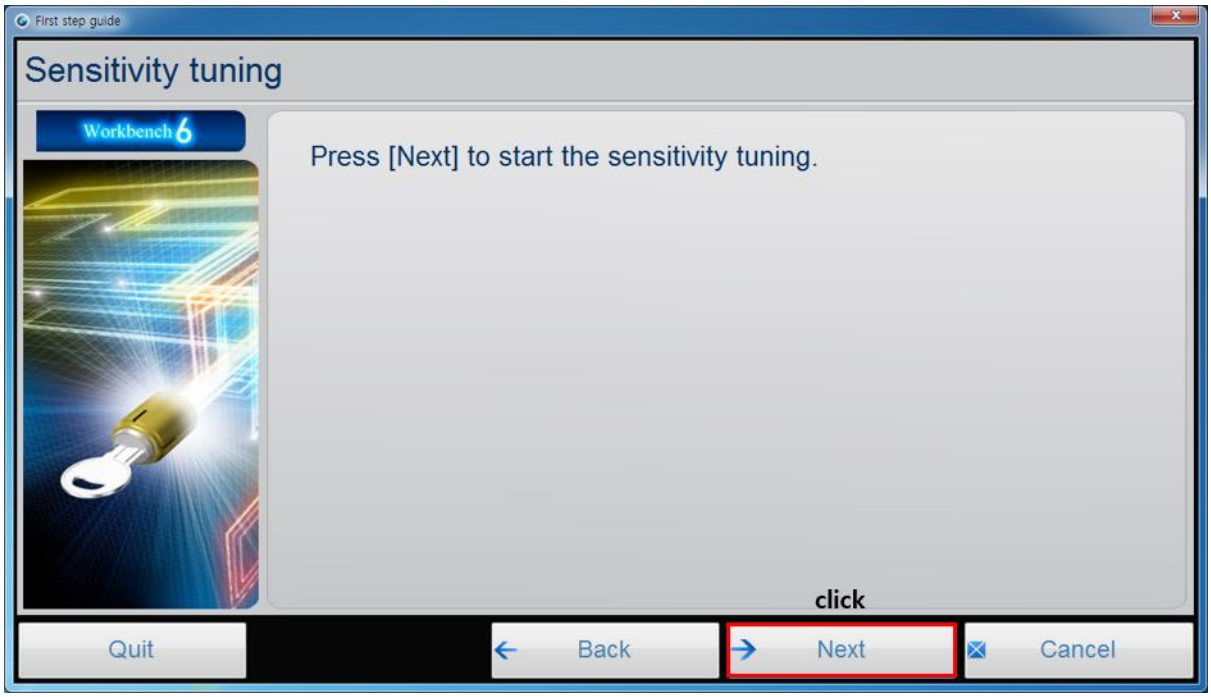
These values can be different

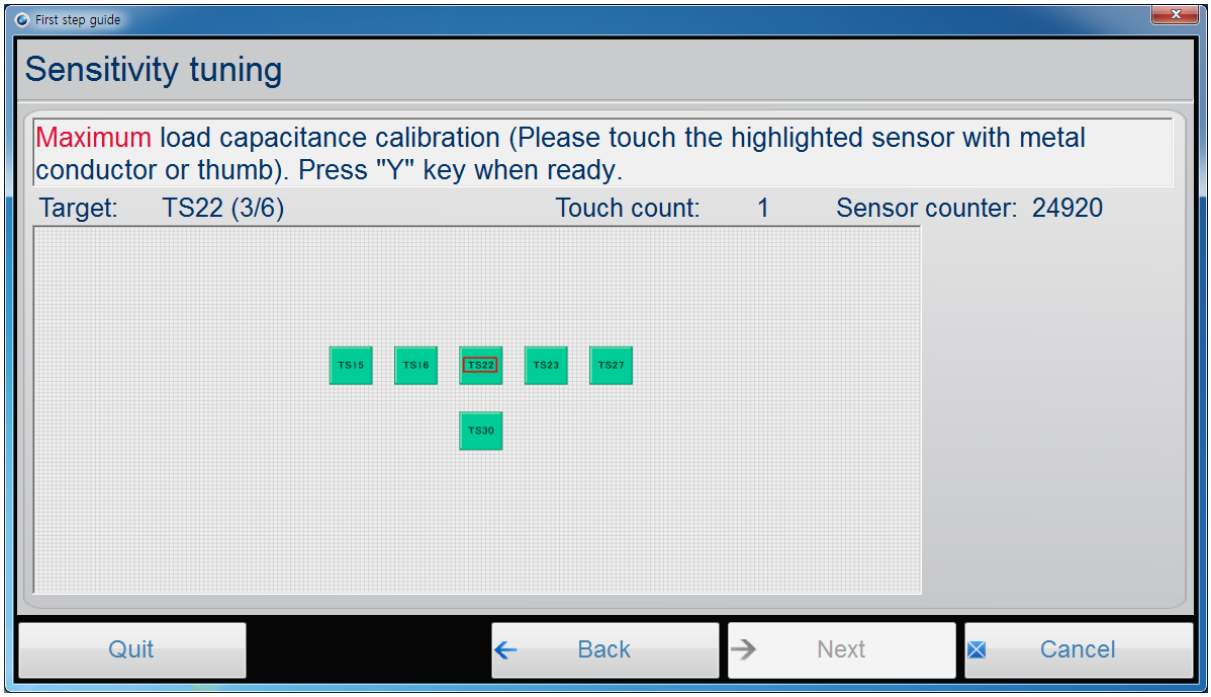


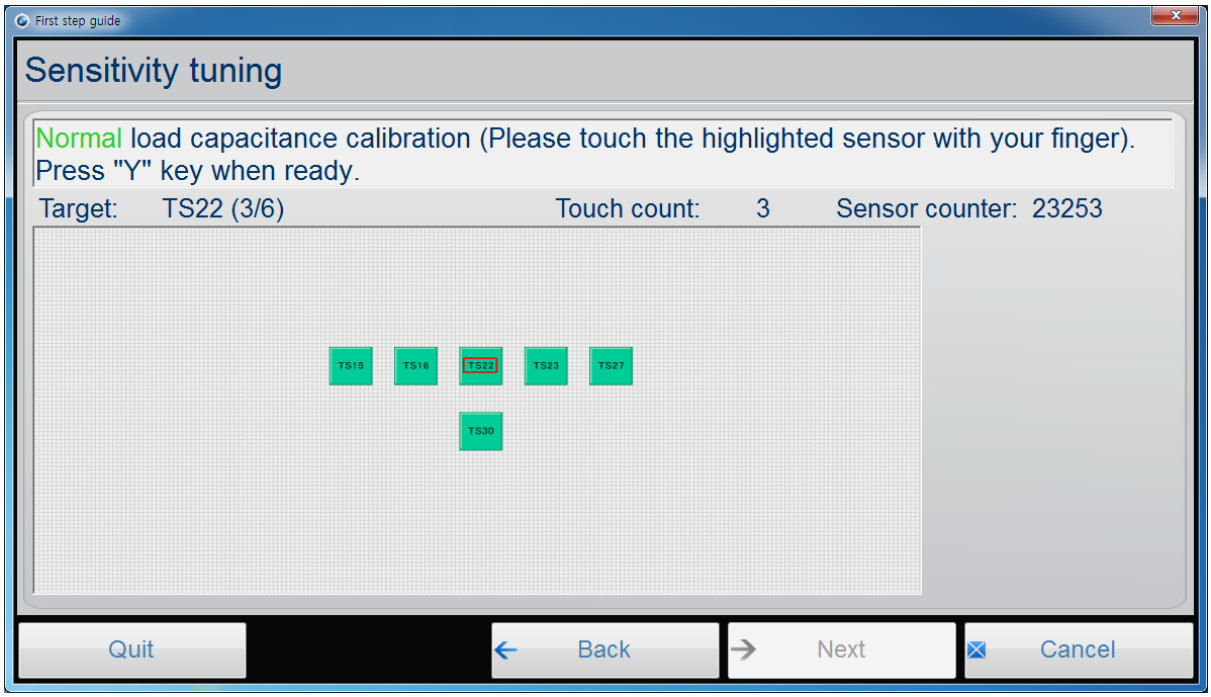


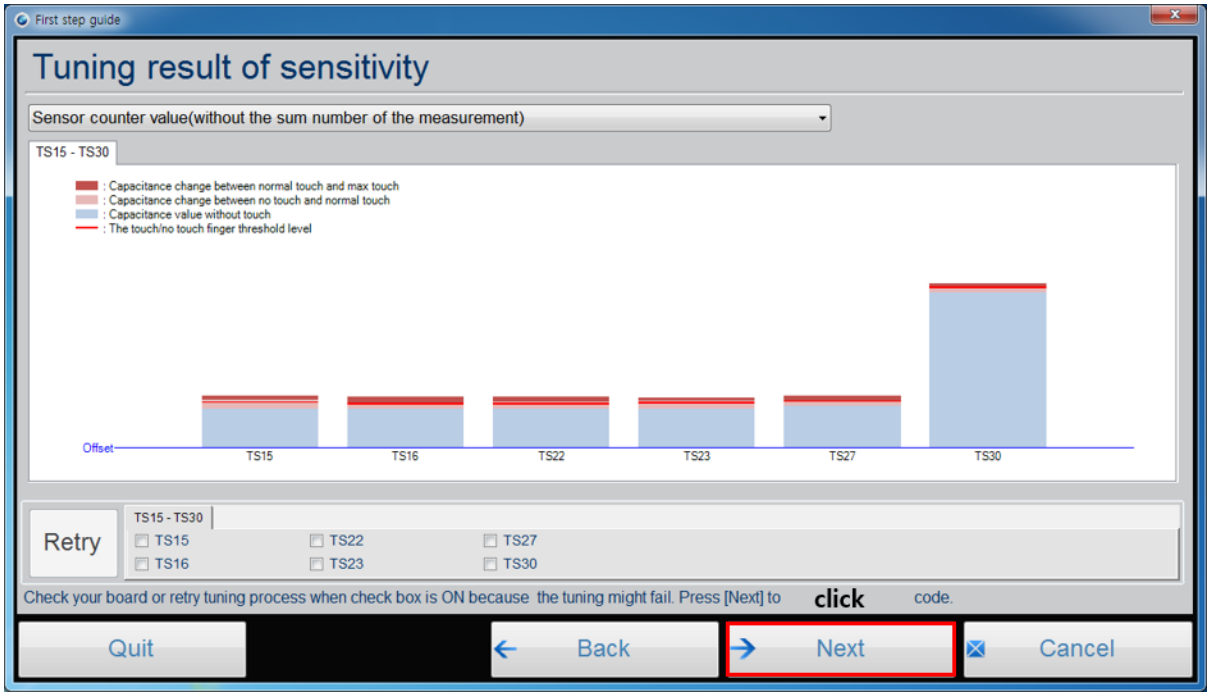


