SAMD21 ARM Cortex-M0+ Microcontroller & Harmony v3

SAM2002ADV v1.01

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

CAE Taiwan Team
Microchip Technology Inc.

April 2024

SAM series course history and brief

- **SAM1001 – MCU32 Peripheral Library (PLIB)**
  - (2017~2018)
    - Peripheral Library (PLIB) on APP045v2.0 (SAMD21 Arm Cortex-M0+)
    - Advance Software Framework (ASF) v3.x + GNU C compiler + Microchip Studio 7
  - (2019~2022)
    - Peripheral Library (PLIB) on APP045v2.0/v4.x (SAMD21 Arm Cortex-M0+)
    - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
  - (2023~2024)
    - Peripheral Library (PLIB) on APP045v4.x/v5.x (SAMD21 Arm Cortex-M0+)
    - MCC Harmony + XC32 Compiler + MPLAB X IDE
- **SAM2001ADV – MCU32 Advanced Peripheral Library (PLIB)**
  - (2021~2022)
    - Peripheral Library (PLIB) Advanced modules on APP045v4.x (SAMD21 Arm Cortex-M0+)
    - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
  - (2023~2024)
    - Peripheral Library (PLIB) Advanced modules on APP045v4.x/v5.x (SAMD21 Arm Cortex-M0+)
    - MCC Harmony + XC32 Compiler + MPLAB X IDE
- **SAM2002 – MCU32 System Service and Driver**
  - (2019~2022)
    - System Service and Driver on APP045v4.x (SAMD21 Arm Cortex-M0+)
    - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
  - (2023~2024)
    - System Service and Driver on APP045v4.x/v5.x (SAMD21 Arm Cortex-M0+)
    - MCC Harmony + XC32 Compiler + MPLAB X IDE
- **SAM2002ADV – MCU32 Touch and USB Device**
  - (2021~2022)
    - QTouch Library and USB Device Driver on APP045v4.x (SAMD21 Arm Cortex-M0+)
    - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
  - (2023~2024)
    - QTouch Library and USB Device Driver on APP045v4.x (SAMD21 Arm Cortex-M0+)
    - MCC Harmony + XC32 Compiler + MPLAB X IDE

Background Knowledge and Preparation

- **SAM2002ADV is for advanced users, then we will not introduce :**
  - MPALB X IDE and Harmony Configuration. (Please study from SAM2001 course)
  - How to download(prepare) latest Harmony Framework. (Please study from SAM2001 course)
  - How to create a new project. (Please study from SAM2001 course)
  - PLIB (Peripheral Library) implementation. (Please study from SAM2001 and SAM2001ADV course)
  - System Service and Driver implementation. (Please study from SAM2002 course)

- **SAM2002ADV will use below tool versions for Labs.**
  - MPLAB® X IDE v6.15 https://www.microchip.com/mplab/mplab-x-ide
  - MPLAB® XC32 v4.35 https://www.microchip.com/mplab/compilers
  - MPLAB® Code Configurator v5.4.1 (Install in MPLAB X IDE Plug-in)
  - SAMD21 DFP v3.6.144 (Device Family Pack download from X IDE \Tools\Packs)
  - CMSIS v5.8.0 (Common Microcontroller Software Interface Standard)
  - MCC Core v5.6.1 (Install with MPLAB X IDE v6.15)
  - MPLAB® MCC Harmony Framework (From X IDE MCC Harmony Content Manager)
    - Required Content (Must install)
      - csp 3.18.4
      - harmony-services 1.3.2
      - CMSIS_5 5.9.0
      - quick_docs 1.6.0
    - Optional Content (Manual select in Content Manager)
      - touch v3.15.0
      - usb v3.12.0
      - dev_packs v3.18.1
      - core v3.13.3

Advance Course

- Introduction to QTouch® Peripheral Touch Controller (PTC)
- Harmony Touch Configurator
- APP045 EVB QTouch® Design and Pattern
  - Lab1 Touch Buttons
  - Lab2 Touch Slider
  - Lab3 Data Visualizer for Touch
- USB Fundamentals
- USB Device Library (USB Device Architecture)
- Harmony USB Device - Device Layer Configurator
  - Lab4 USB Device - Device Layer Initialize
- USB Device CDC Class
  - Lab5 USB Device CDC Class - Basic TX and RX
- USB Device HID Class
  - Lab6 USB Device HID Class - Basic TX and RX

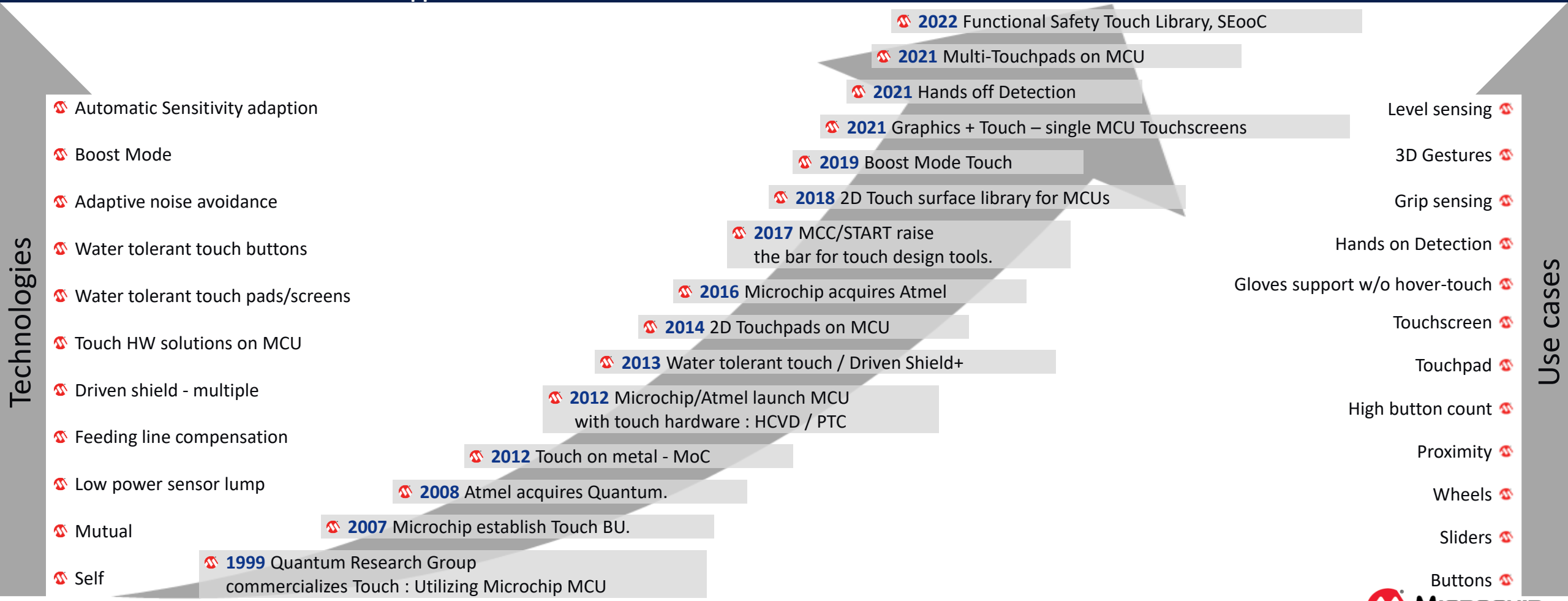
Introduction to QTouch® Peripheral Touch Controller (PTC)

Introduction to QTouch® Peripheral Touch Controller (PTC)

Long-Term innovative Player in Touch

Microchip serves **all Markets and all use cases** for touch.

Consumer | **Appliances** | **Medical** | **Industrial** | **Automotive** | **Wearables** | **IoT** | **Avionic**

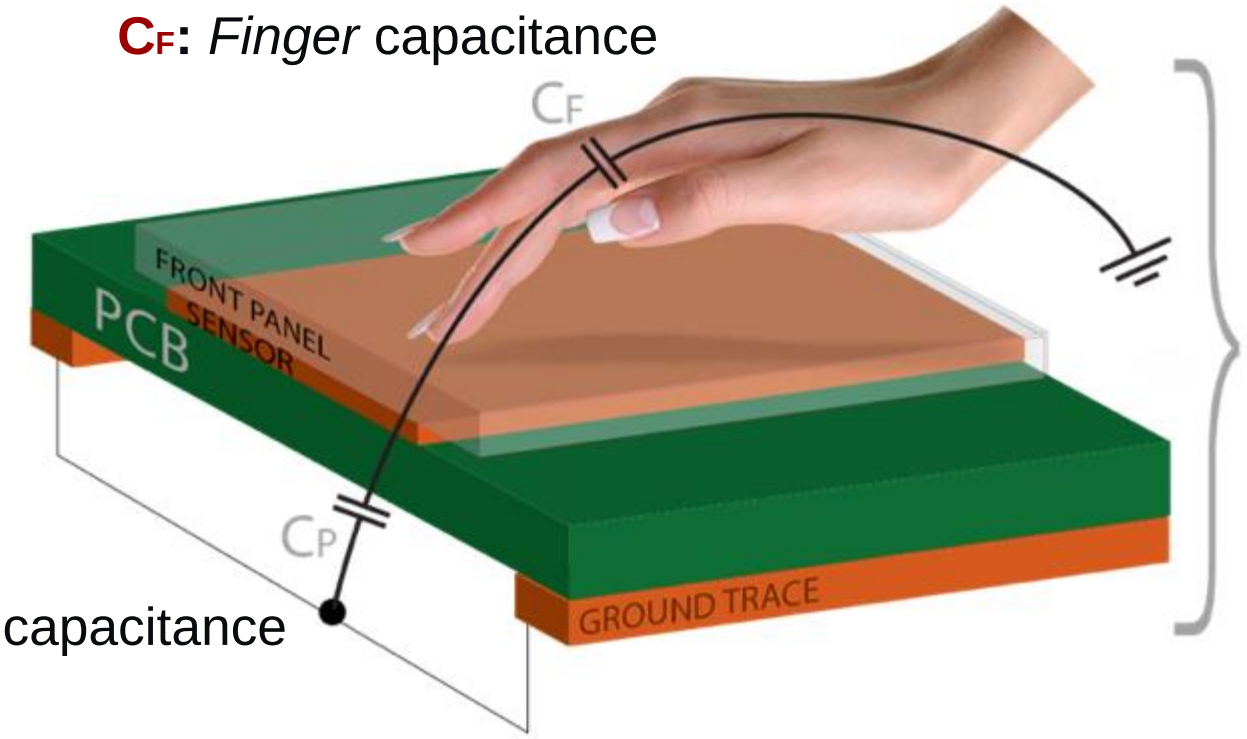


MICROCHIP

Introduction to QTouch® Peripheral Touch Controller (PTC)

Capacitive Touch Sensor

C_F: *Finger* capacitance



C_S: total *Sensor* capacitance

C_P: *Parasitic* capacitance

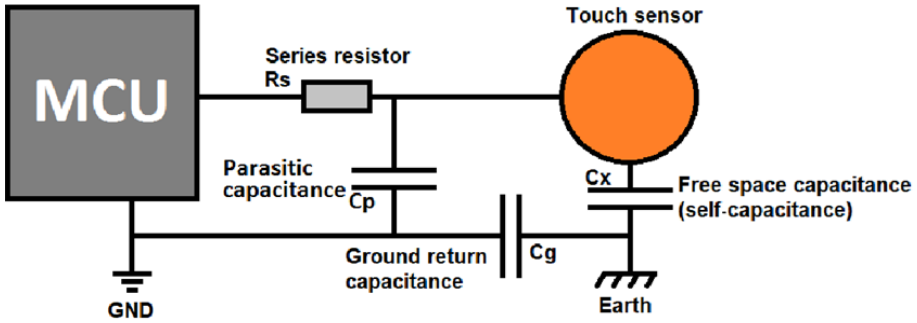
Sensor Capacitance (C_S) = C_P + C_F

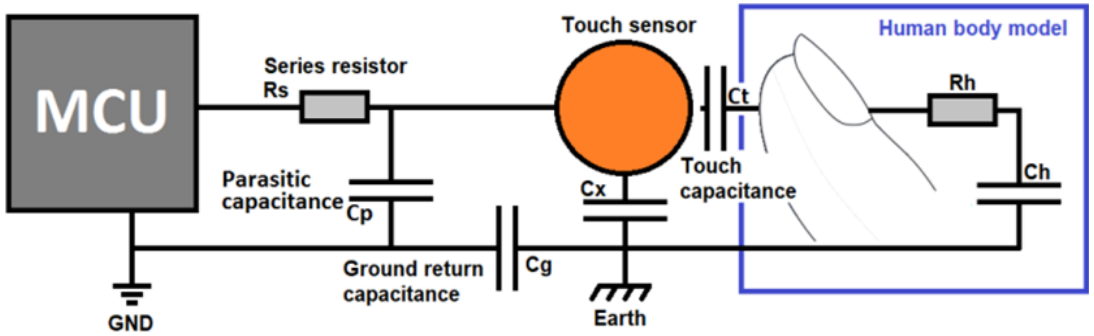
MICROCHIP

Introduction to QTouch® Peripheral Touch Controller (PTC)

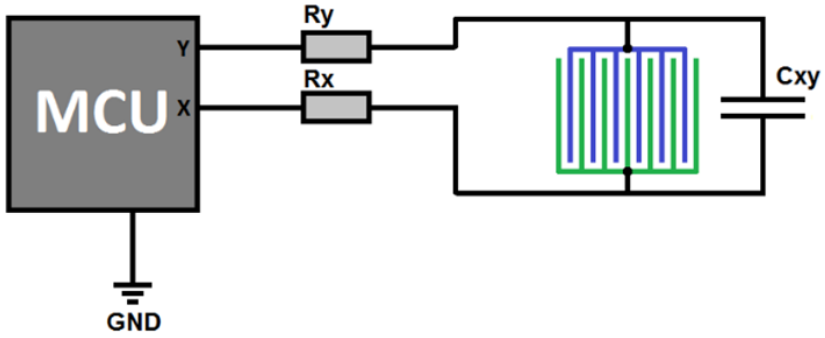
Capacitance Touch Measurement

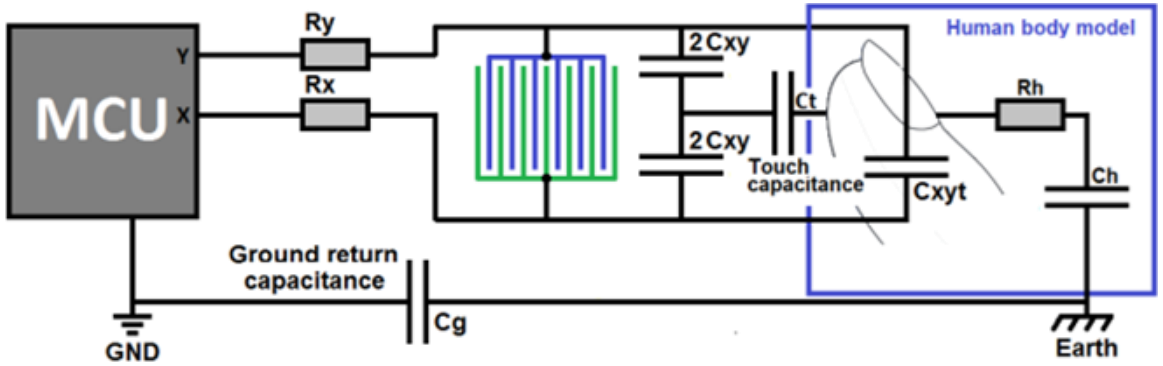
Self Capacitance Touch





Mutual Capacitance Touch



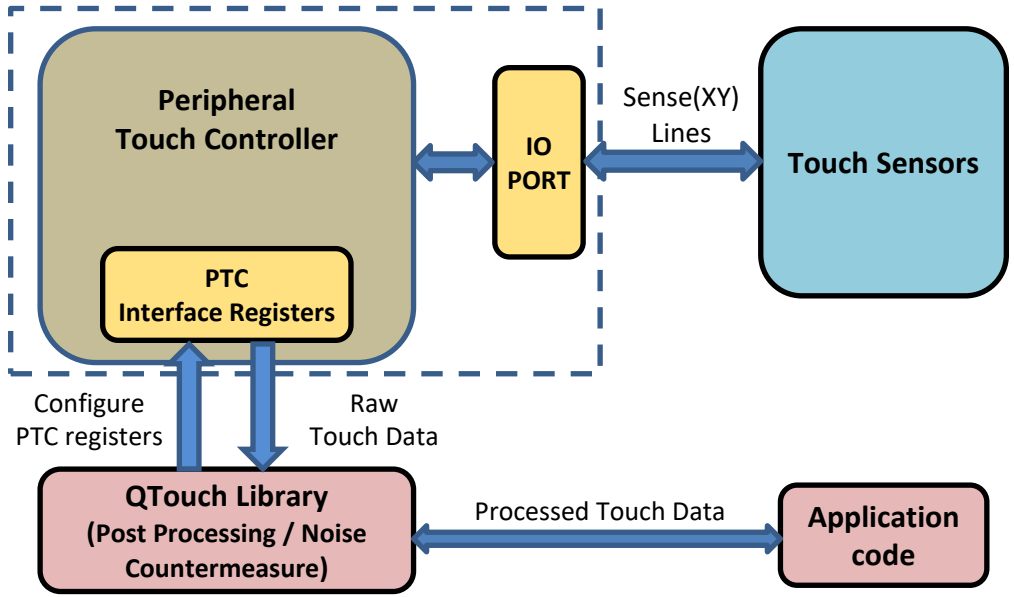


MICROCHIP

Introduction to QTouch® Peripheral Touch Controller (PTC)

Peripheral Touch Controller (PTC)

- The Peripheral Touch Controller (PTC) acquires signals in order to detect touch on capacitive sensors.
- The external capacitive touch sensor is typically formed on a PCB, and the sensor electrodes are connected to the analog front end of the PTC through the I/O pins in the device.
- The PTC supports both self- and mutual-capacitance sensors.
- In the mutual-capacitance mode, sensing is done using capacitive touch matrices in various X-Y configurations, including indium tin oxide (ITO) sensor grids. The PTC requires one pin per X-line and one pin per Y-line.
- In the self-capacitance mode, the PTC requires only one pin (Y-line) for each touch sensor. It also supports a Shield Driver to provide a driven shield on electrodes for better noise immunity and moisture tolerance.