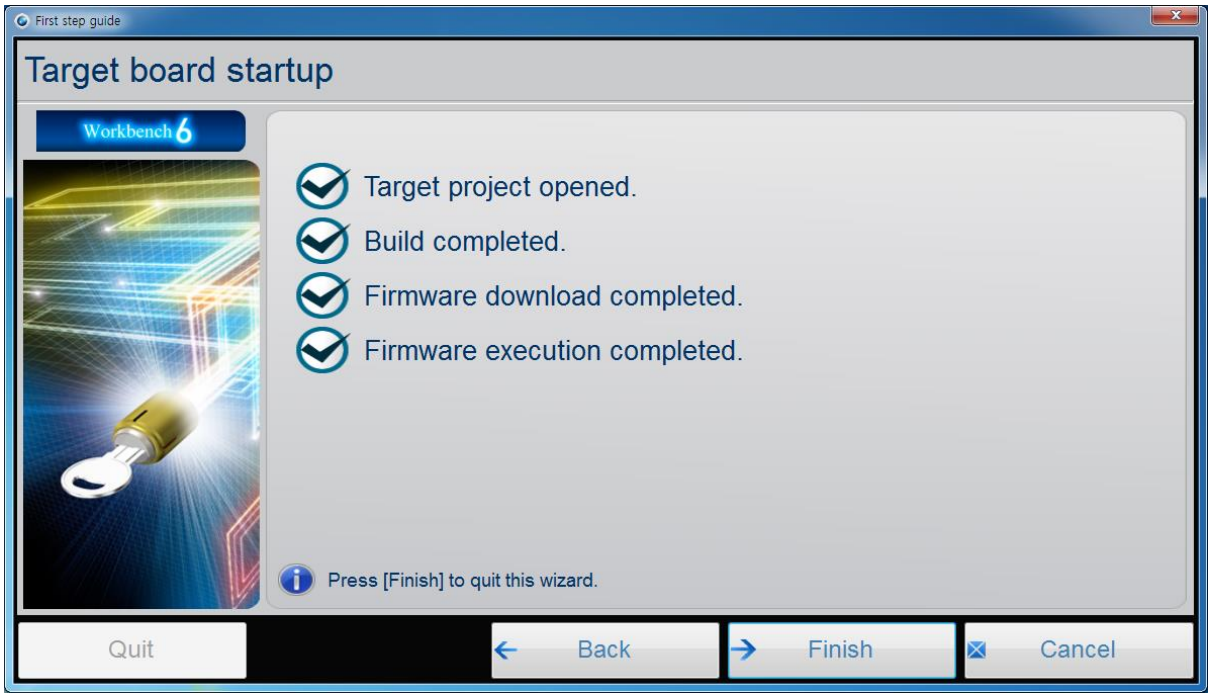
These values can be different


First step guide
Sensitivity tuning
Workbench 6
Press [Next] to start the sensitivity tuning.
click
Quit
Back
Next
Cancel