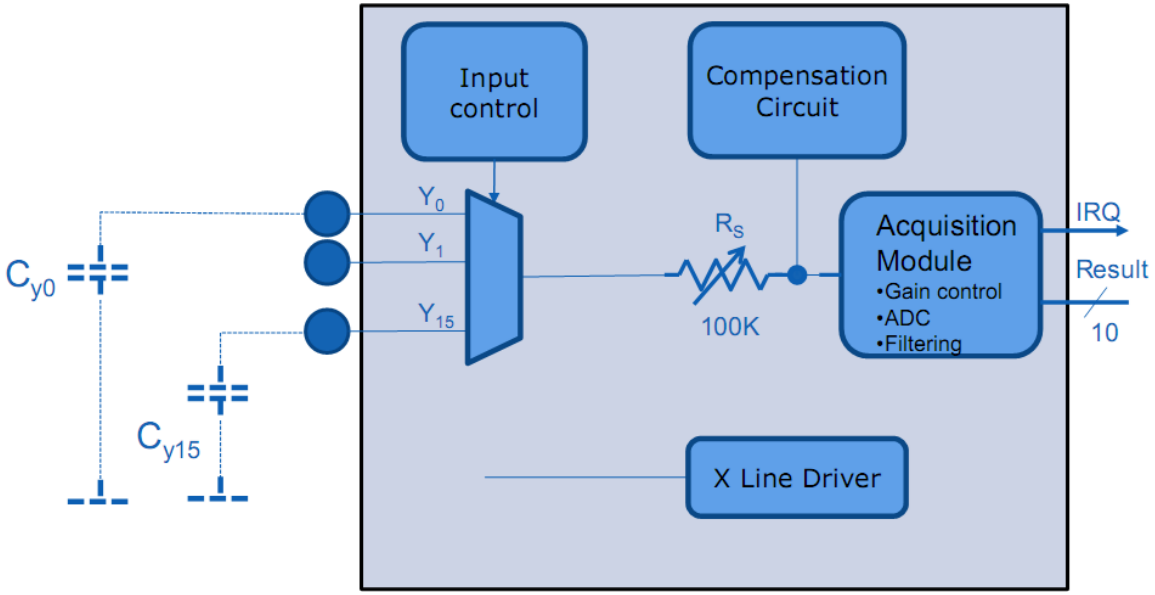


MICROCHIP

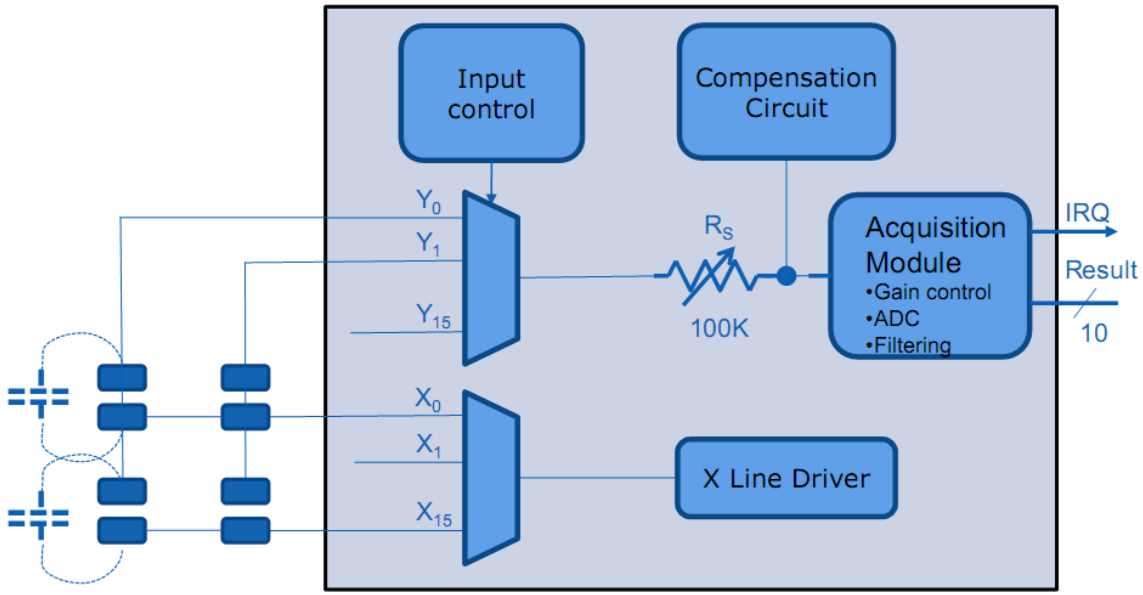
Introduction to QTouch® Peripheral Touch Controller (PTC)

PTC Block Diagram

Self Capacitance Touch

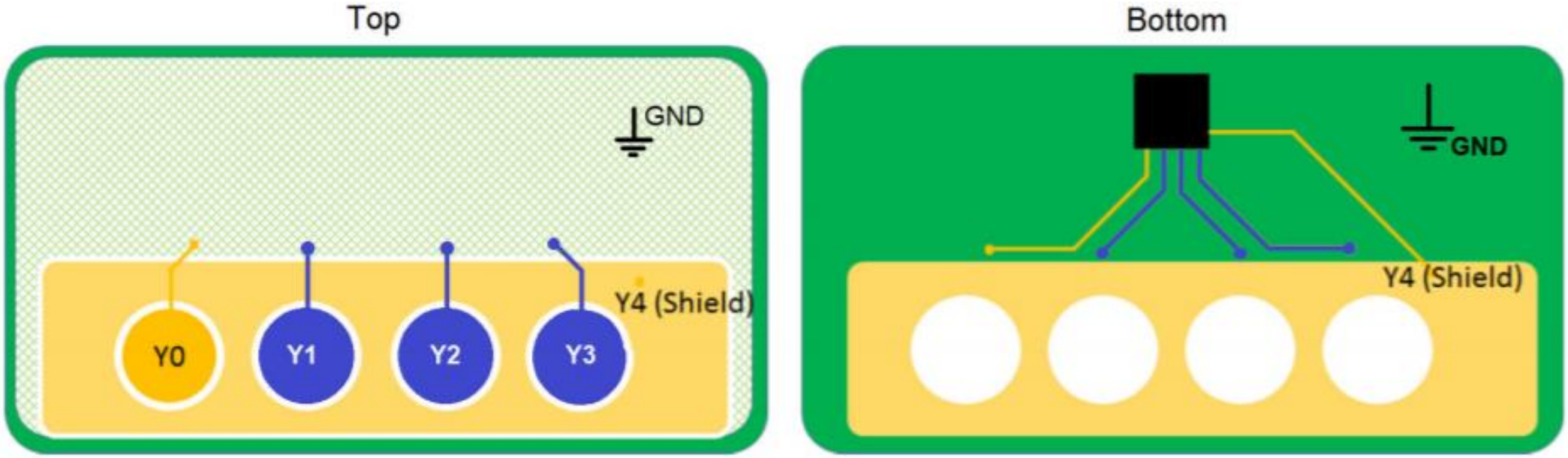


Mutual Capacitance Touch

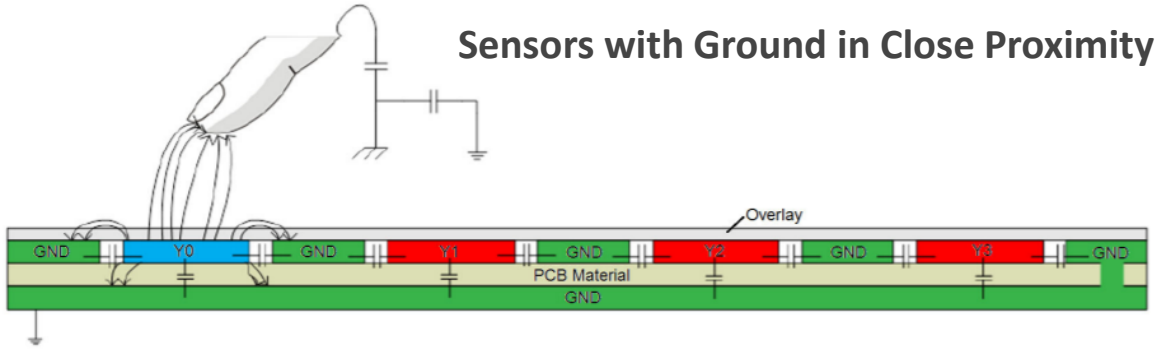


Introduction to QTouch® Peripheral Touch Controller (PTC)

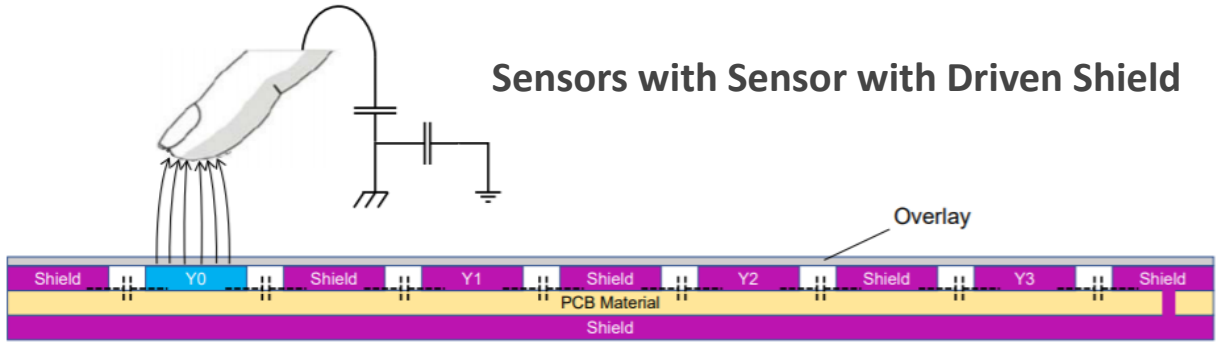
Driven Shield



Sensors with Ground in Close Proximity



Sensors with Sensor with Driven Shield



https://microchipdeveloper.com/touch:guide-for-ptc-driven-shield-design

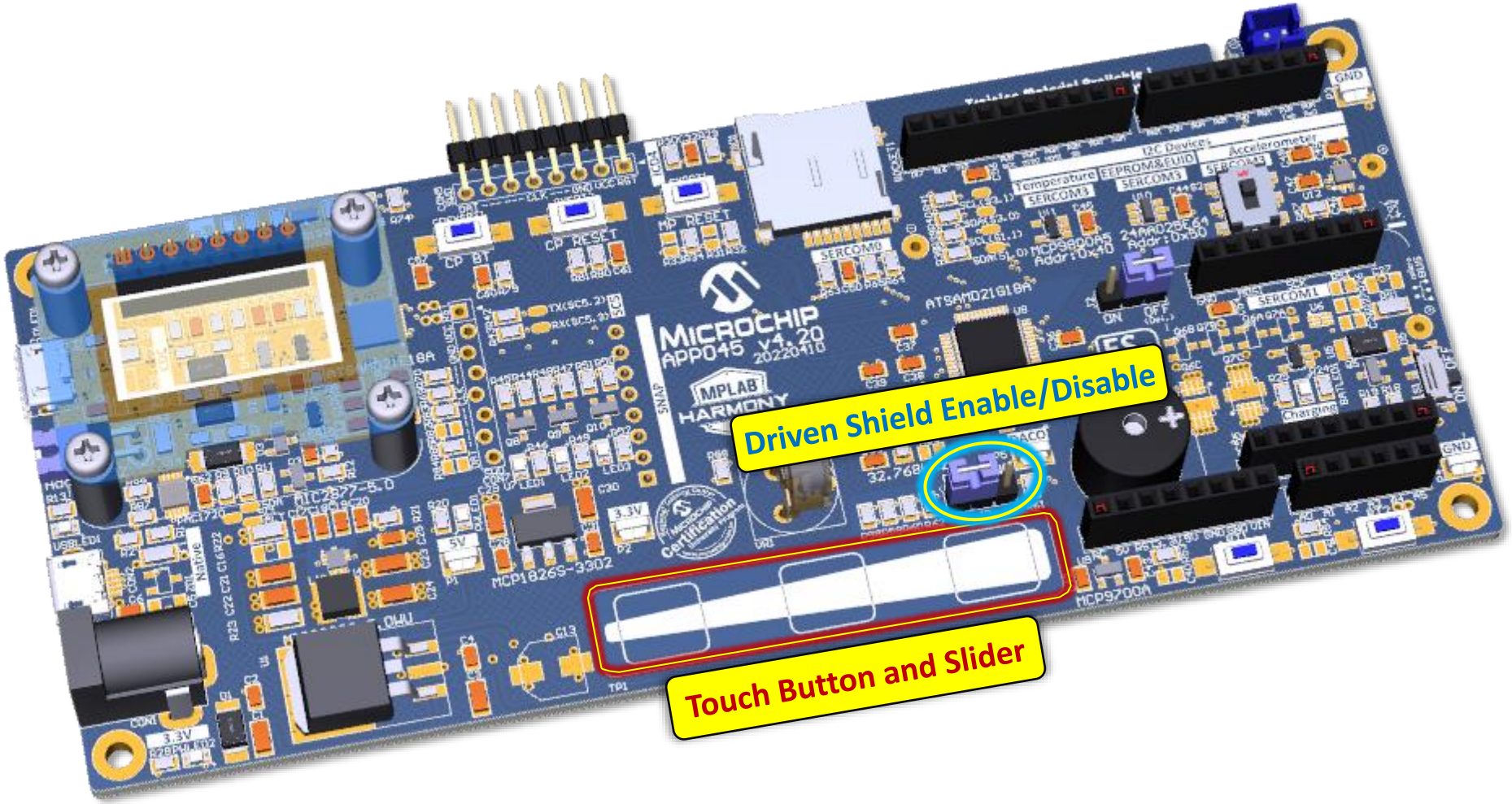
MICROCHIP

APP045 EVB QTouch® Design and Pattern

MICROCHIP

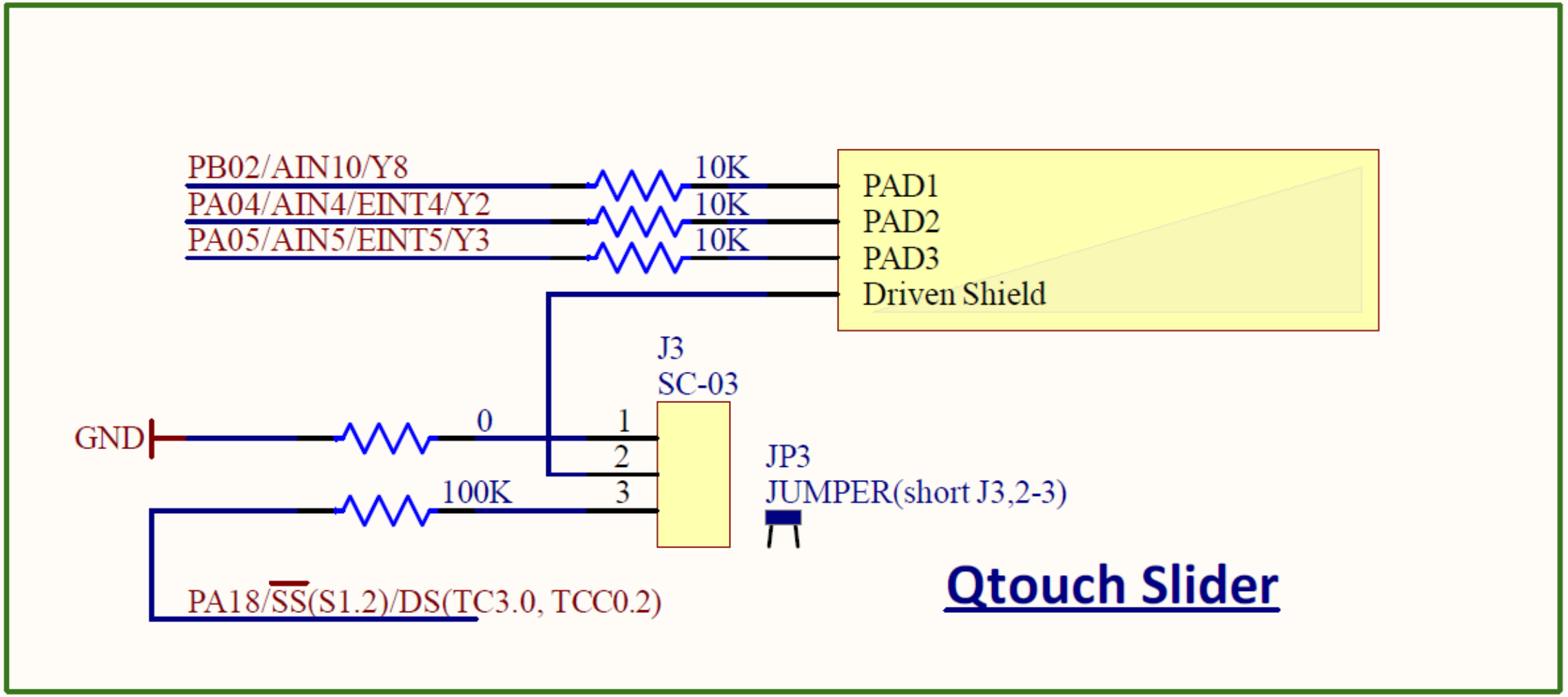
APP045 EVB QTouch® Design and Pattern

- **APP045v4 EVB to designed a three channels Self Capacitance Sensing Touch Button and Touch Slider with optional Driven Shield feature.**



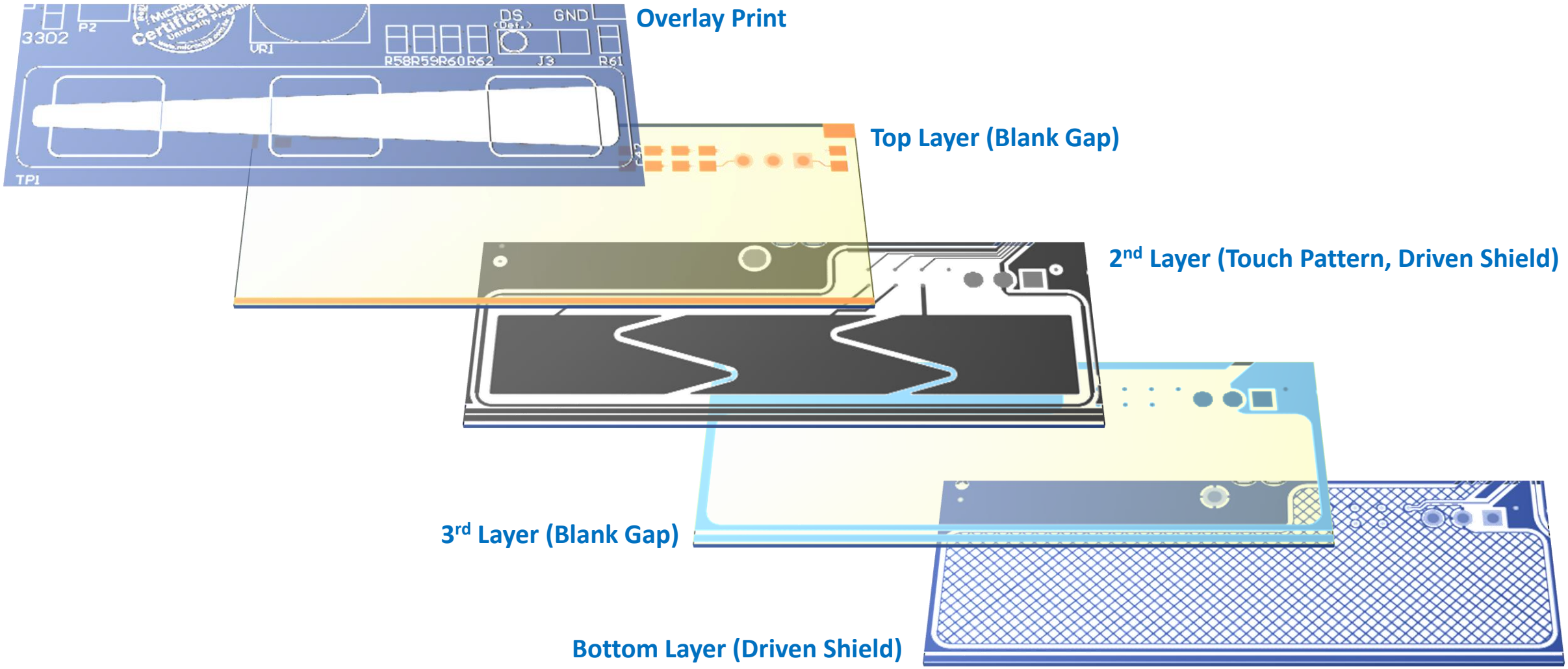
APP045 EVB QTouch® Design and Pattern

Schematic



APP045 EVB QTouch® Design and Pattern

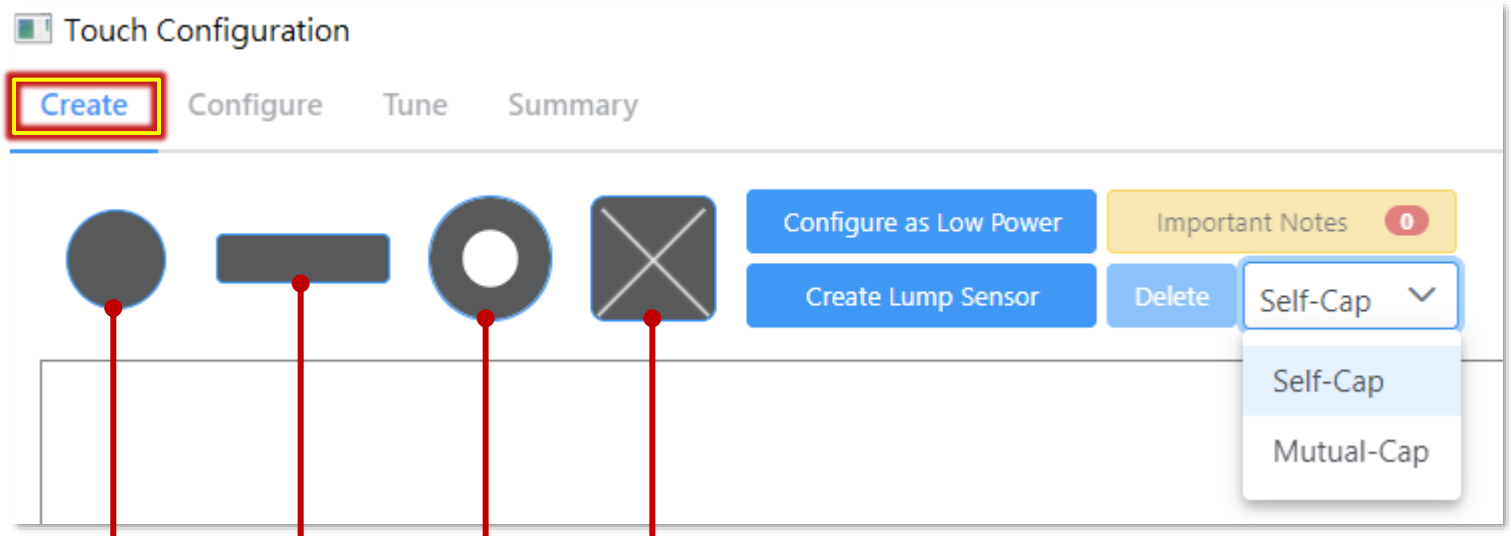
PCB Stack

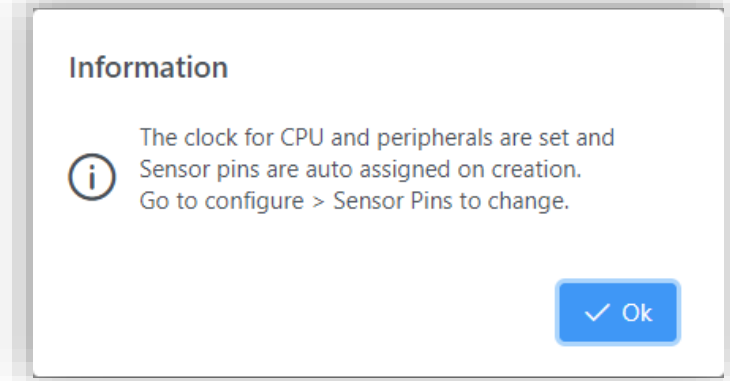


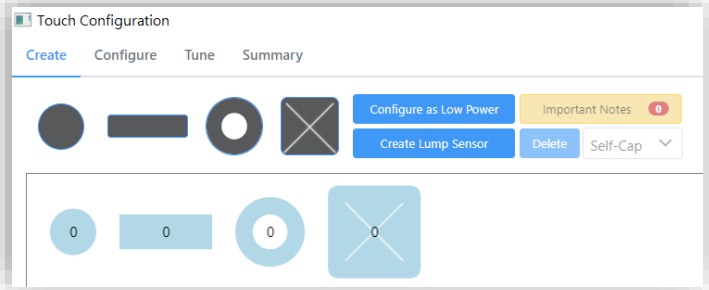
Harmony Touch Configurator

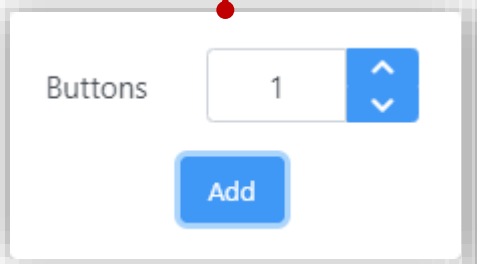
Harmony Touch Configurator

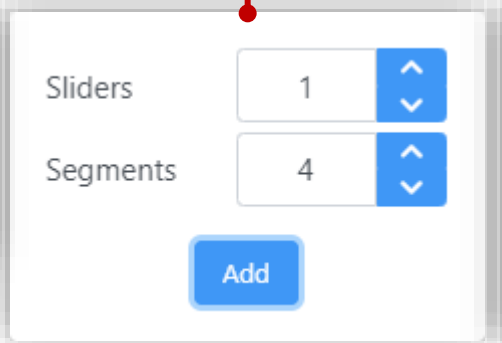
Create : Select Sensing Type and add Touch Sensors