First step guide
Sensitivity tuning
Maximum load capacitance calibration (Please touch the highlighted sensor with metal conductor or thumb). Press "Y" key when ready.
Target: TS22 (3/6)
Touch count: 1
Sensor counter: 24920
TS15
TS16
TS22
TS23
TS27
TS30
Quit
Back
Next
Cancel


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
TS15
TS16
TS22
TS23
TS27
TS30
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 - TS30
TS15
TS16
TS22
TS23
TS27
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 Source Change

2.1 Register

First, let's look at the registers for LED control. For details, refer to RX231 User's Manual: Hardware.

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

The Port Direction Register (PDR) is responsible for setting the input and output of the port pins. The default value is 0. If set to 0, input will be set and if set to 1, output will be set.

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.ODR 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) is a register that can hold the output data of the port pin. If the setting of PMR and PDR registers is correct, when PODR is 1, it will output HIGH continuously.

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

	b7	b6	b5	b4	b3	b2	b1	b0
	B7	B6	B5	B4	B3	B2	B1	B0
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) is the register that receives the input data of the port pin. It is a read-only register and can read the current data of the pin irrespective of the settings of the PDR and PMR registers.

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

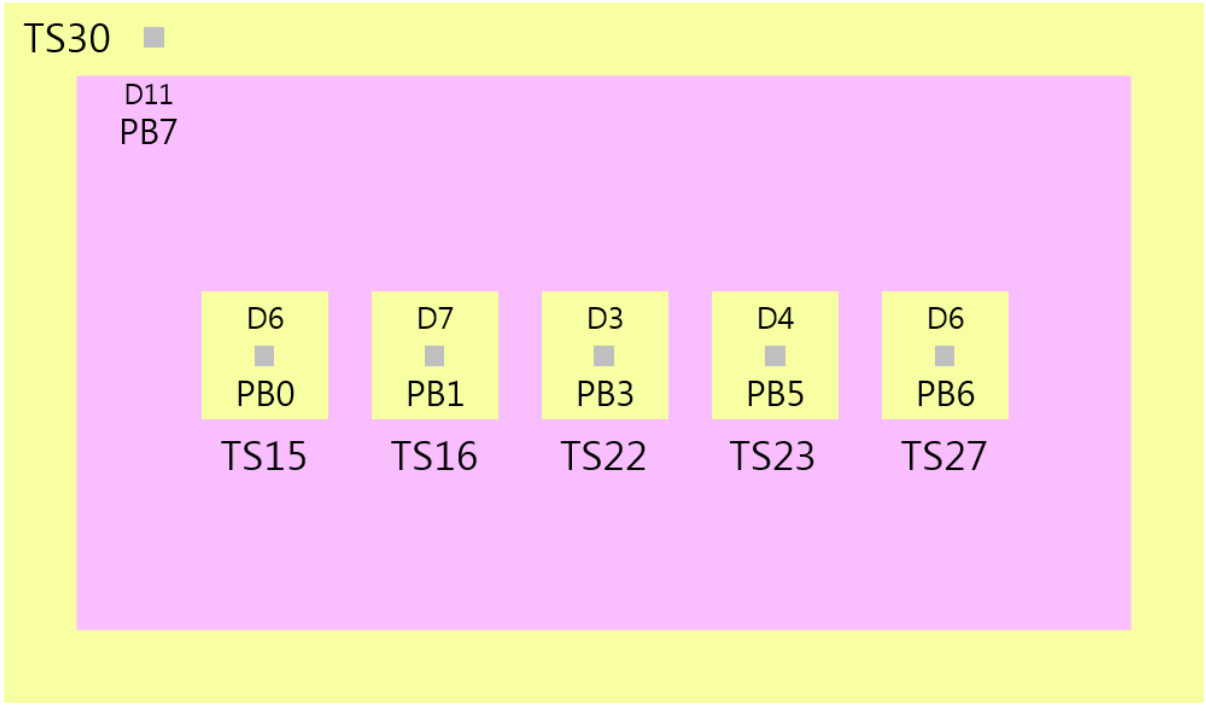
	b7	b6	b5	b4	b3	b2	b1	b0
	B7	B6	B5	B4	B3	B2	B1	B0
Value after reset:	0	0	0	0	0	0	0	0

Bit	Symbol	Bit Name	Description	R/W
b0	B0	Pm0 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

The Port Mode Register (PMR) defaults to 0, and a setting to 1 will cause the pin to be used for functions such as interrupt and communication, but setting to 0 enables control through the PODR register.