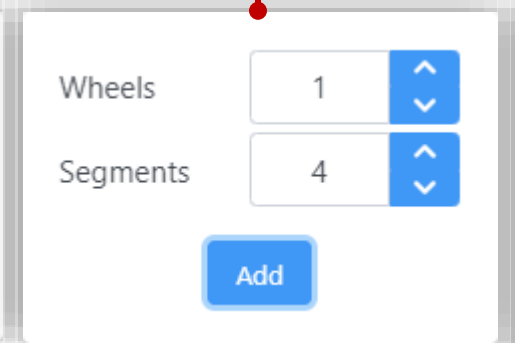


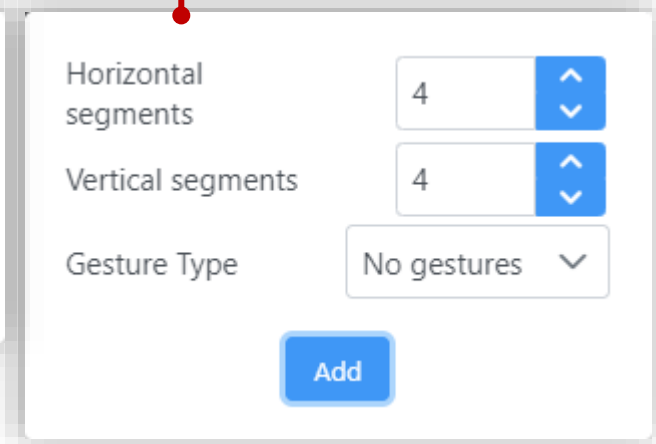






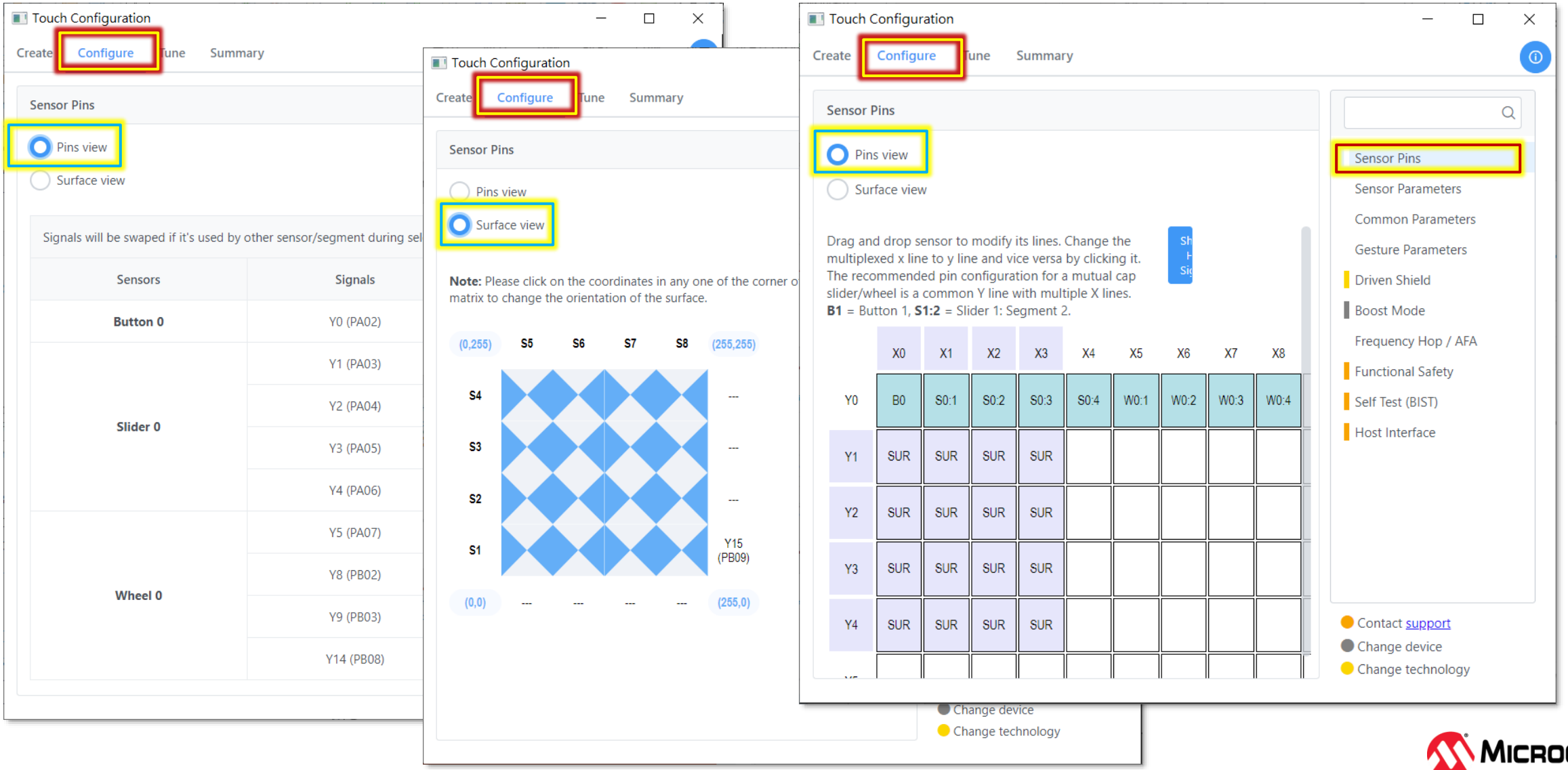






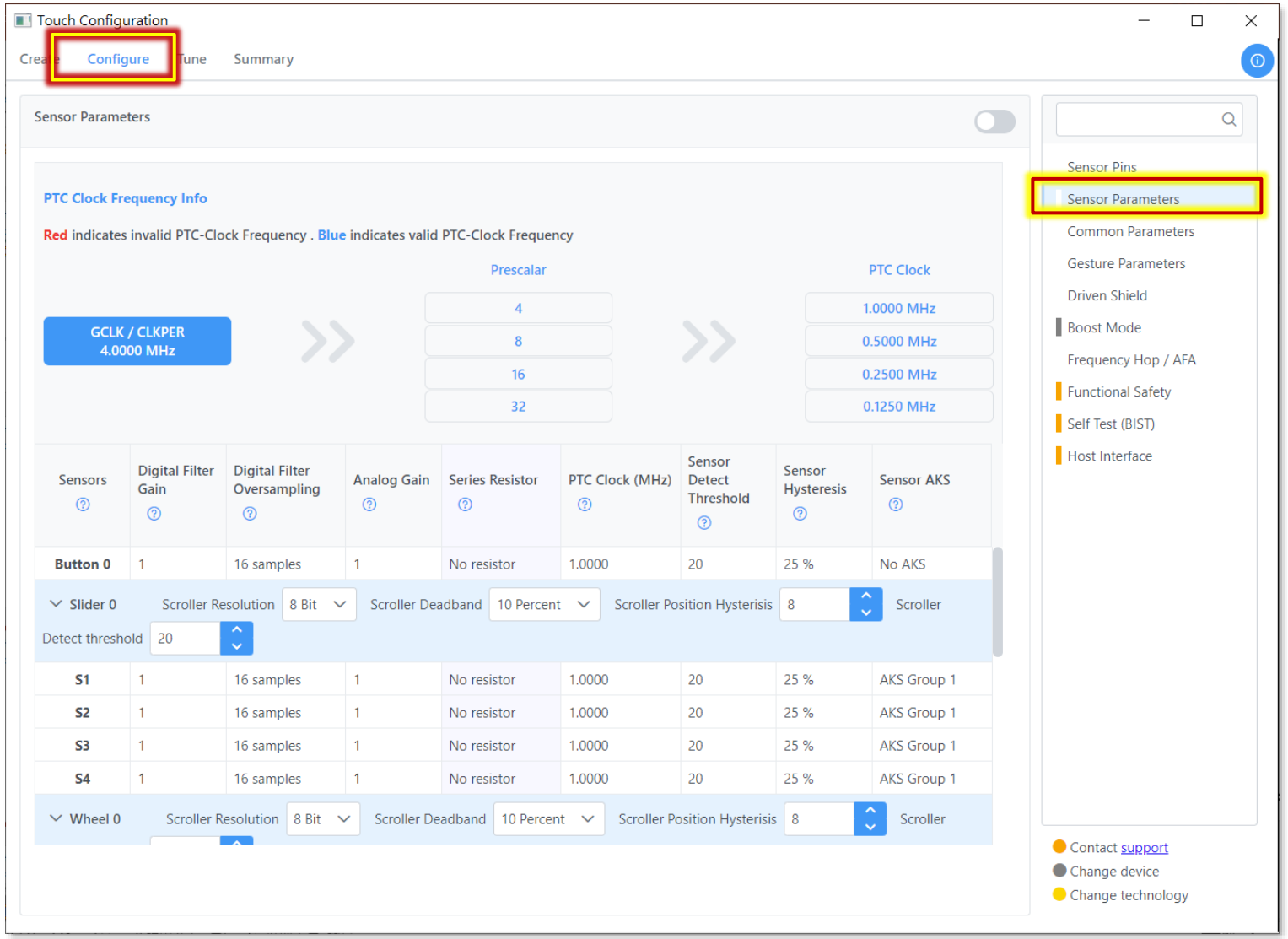
Harmony Touch Configurator

Configure – Sensor Pins : Sensors Pins signal configuration



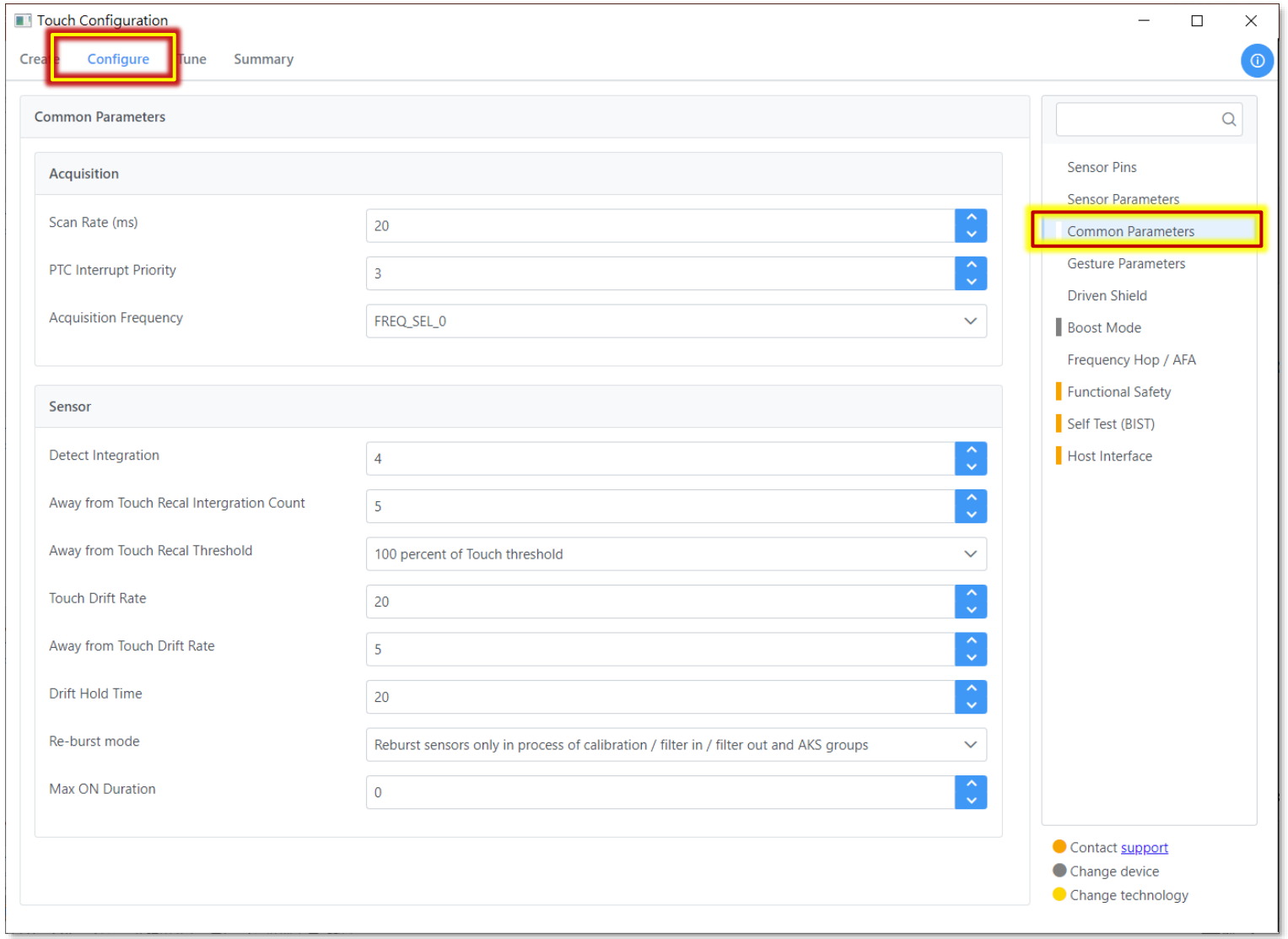
Harmony Touch Configurator

Configure – Sensor Parameters : Filter/Gain/Clock/Threshold/AKS, etc



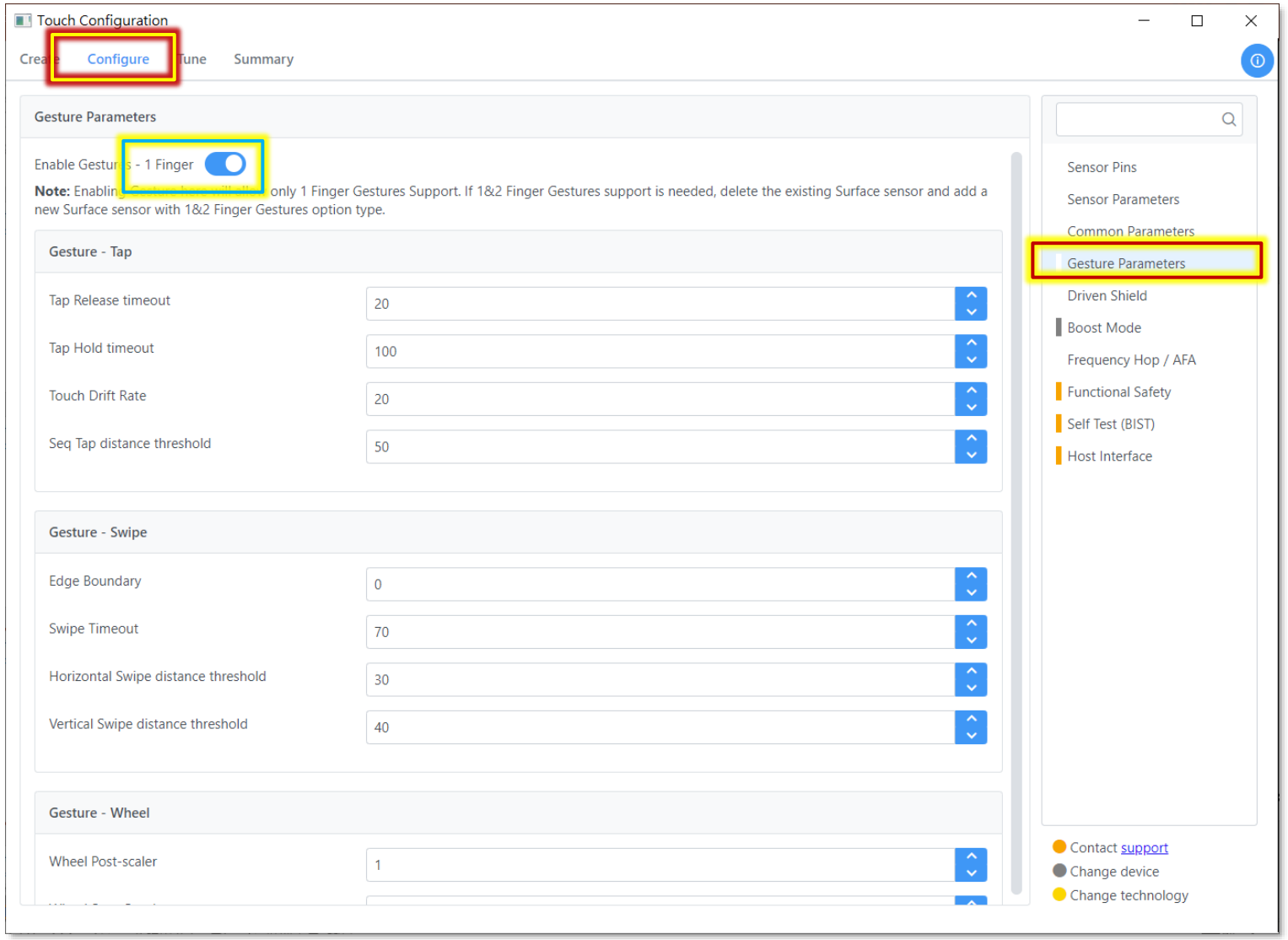
Harmony Touch Configurator

Configure – Common Parameters : Acquisition, Touch Drift, Recalibration threshold, etc



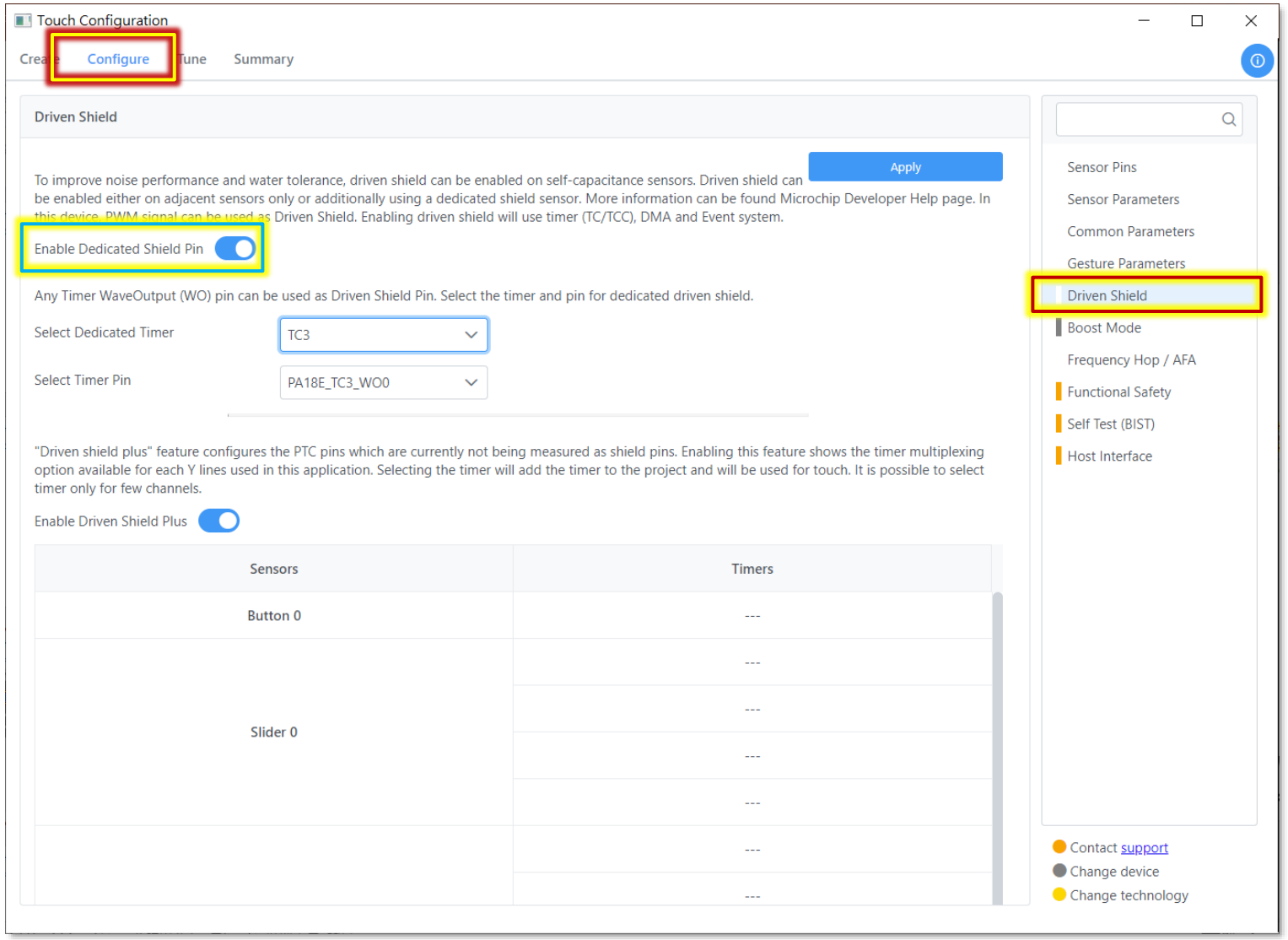
Harmony Touch Configurator

Configure – Gesture Parameters : Gesture Tap/swipe/wheel settings



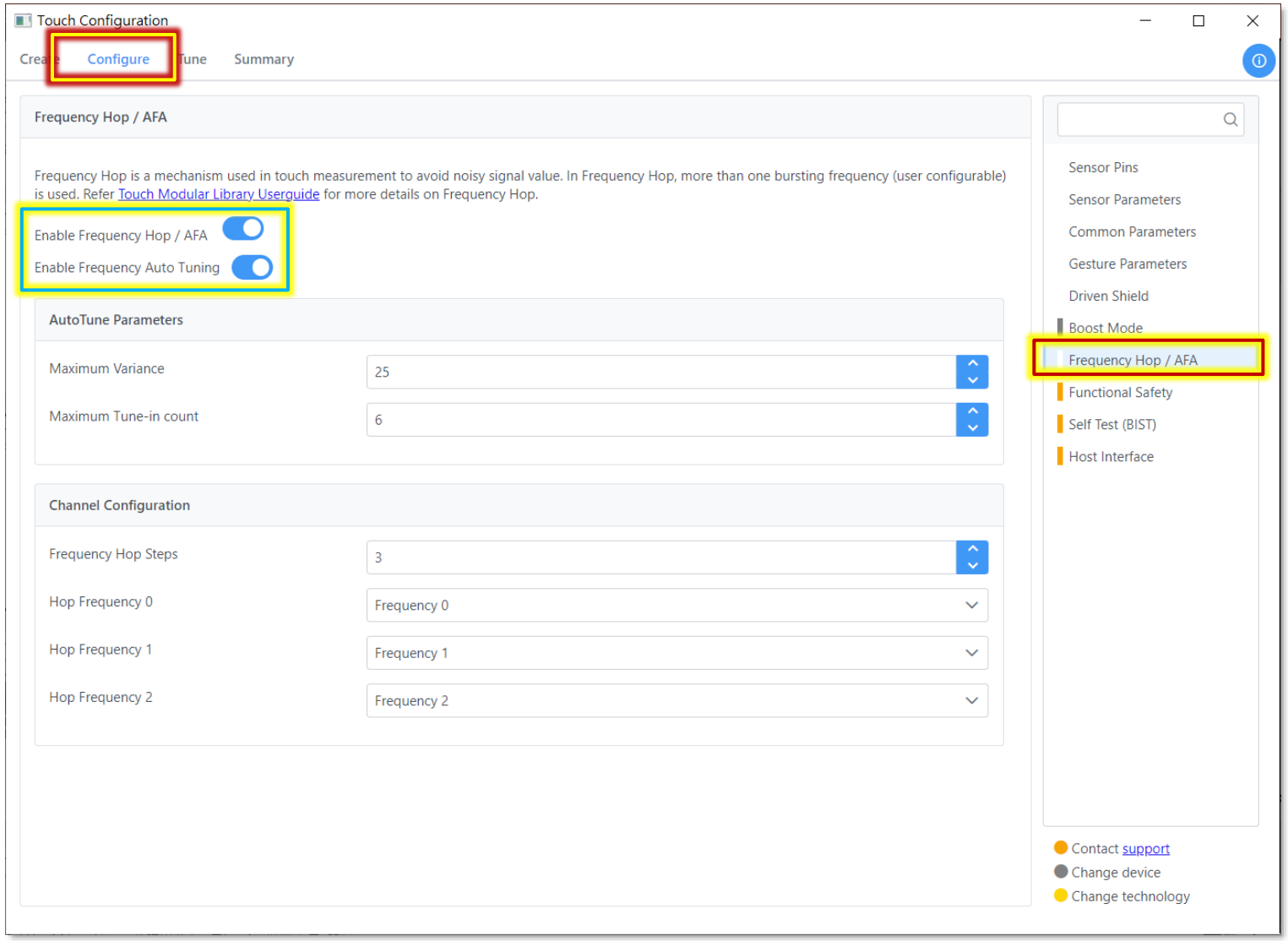
Harmony Touch Configurator

Configure – Driven Shield : Sensor design with external clock for Touch Nosie guard.



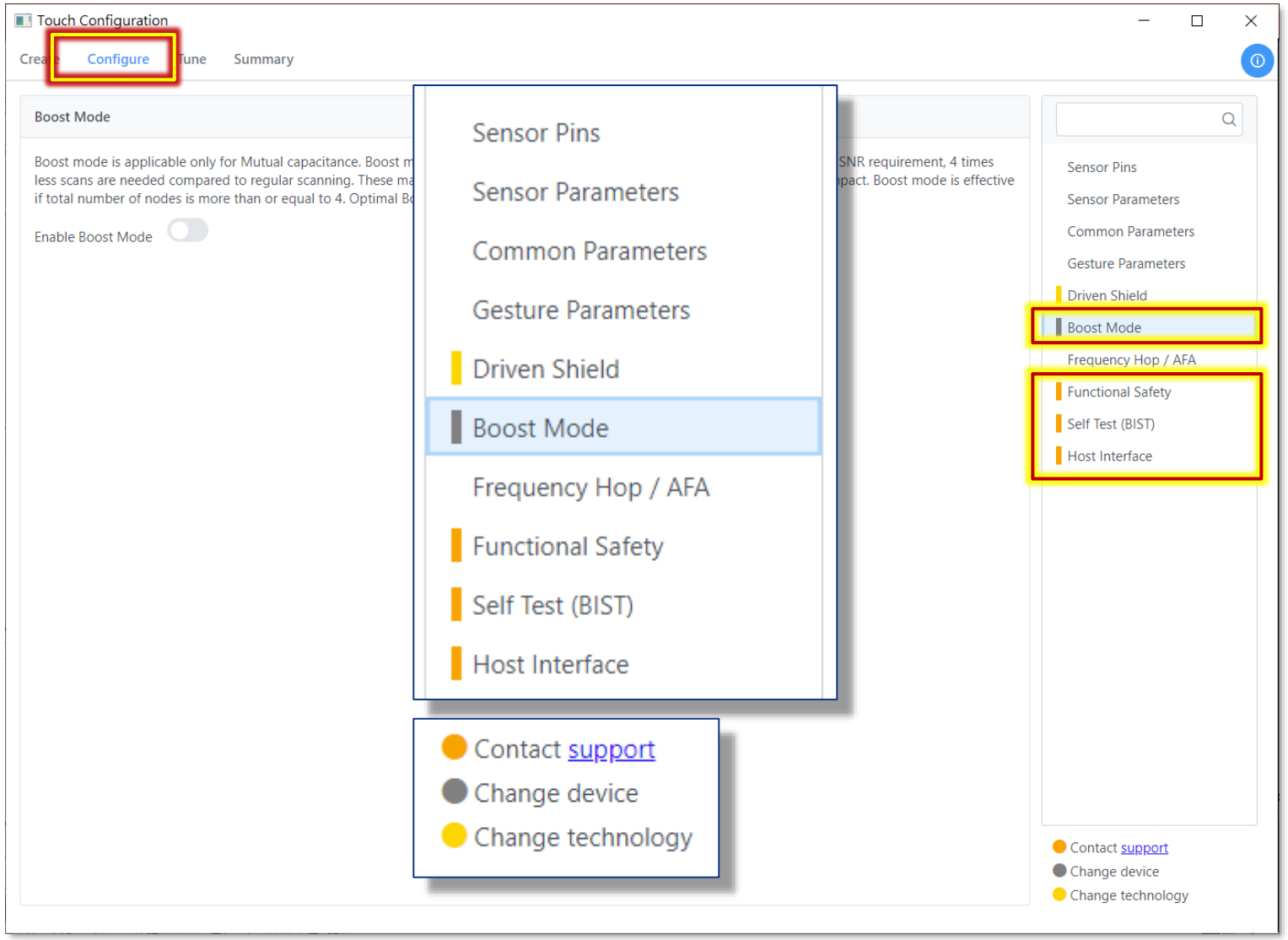
Harmony Touch Configurator

Configure – Frequency Hop / AFA



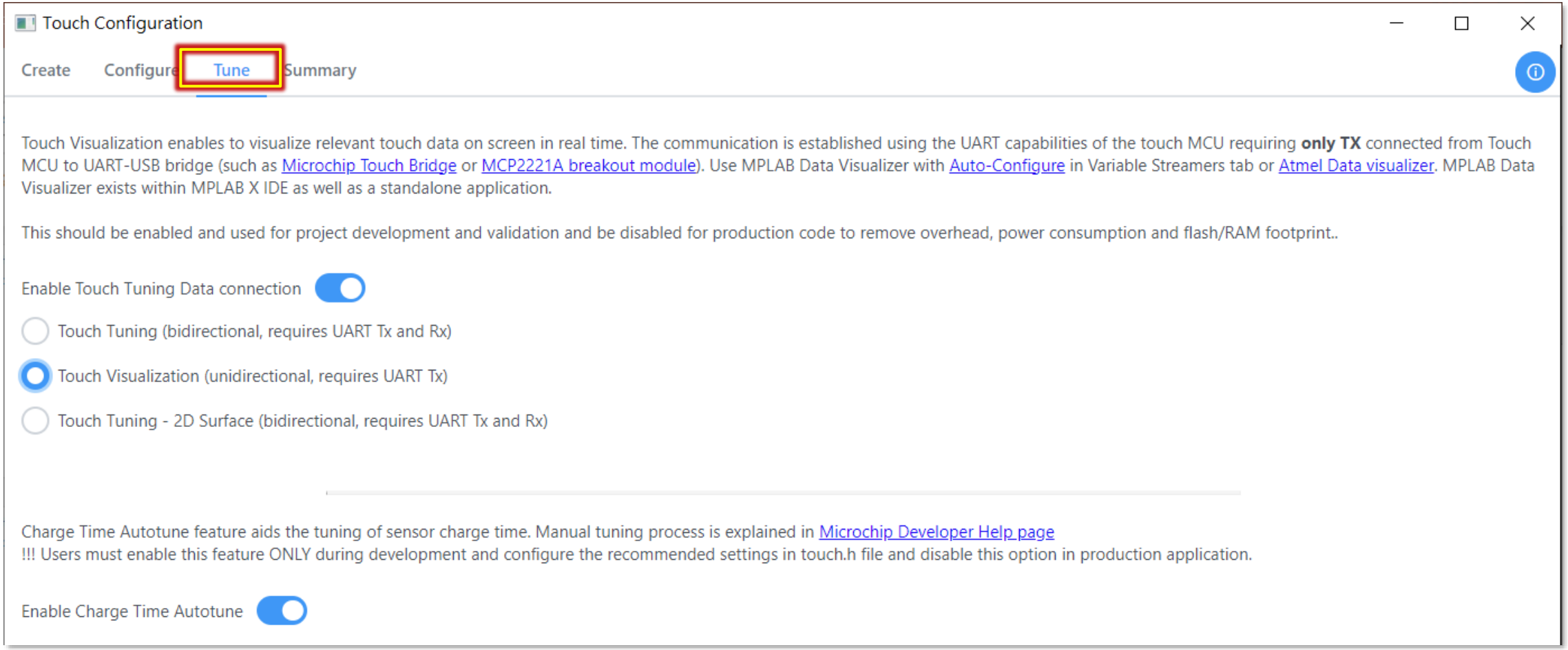
Harmony Touch Configurator

Configure – Others



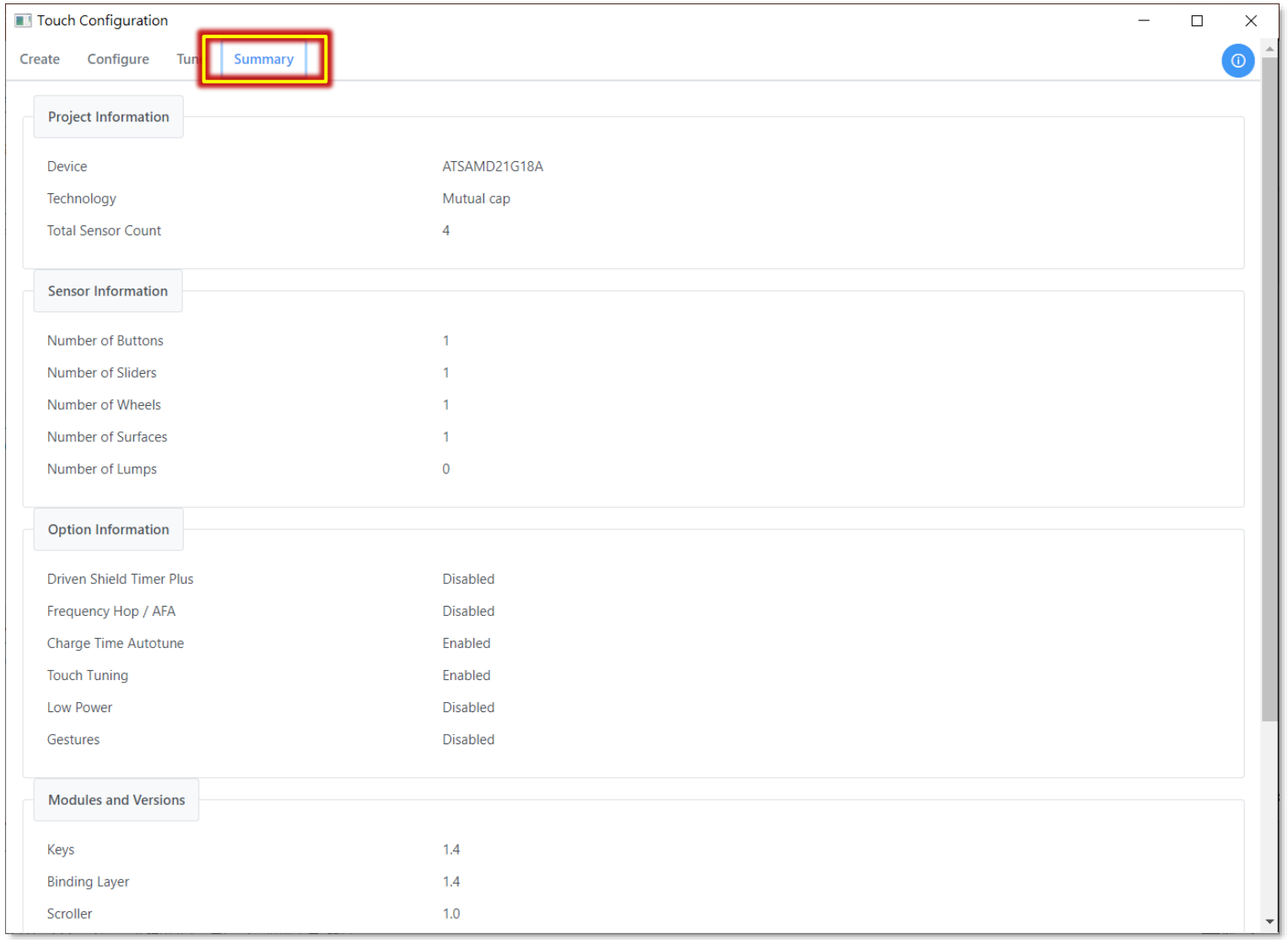
Harmony Touch Configurator

Tune – Enable the debug output for Touch Tuning



Harmony Touch Configurator

Summary – Configuration summary

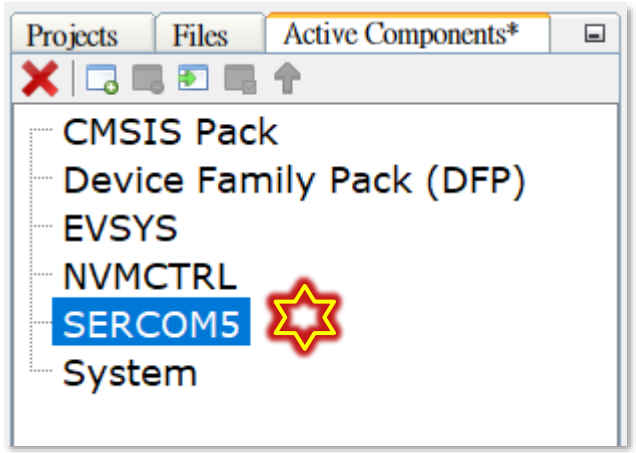


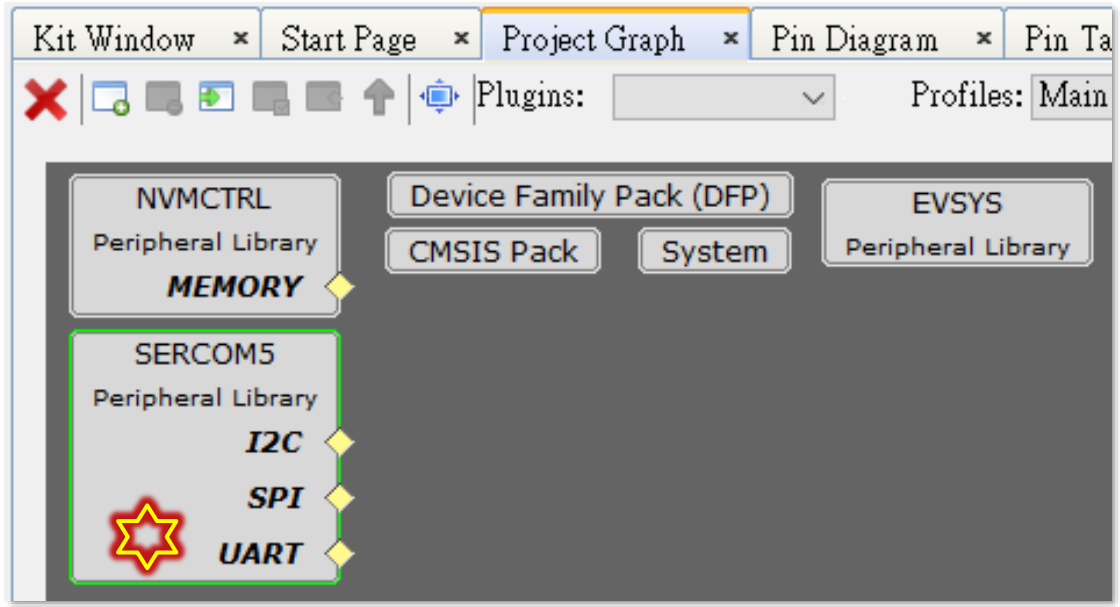
MICROCHIP

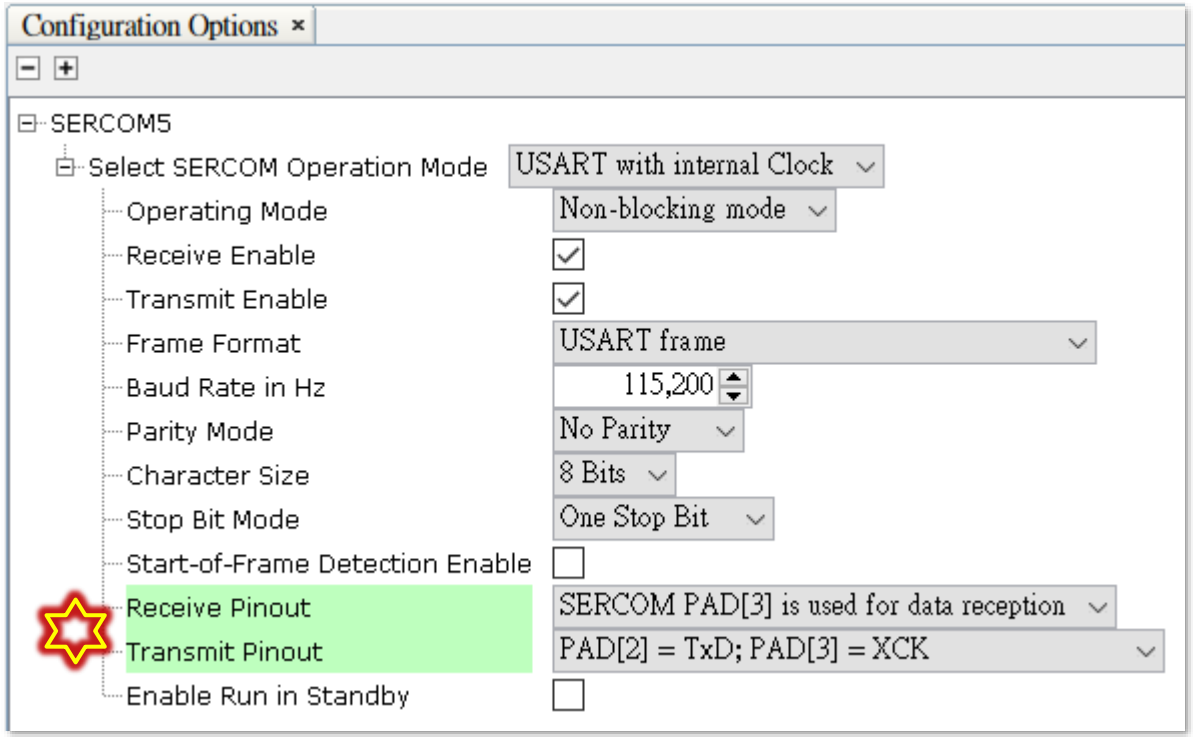
🛠 Lab1 Touch Buttons

Lab1 Touch Buttons

- **Create a new project and add SERCOM5 in MHC for UART debug message.**





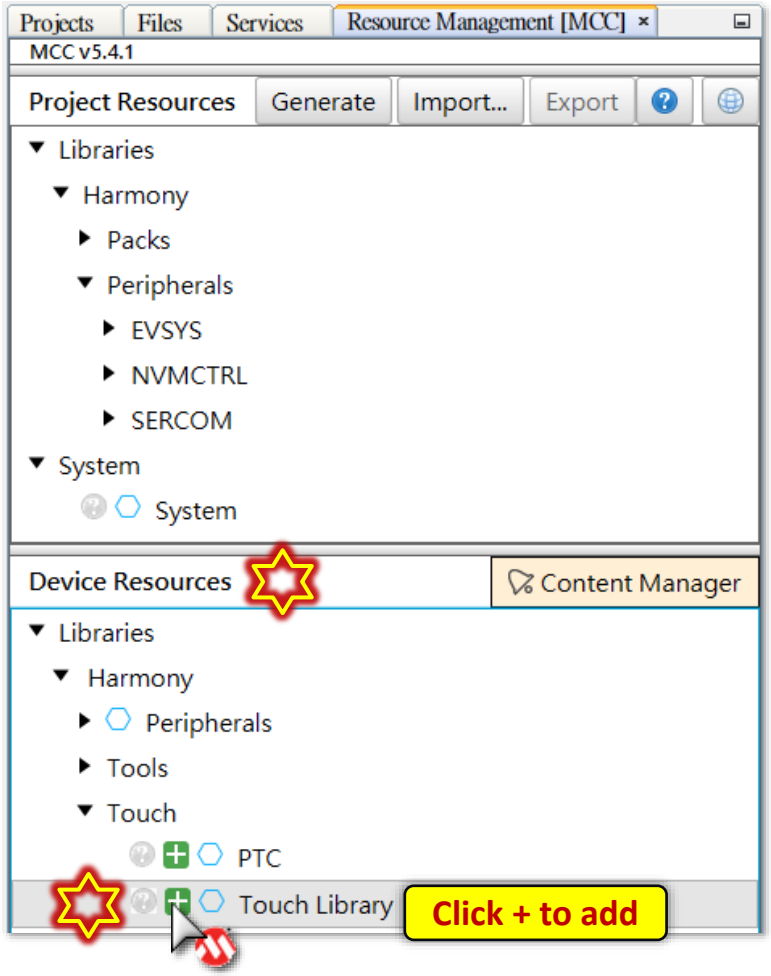


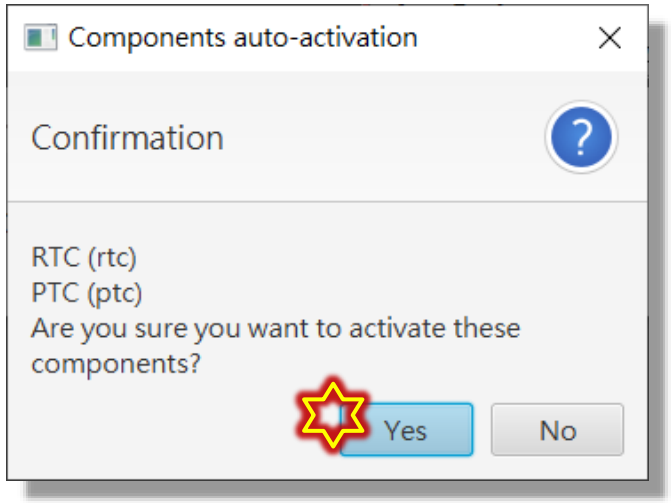
Pin Number	Pin ID	Custom Name	Function
37	PB22		SERCOM5_PAD2
38	PB23		SERCOM5_PAD3

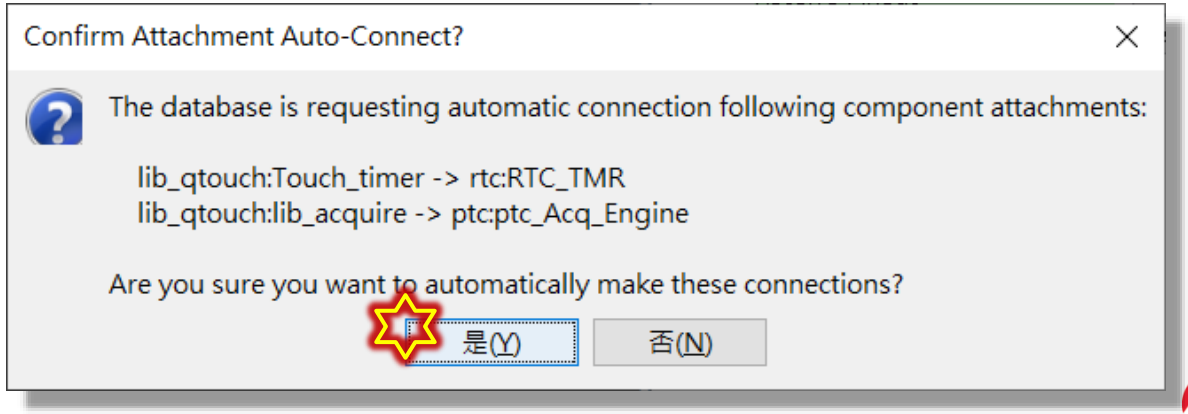
MICROCHIP

Lab1 Touch Buttons

- Find Touch Library under Touch of Device Resources and add it to your project.
- Please wait a while after click to add the Touch Library.
- RTC and PTC will request to Auto-Activation together and Auto-Connection.