2.2 Source Code




```
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_result consists of button, slider, wheel. Only the button is explained.

The value of ts_result.button is [0 ~ 2] and the output of the value is as follows.

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

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

When TS15 is recognized as a touch, the value of the button[0] is 2^{15} = 32768

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

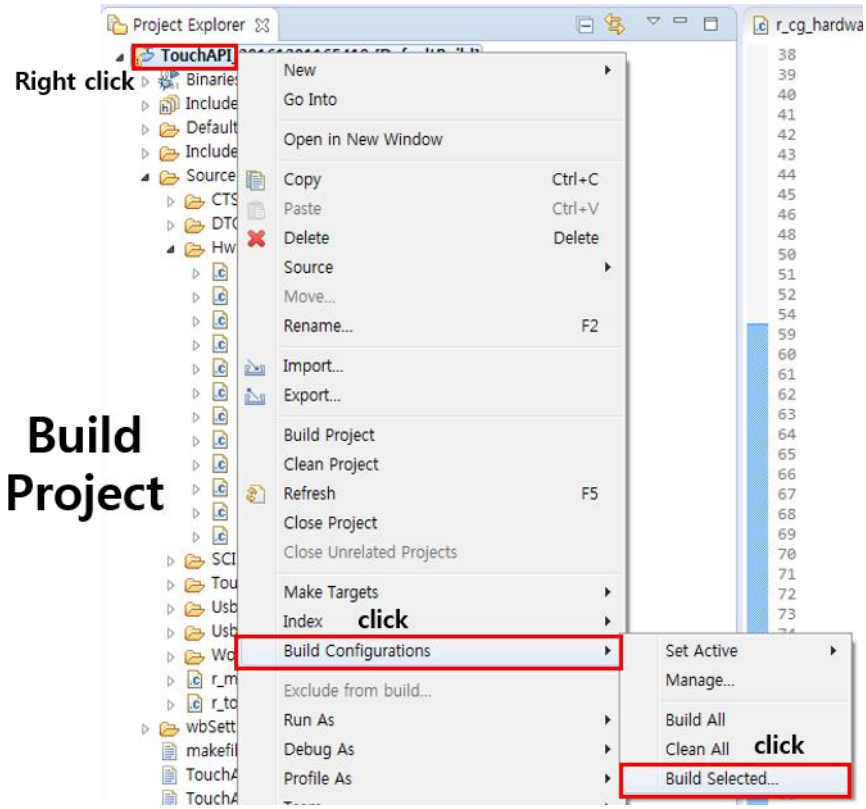
Expression	Type	Value
ts_result.button	uint16_t [3]	0x5d0 <ts_result>
(x)= ts_result.button[0]	uint16_t	32768
(x)= ts_result.button[1]	uint16_t	0
(x)= ts_result.button[2]	uint16_t	0

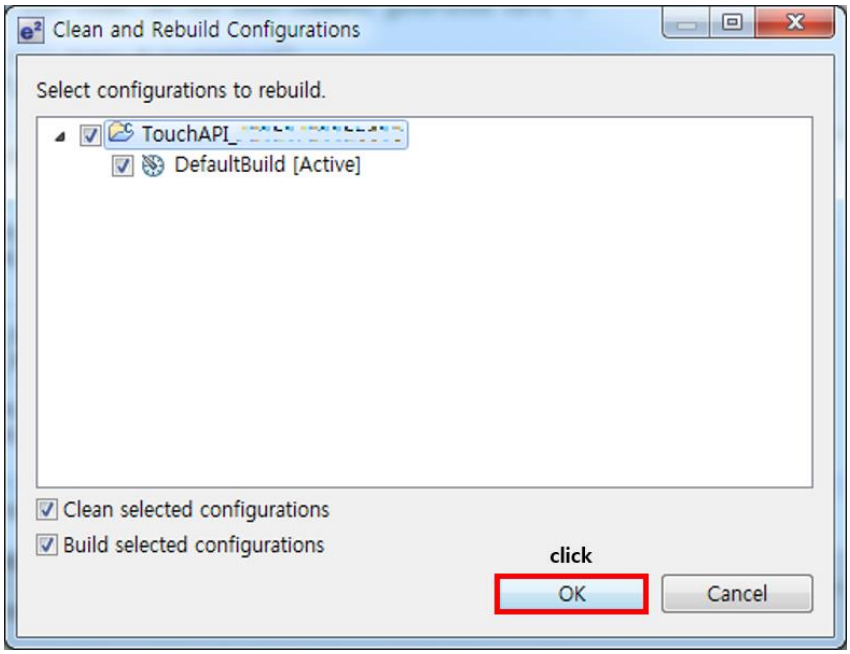
When TS27 is recognized as a touch, the value of the button[1] is 2^{11} = 2048