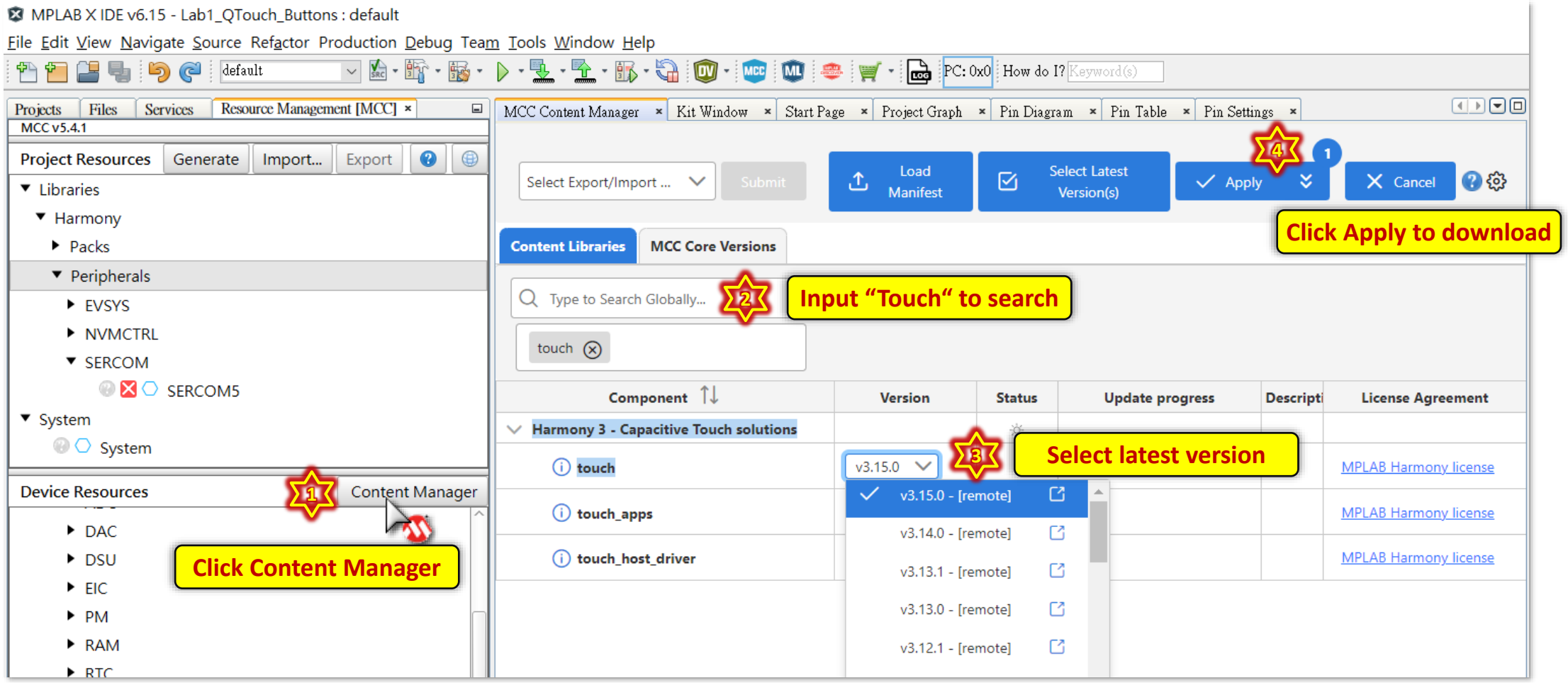




MICROCHIP

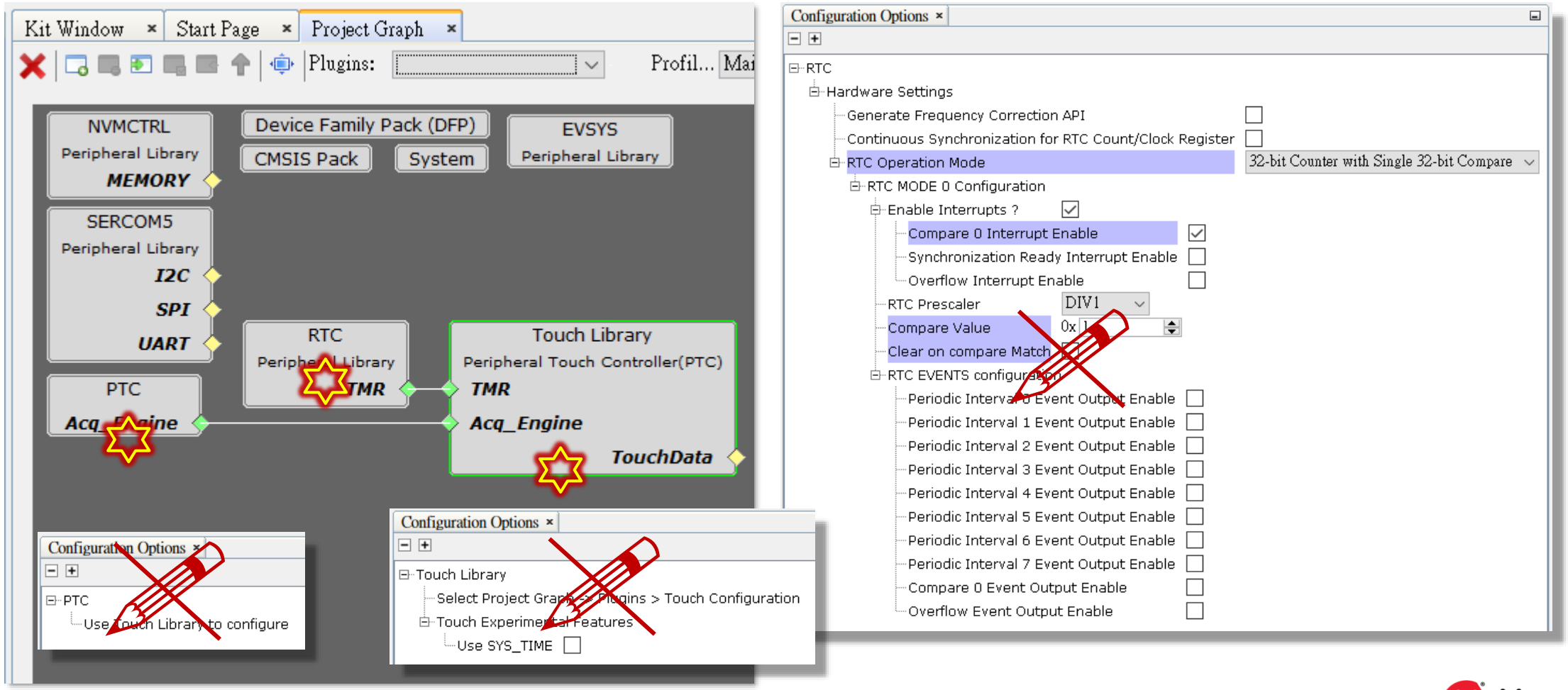
Lab1 Touch Buttons

- **Hint !! If you cannot find Touch Library in Device Resources, please go Content Manager to download it as below.**



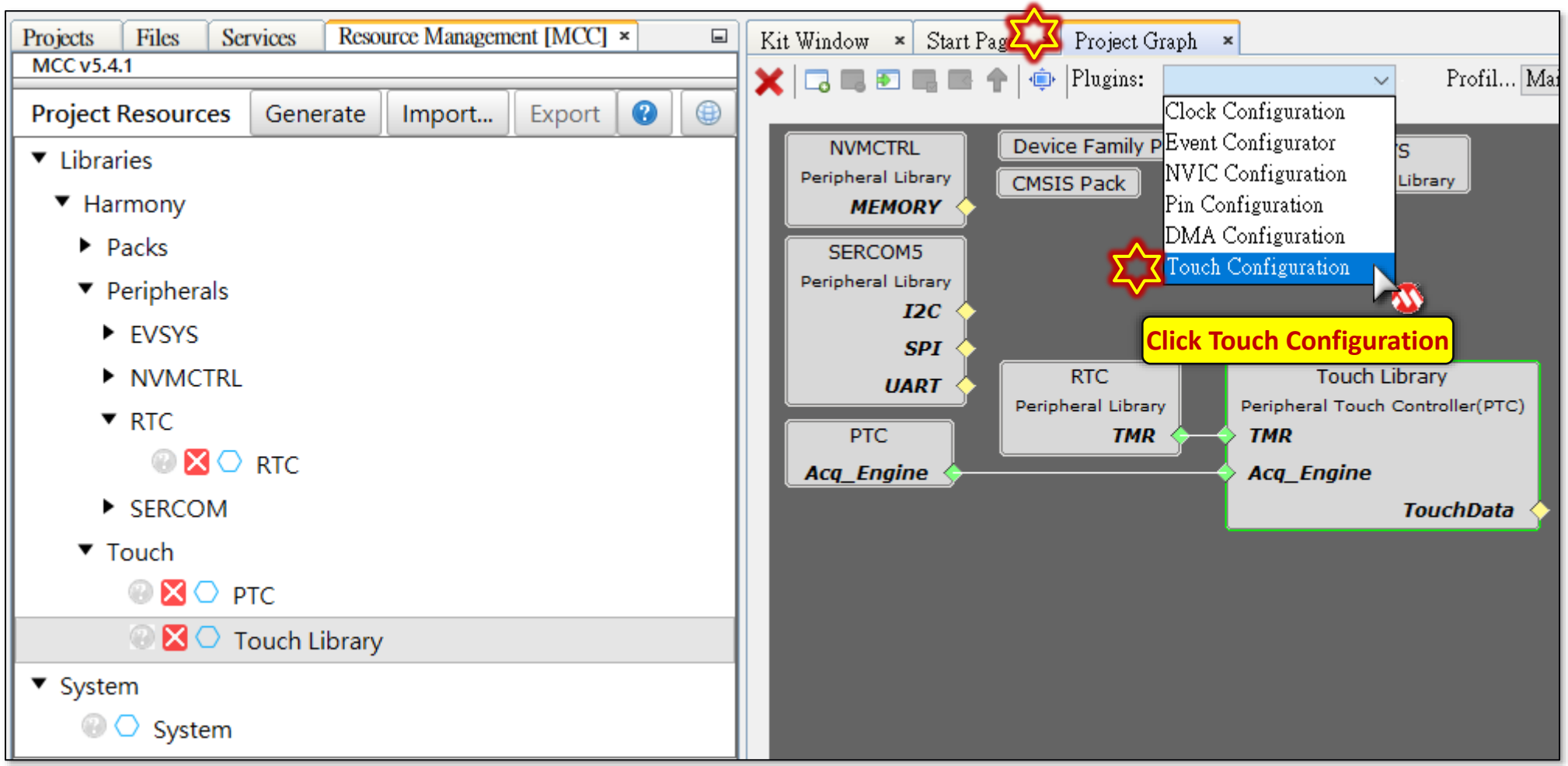
Lab1 Touch Buttons

- After add the Touch Library and Auto-Connection.
- Don’t need to configure the PTC, Touch Library or RTC, we will use Touch Configurator.



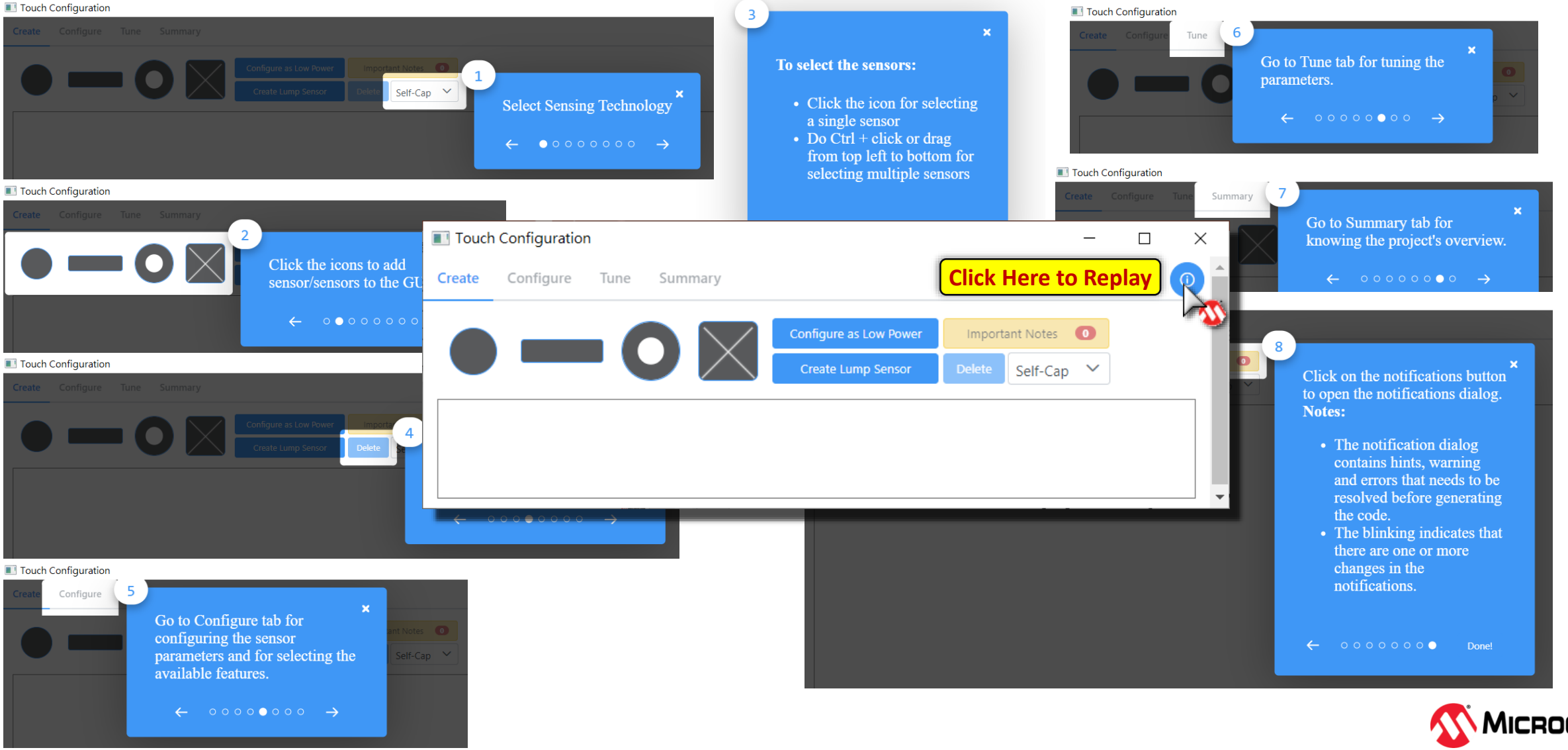
Lab1 Touch Buttons

- Open **Touch Configuration** in **Plugins** of Project Graph.



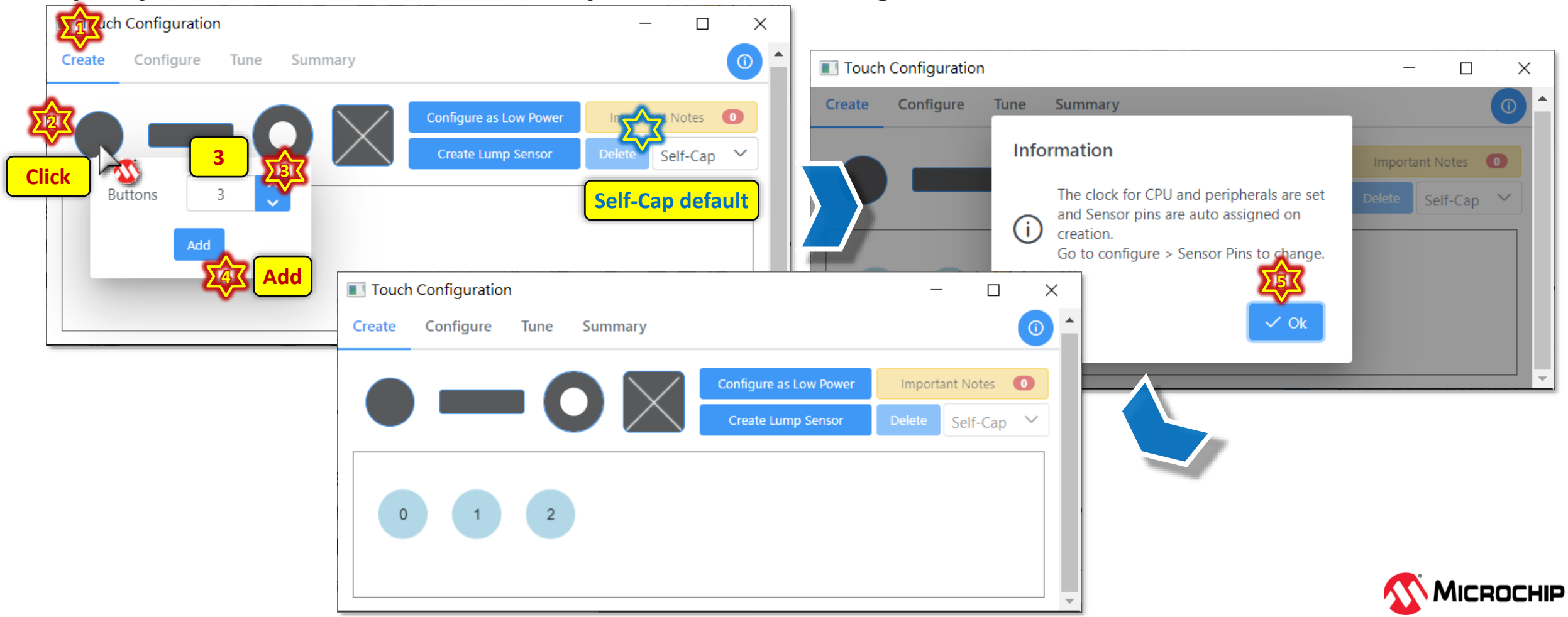
Lab1 Touch Buttons

- **An Info wizard will help you to quick understand what you could do in configurator.**



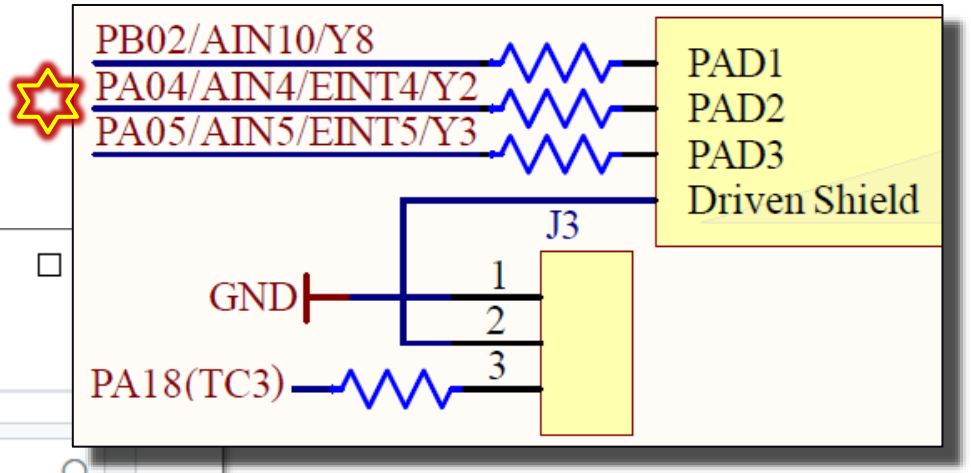
Lab1 Touch Buttons

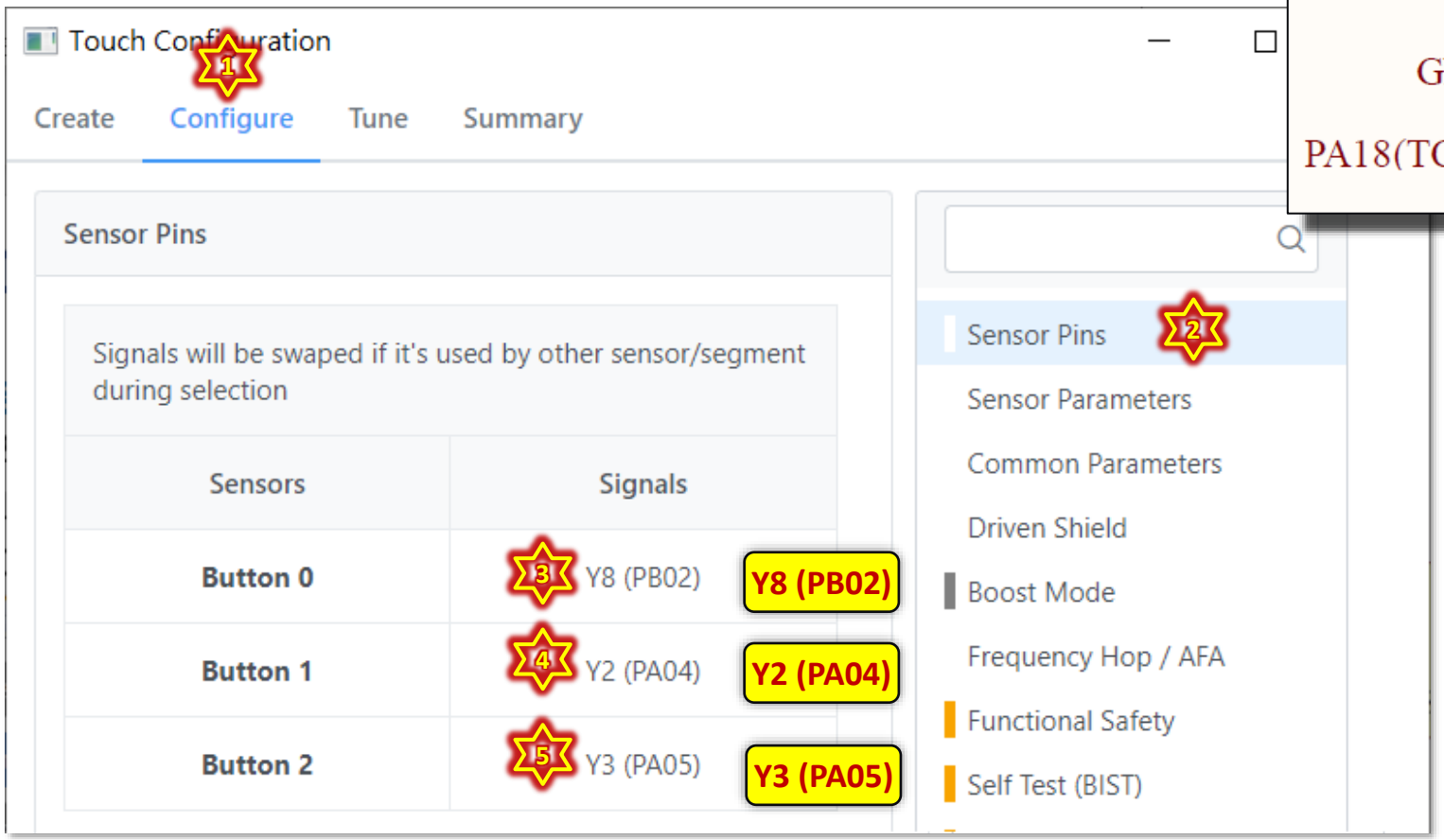
- In Create Tab, click on Button Icon and input number Buttons to 3.
- Click Add then a Warning dialog will show up to notice “The clock for CPU and peripherals are set and Sensor pins are auto assigned on creation”.



Lab1 Touch Buttons

- In Configure Tab, Click on Sensor Pins and configure Buttons Signals pins as below.
  - Button0 (Signals) : Y8 (PB02)
  - Button1 (Signals) : Y2 (PA04)
  - Button2 (Signals) : Y3 (PA05)

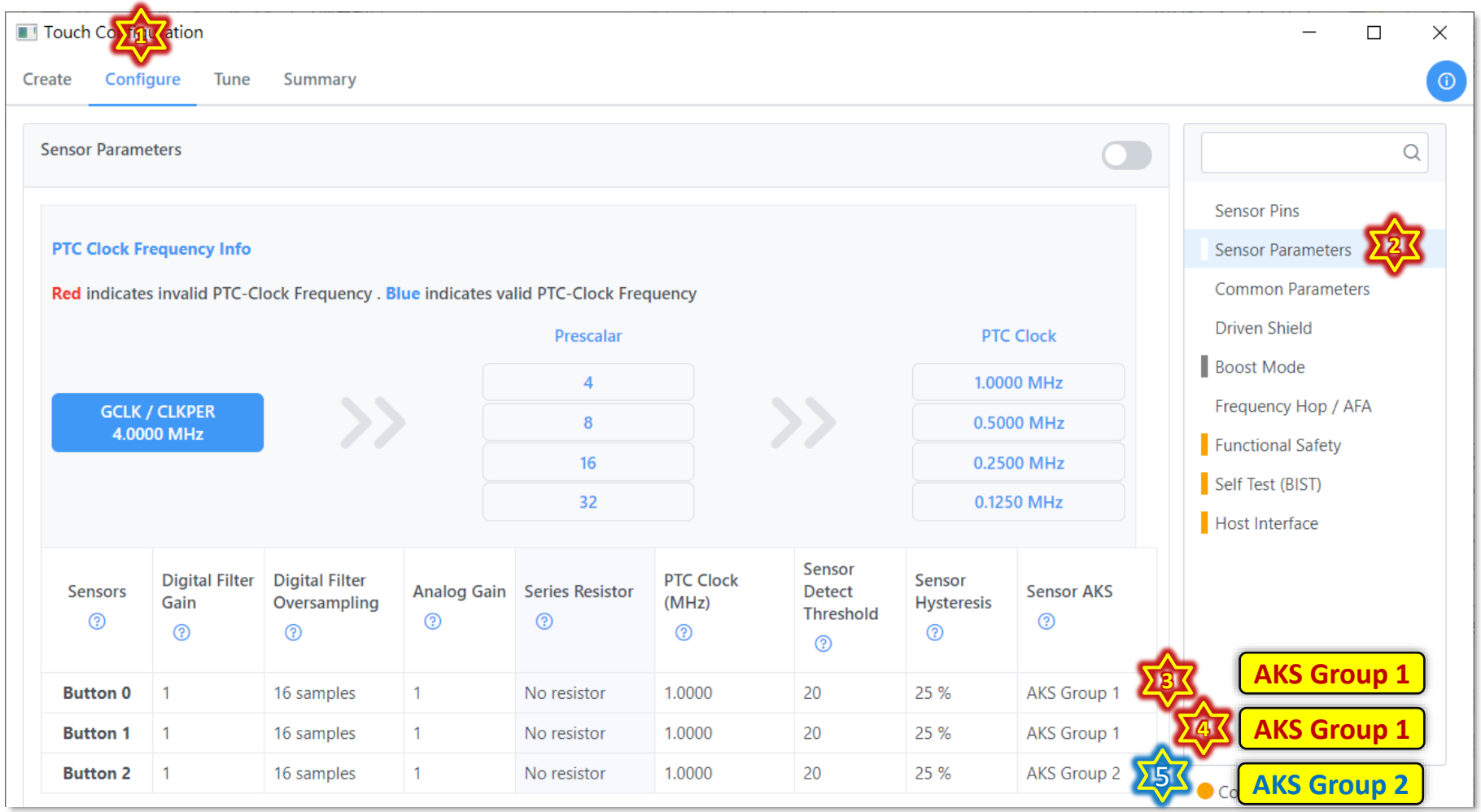




MICROCHIP

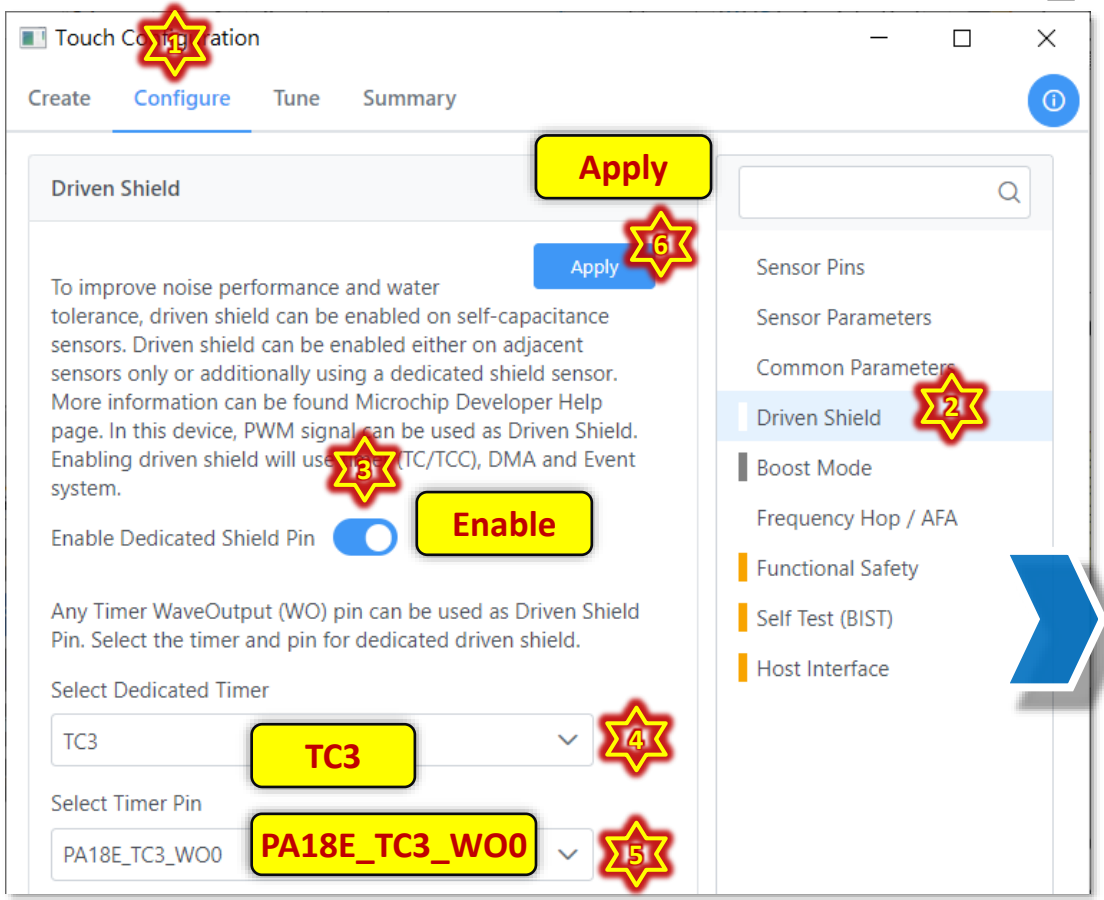
🛠 Lab1 Touch Buttons

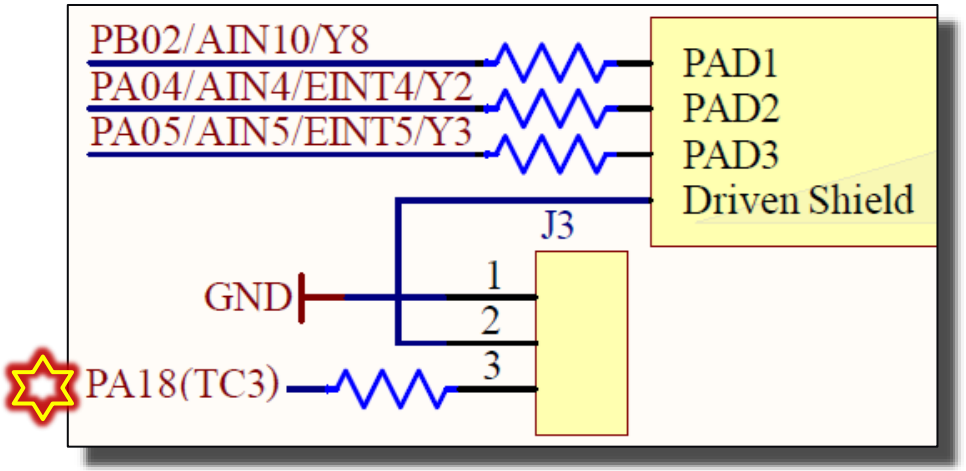
- **In Configure Tab, Click on Sensor Parameters and configure Sensor AKS(Adjacent Keys Suppression) group as below.**
  - Button0 : AKS Group 1 , Button1 : AKS Group 1 and Button2 : AKS Group 2

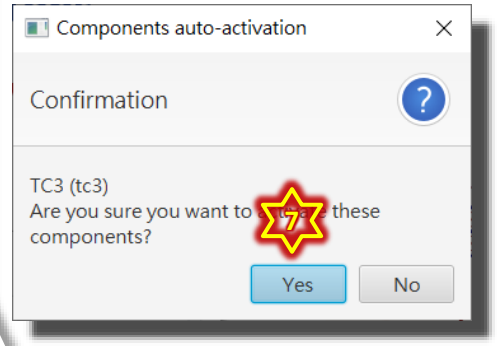


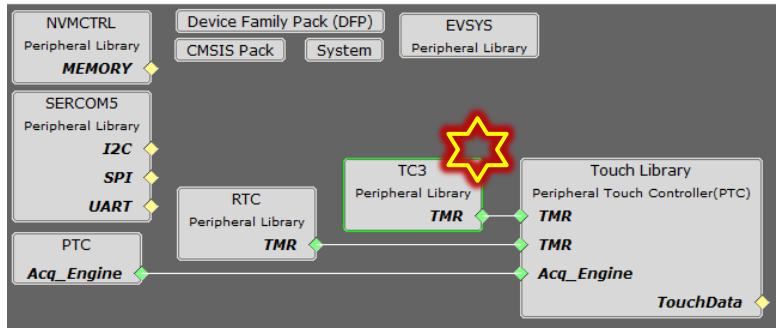
Lab1 Touch Buttons

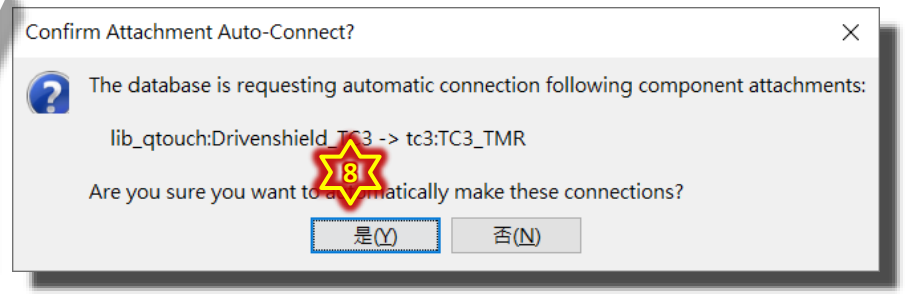
- **In Configure Tab, Click on Driven Shield and configure Dirven Shield as below.**
  - Enable Dedicated Dirven Shield : 
  - Select Dedicated Timer : TC3
  - Select Timer Pin : PA18E_TC3_WO0