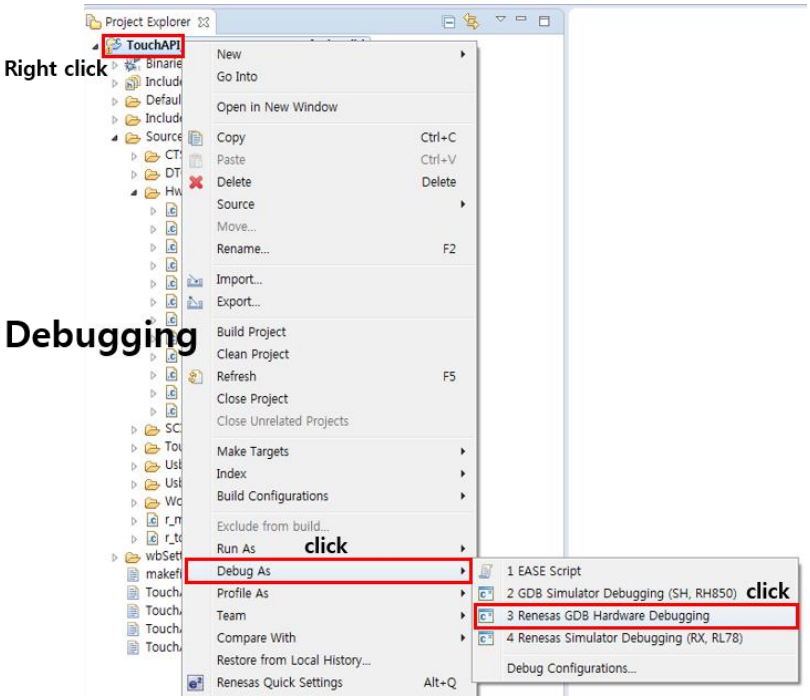
```
else if(1 == (ts_result.button[1] >> 11))    //TS27
{
    PORTB.PODR.BYTE = 0x
}
else if(1 == (ts_result.
{
    PORTB.PODR.BYTE = 0x
}
```

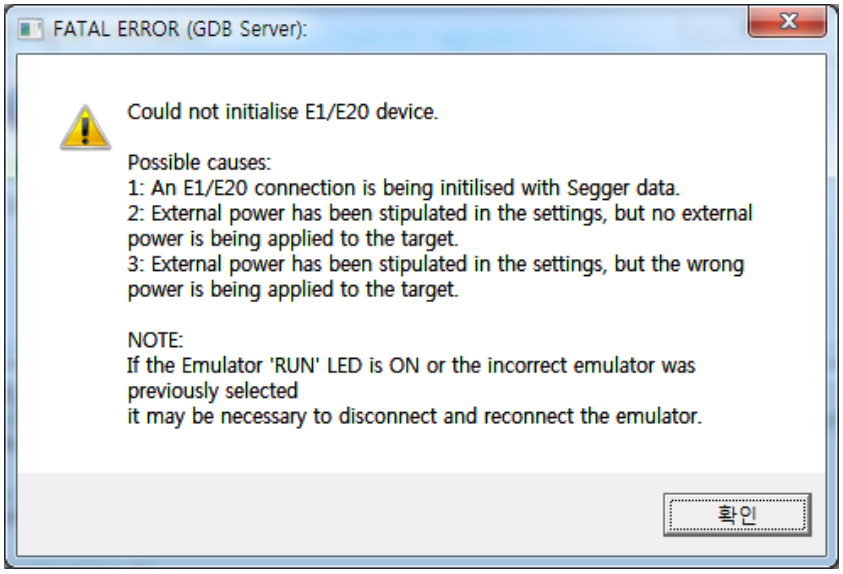
Expression	Type	Value
ts_result.button	uint16_t [3]	0x5d0 <ts_result>
(x)= ts_result.button[0]	uint16_t	0
(x)= ts_result.button[1]	uint16_t	2048
(x)= ts_result.button[2]	uint16_t	0

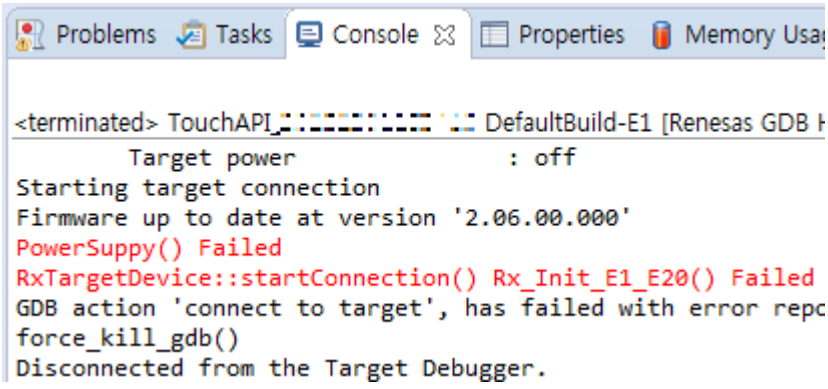
3 Project Writing



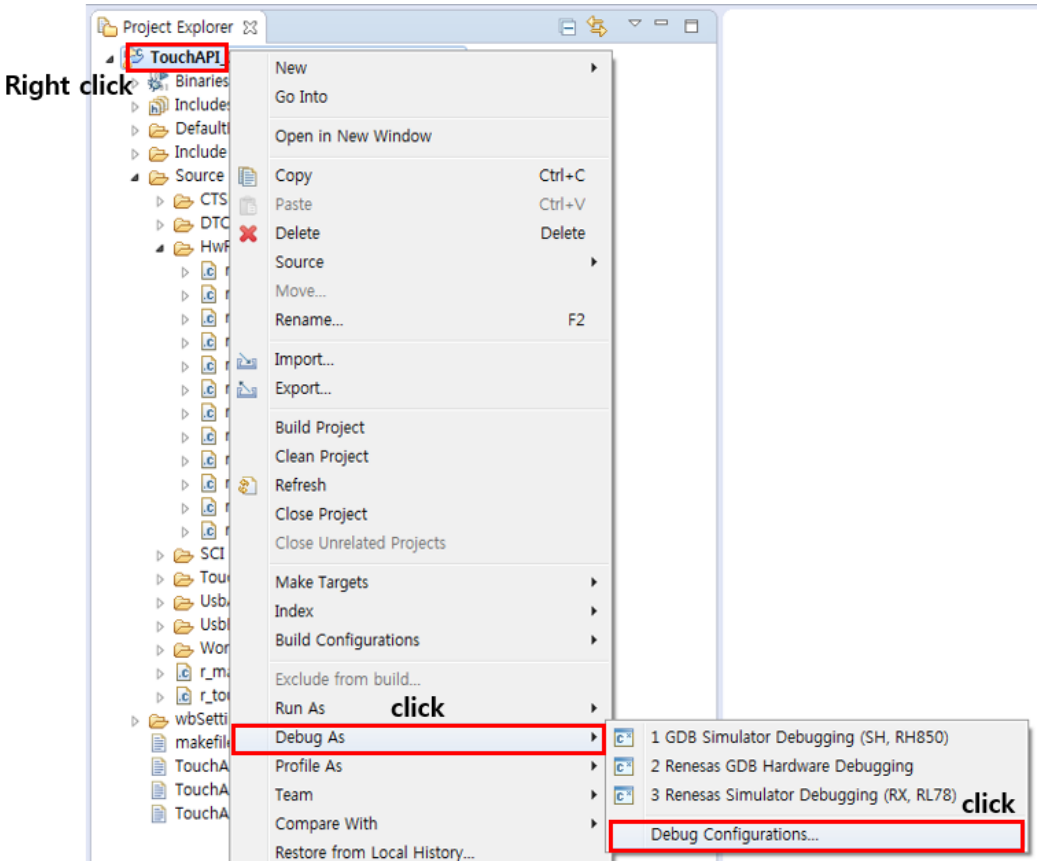


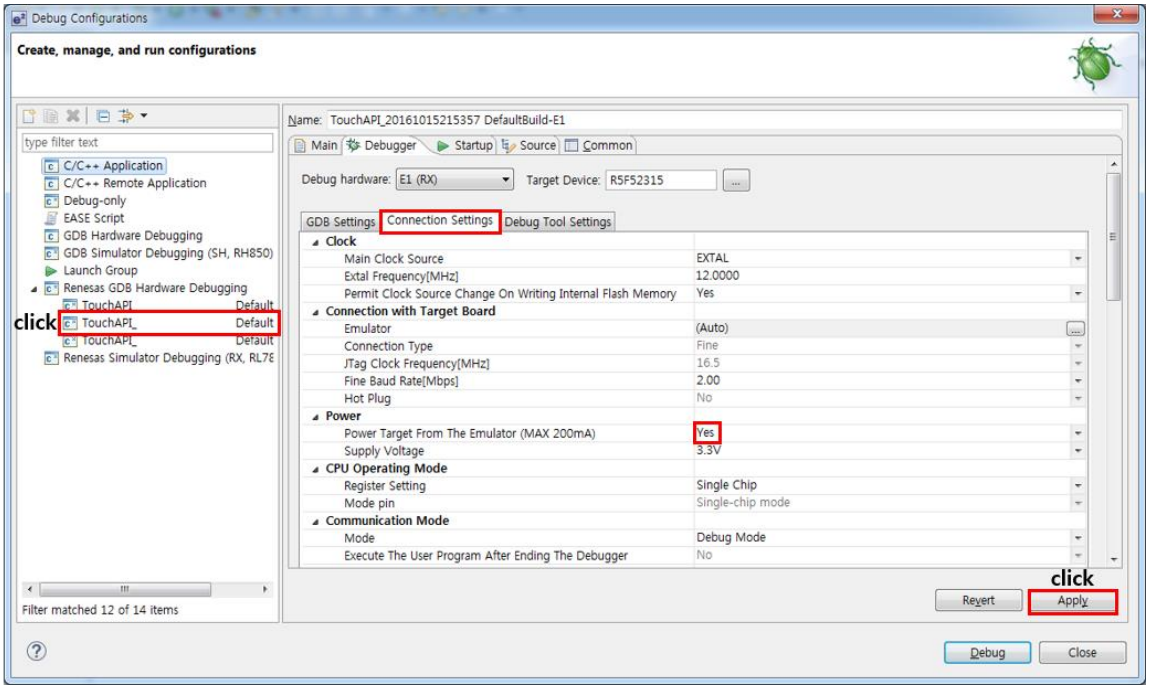






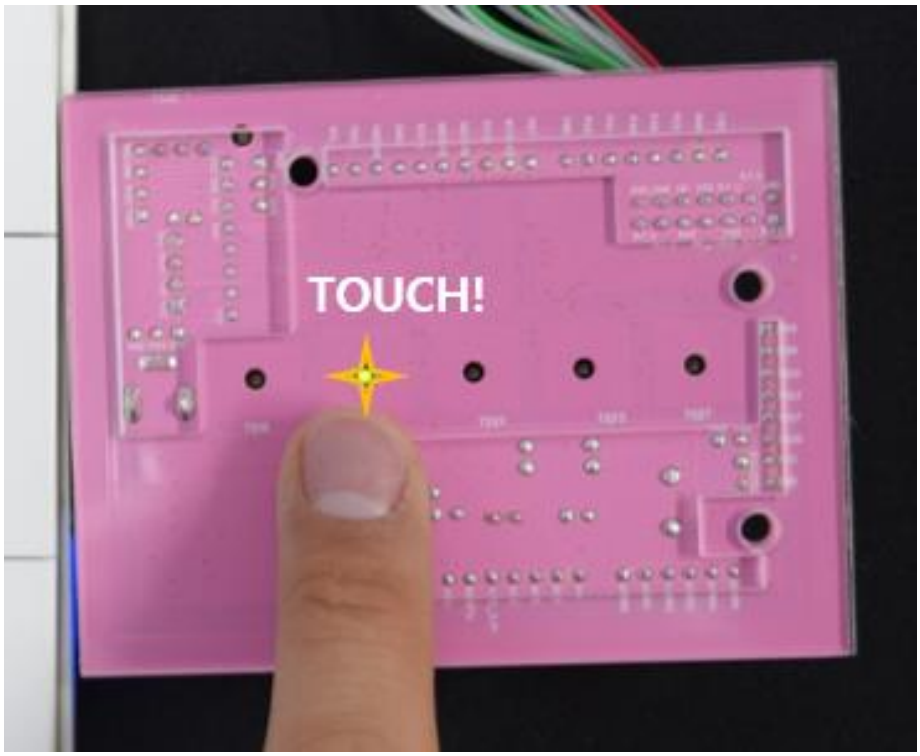
If the above error message appears during debugging, please refer to the setting below.





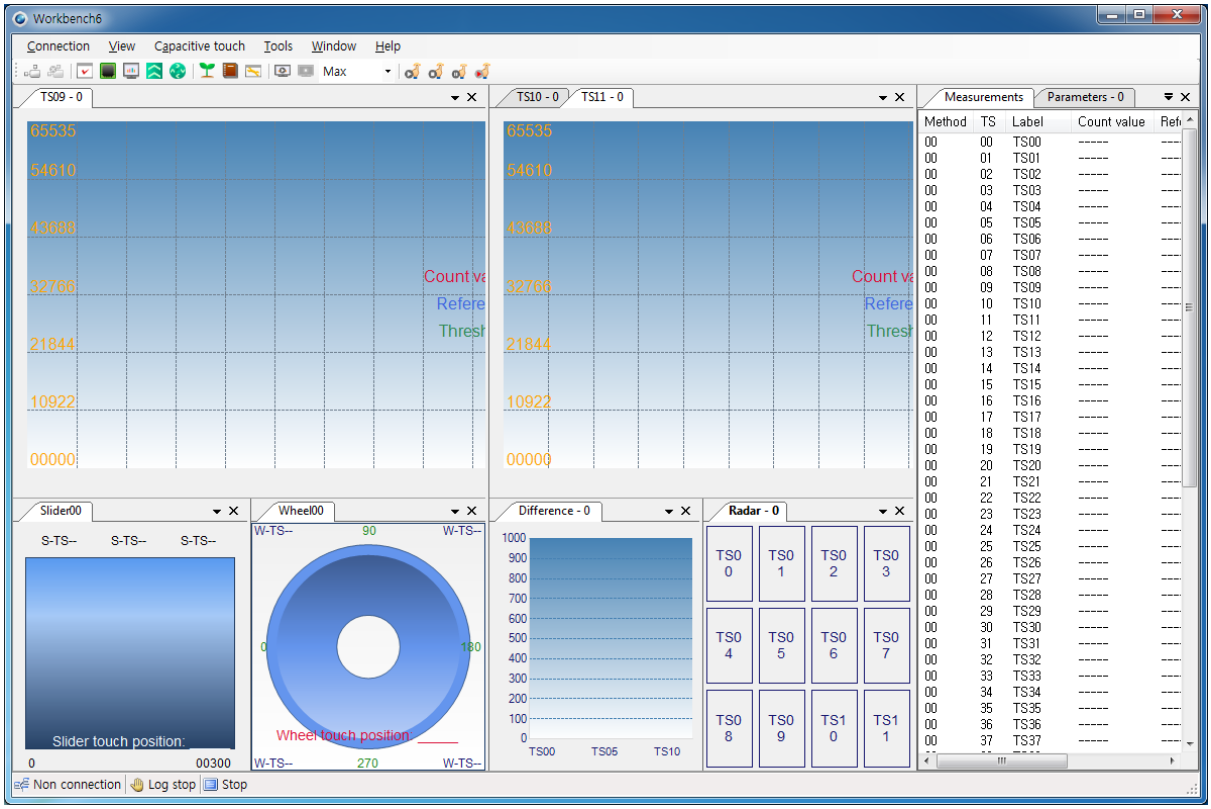
Immediately after the project is created, the option to supply power through the emulator is set to NO as shown above. If you change to Yes and proceed, you will be able to proceed without the need for additional power (adapter, micro USB, etc.).

4 Execution

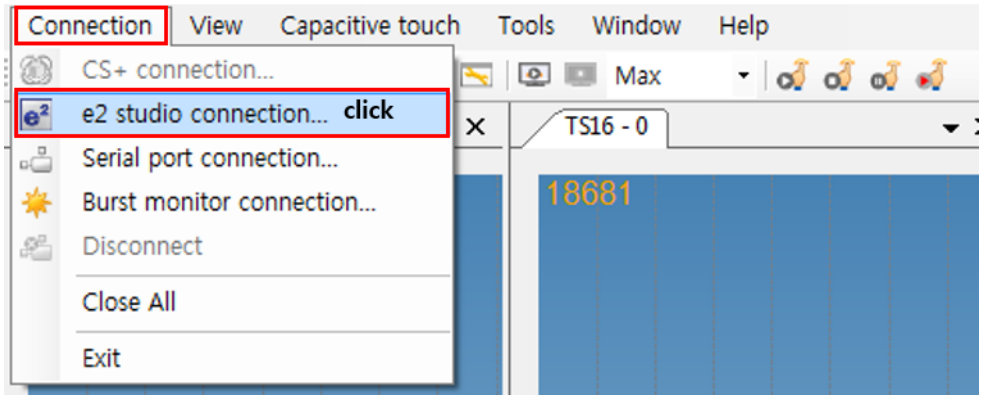


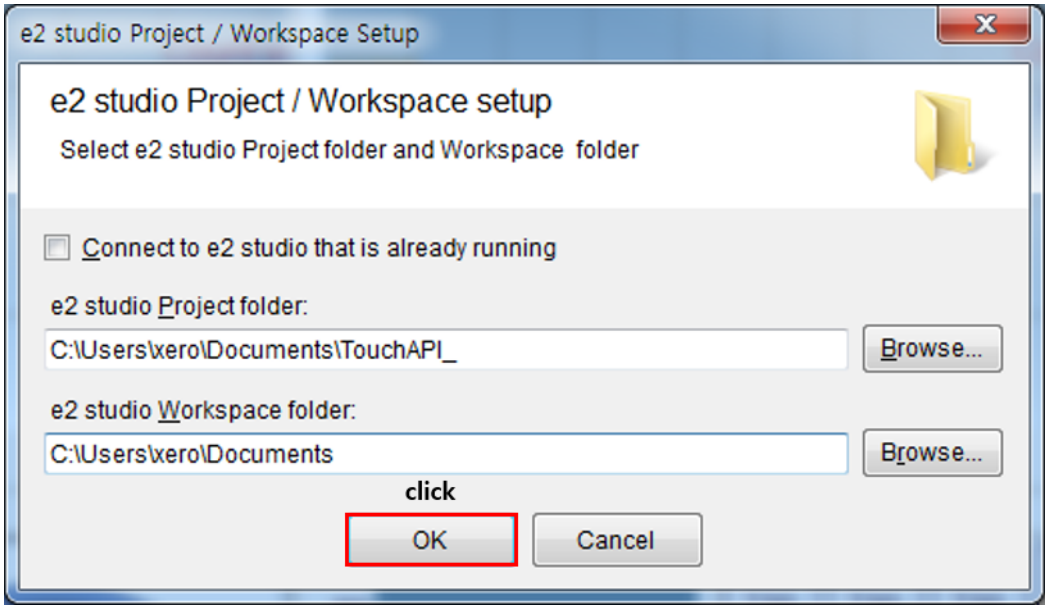
When you press each of the 5 buttons, the corresponding LED will light up. If you put your finger 0.5 ~ 1cm near to the proximity sensor TS30, the LED will light up.

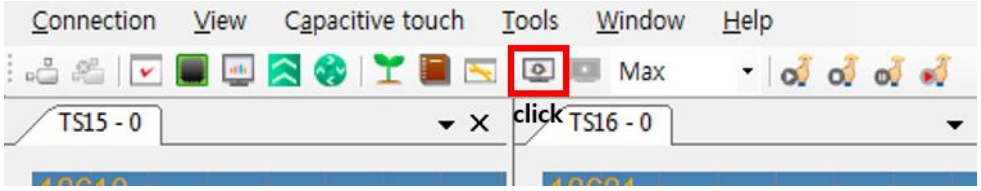
5 Workbench6 tuning

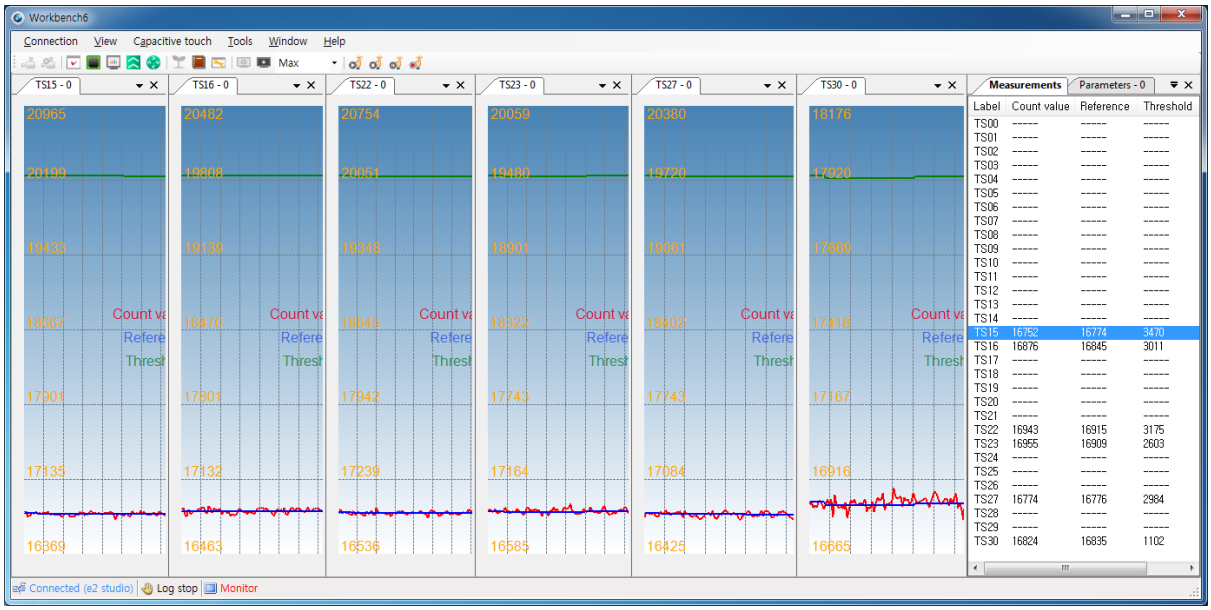


click


Connection
View
Capacitive touch
Tools
Window
Help
CS+ connection...
e2 studio connection...
click
Serial port connection...
Burst monitor connection...
Disconnect
Close All
Exit
Max
TS16 - 0
18681


e2 studio Project / Workspace Setup
e2 studio Project / Workspace setup
Select e2 studio Project folder and Workspace folder
Connect to e2 studio that is already running
e2 studio Project folder:
C:\Users\xero\Documents\TouchAPI_
Browse...
e2 studio Workspace folder:
C:\Users\xero\Documents
Browse...
click
OK
Cancel


Connection
View
Capacitive touch
Tools
Window
Help
Max
TS15 - 0
click
TS16 - 0

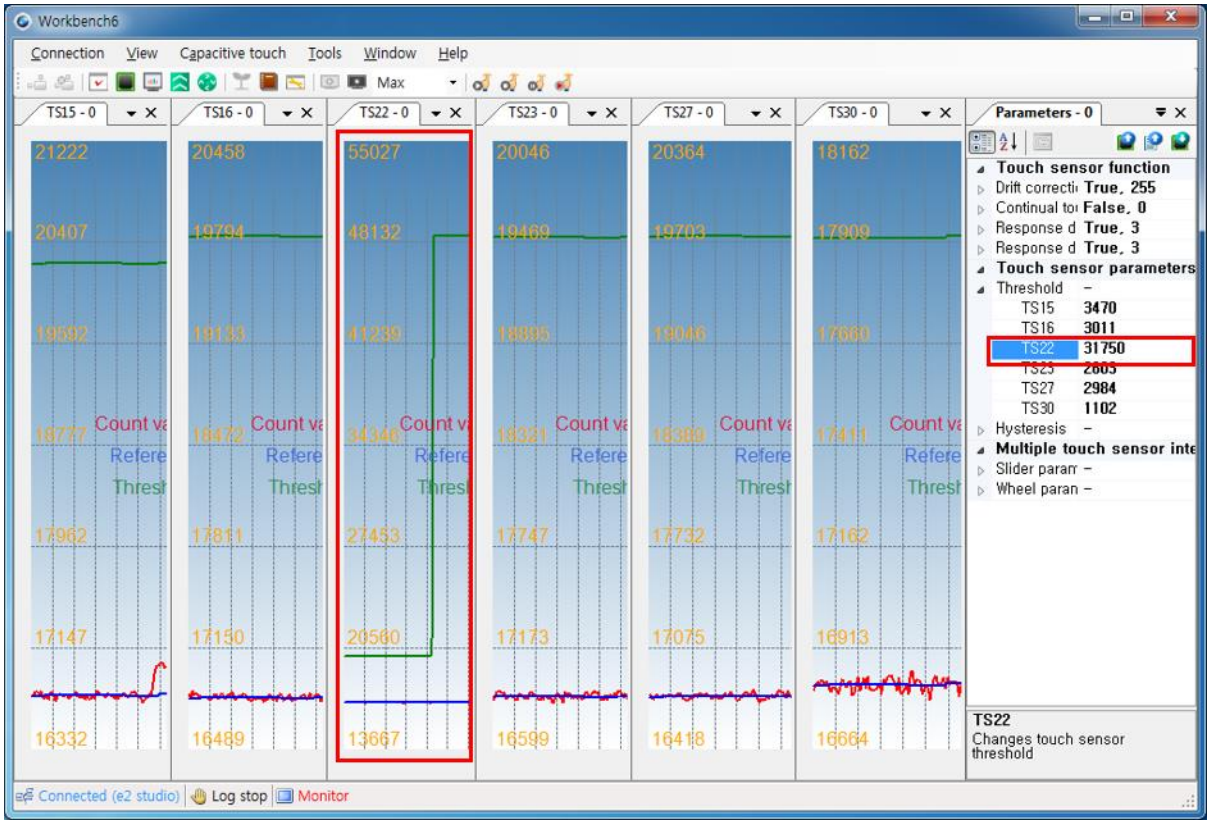


Count Value : Continually changing touch sensor value

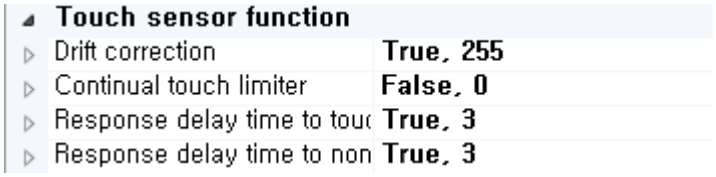
Reference Value : Average of Count value

Threshold Value : Touch recognition range value

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



Threshold values can be changed on the Parameters tab as follows.



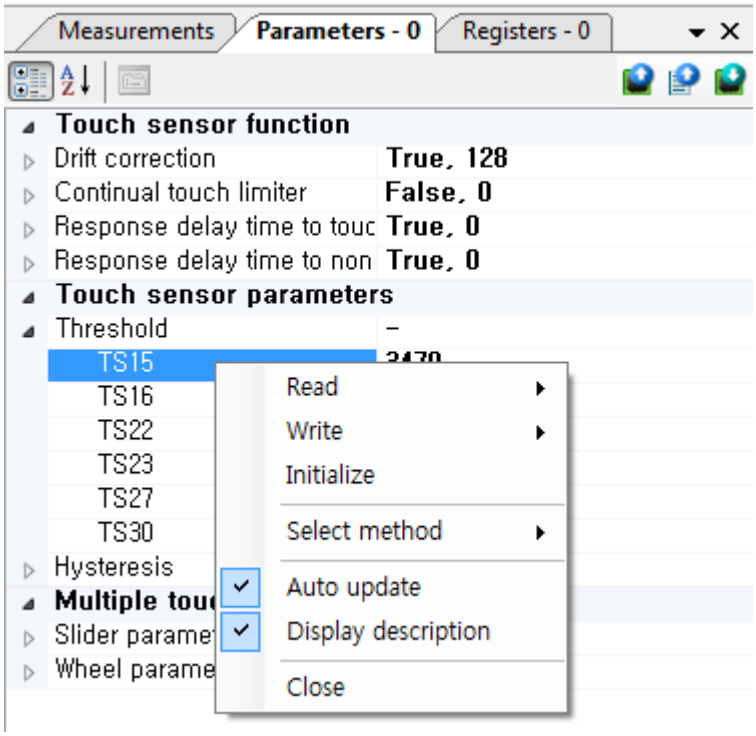
In the touch sensor function, you can specify the calculation speed of the reference value, and the touch response time.

The reference value is calculated by averaging the count value every 4 seconds.

The reference value can be calculated quickly by adjusting the Drift Correction part.

≒ Drift Correction Value * 0.016 [Sec] (range : 0 ~ 255)

Touch response time is set to 3 by default, but it is recommended to set it to 0.



If you right click on the Parameters tab, then above window will appear.

Read	Read From target system	Read Parameter value from connected board
	Read from parameter file	Read parameter file saved on PC
Write	Write to target system	Save to connected board
	Write to parameter file	Save to parameter file
	Write to Touch API	Save to project
Initialize		Reset to initial value before change