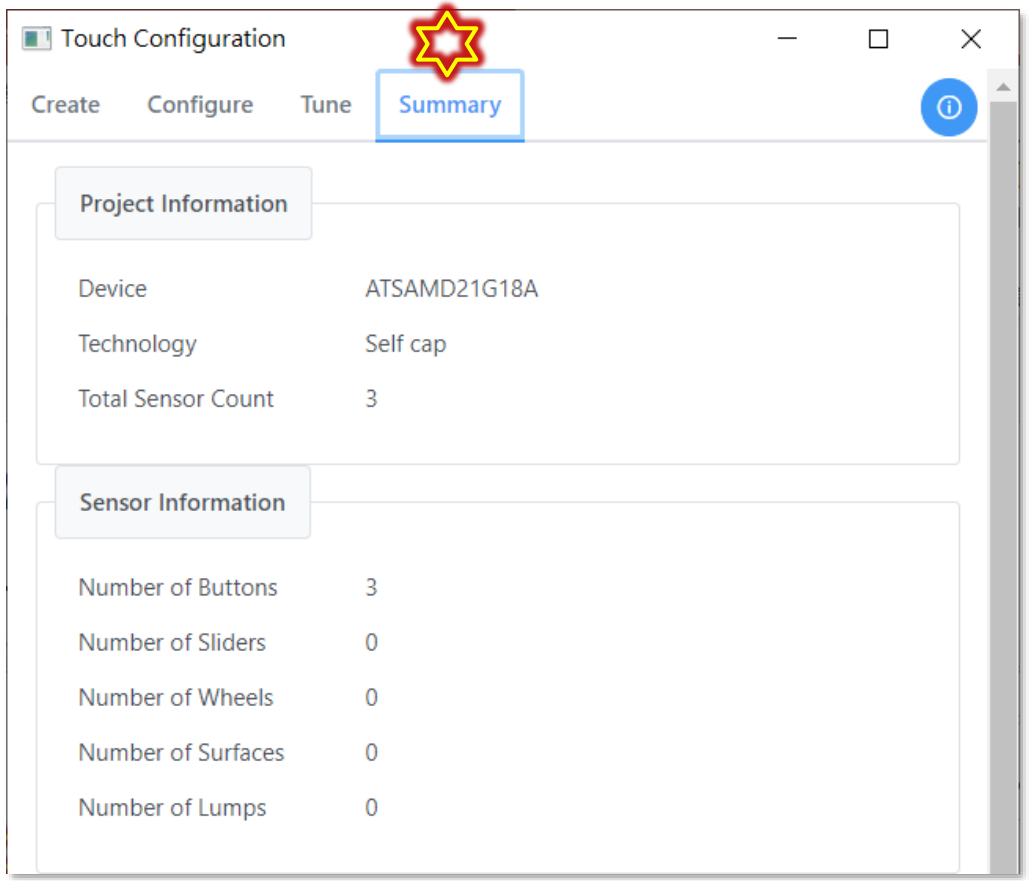


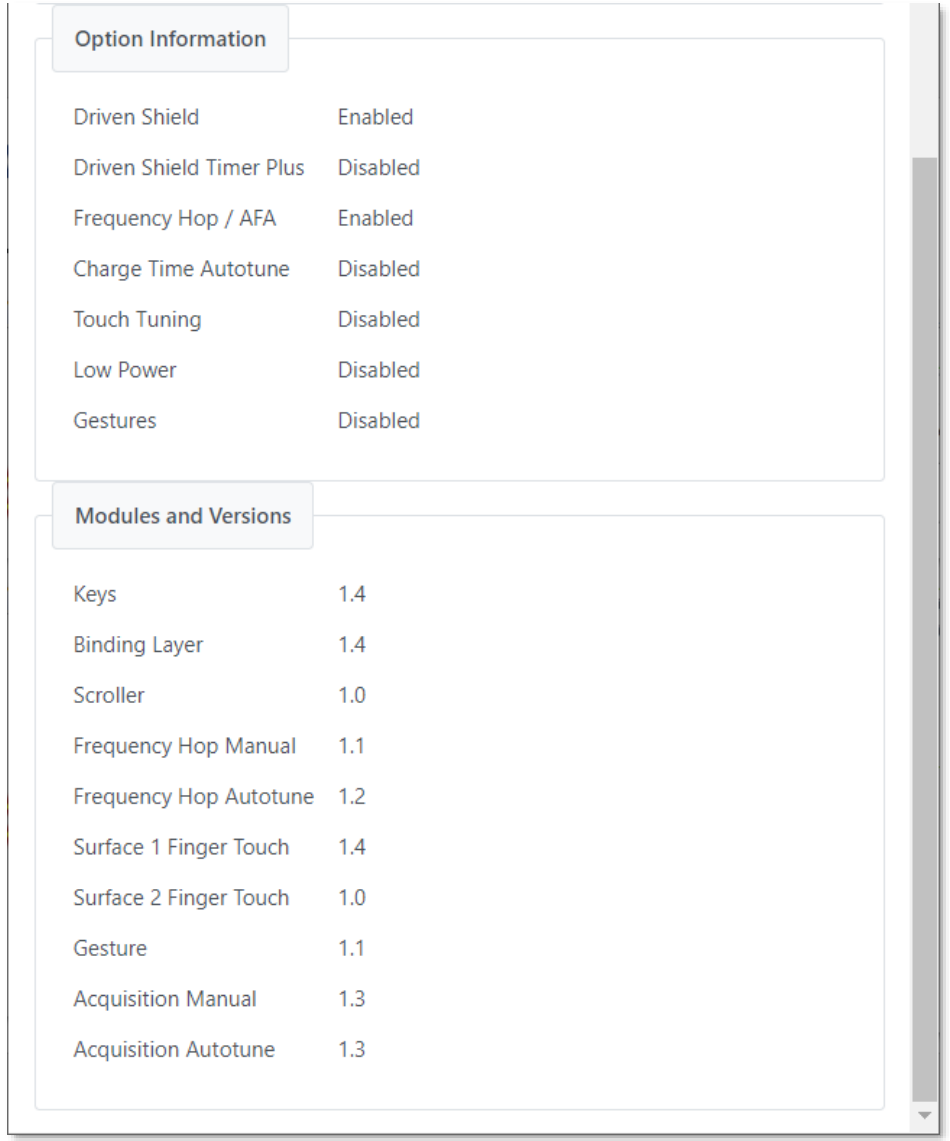




Lab1 Touch Buttons

- In Summary Tab, to figure out current configuration of Touch.

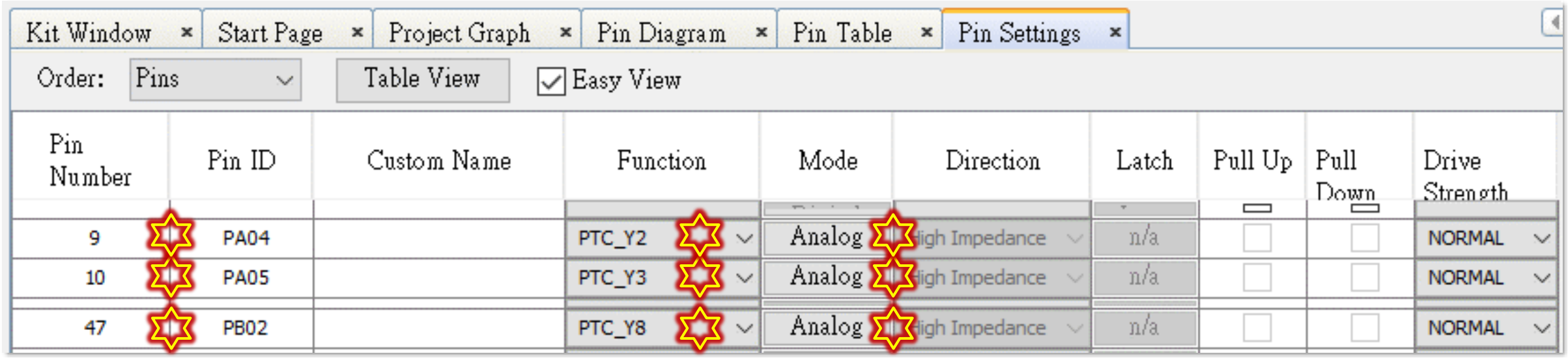




MICROCHIP

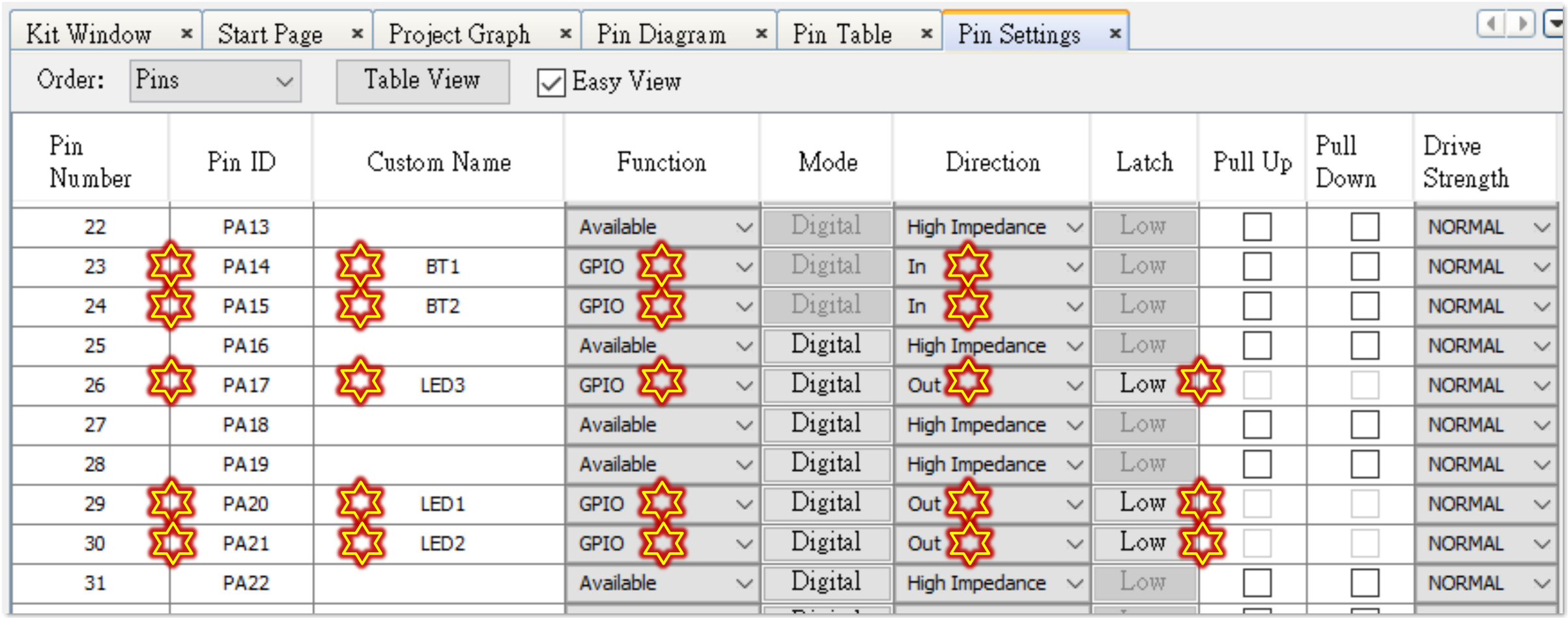
Lab1 Touch Buttons

- Check the Touch Sensor Pins in Plugin Pin Configuration of Project Graph as below.



Lab1 Touch Buttons

- **Configure LEDs and Buttons in Plugin Pin Configuration of Project Graph as below.**



Kit Window | Start Page | Project Graph | Pin Diagram | Pin Table | Pin Settings

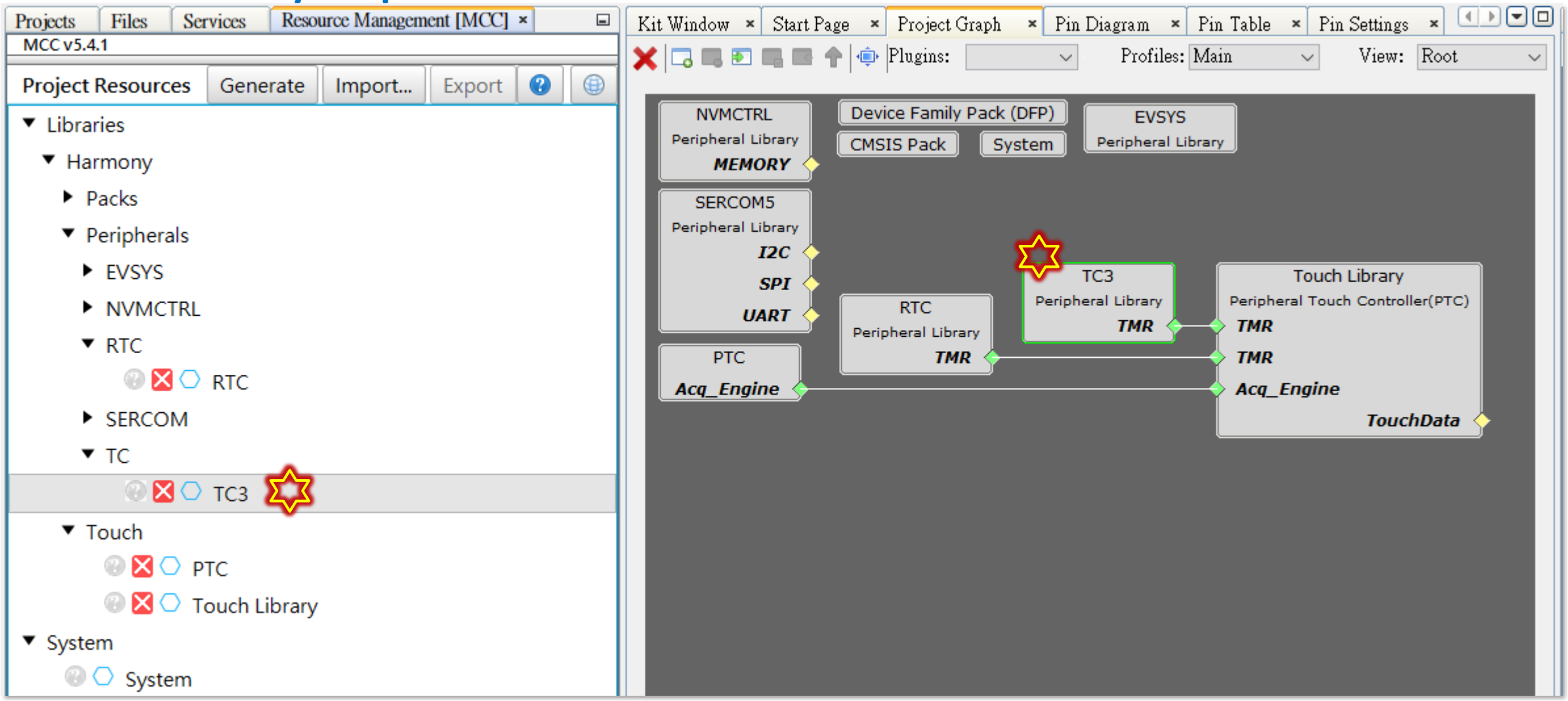
Order: Pins | Table View | Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
22	PA13		Available	Digital	High Impedance	Low	☐	☐	NORMAL
23	PA14	BT1	GPIO	Digital	In	Low	☐	☐	NORMAL
24	PA15	BT2	GPIO	Digital	In	Low	☐	☐	NORMAL
25	PA16		Available	Digital	High Impedance	Low	☐	☐	NORMAL
26	PA17	LED3	GPIO	Digital	Out	Low			NORMAL
27	PA18		Available	Digital	High Impedance	Low	☐	☐	NORMAL
28	PA19		Available	Digital	High Impedance	Low	☐	☐	NORMAL
29	PA20	LED1	GPIO	Digital	Out	Low			NORMAL
30	PA21	LED2	GPIO	Digital	Out	Low			NORMAL
31	PA22		Available	Digital	High Impedance	Low	☐	☐	NORMAL

MICROCHIP

Lab1 Touch Buttons

- Check Harmony Graph to see the difference after added Driven Shield.



🛠 Lab1 Touch Buttons

API functions in Touch.c

uint8_t measurement_done_touch	Touch acquisition done flag
touch_process ()	Main processing function of touch library. This function initiates the acquisition, calls post processing after the acquisition complete and sets the flag for next measurement based on the sensor status.
touch_init ()	Initialization PTC, timer, and datastreamer modules of touch library.
touch_timer_config ()	Initialization necessary RTC timer for Touch in touch_init()
touch_sensors_config ()	Initialization of touch key sensors in touch_init()
get_sensor_node_signal ()	Get sensor node aquiction signals.
update_sensor_node_signal ()	Update sensor node aquiction signals.
get_sensor_node_reference ()	Get sensor node channel reference.
update_sensor_node_reference ()	Update sensor node channel reference.
get_sensor_cc_val ()	Get sensor node compansation capacitor value.
update_sensor_cc_val ()	Update sensor node compansation capacitor value.
get_sensor_state ()	Get sensor node state
update_sensor_state ()	Update sensor node state
calibrate_node ()	Sensor node calibration

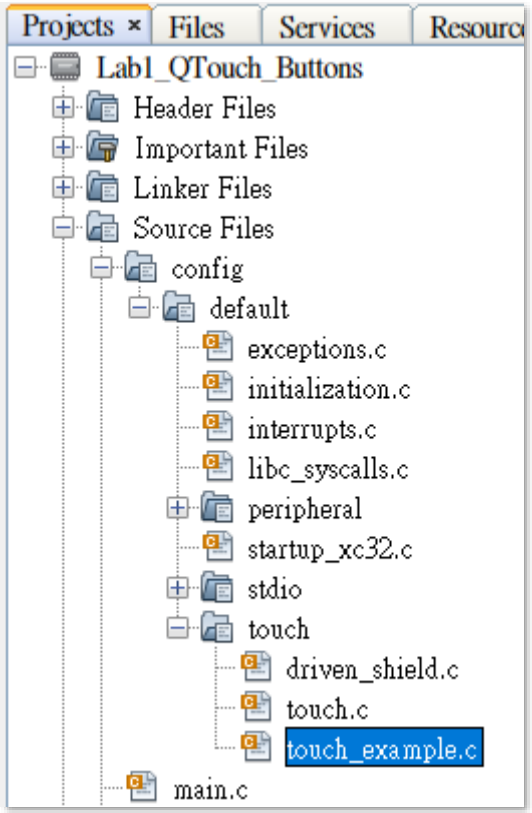
API functions in driven_shield.c

drivenshield_configure ( )	Iinitialization Driven Shield for Touch in touch_init()

MICROCHIP

Lab1 Touch Buttons

- **Example functions in touch_example.c depend to sensors you to configured.**




```c
#include "touch_example.h"

void touch_mainloop_example(void){

    /* call touch process function */
    touch_process();

    if(measurement_done_touch == 1u)
    {
        measurement_done_touch = 0u;
        // process touch data
    }
}
```

```c
/*============================================================================
void touch_status_display(void)
----------------------------------------------------------------------------
Purpose: Sample code snippet to demonstrate how to check the status of the
         sensors
Input  : none
Output : none
Notes  : none
============================================================================*/
void touch_status_display(void)
{
    uint8_t key_status = 0u;
    key_status = get_sensor_state(0) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }

    key_status = get_sensor_state(1) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }

    key_status = get_sensor_state(2) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }
}
```

🛠 Lab1 Touch Buttons

- **Implement below function to output message on UART console.**

```
#include <string.h>
#include <stdio.h>
#include <stdarg.h>

extern volatile uint8_t measurement_done_touch;

#define SYS_CONSOLE_PRINT_BUFFER_SIZE   200
static char consolePrintBuffer[SYS_CONSOLE_PRINT_BUFFER_SIZE];
void myprintf(const char *format, ...)
{
    size_t len = 0;
    va_list args = {0};

    va_start(args, format);
    len = vsnprintf(consolePrintBuffer, SYS_CONSOLE_PRINT_BUFFER_SIZE, format, args);
    va_end(args);

    if ((len > 0) && (len < SYS_CONSOLE_PRINT_BUFFER_SIZE))
    {
        consolePrintBuffer[len] = '\0';
        SERCOM5_USART_Write((uint8_t*)consolePrintBuffer, len);
        while (SERCOM5_USART_WriteIsBusy());
    }
}

int main ( void )
{
    SYS_Initialize ( NULL );

    myprintf("\033[2J");
    myprintf("\033[1;1H");
    myprintf("\033[?25l");
    myprintf("-----------------\r\n");
    myprintf("  Touch Button\r\n");
    myprintf("-----------------\r\n");
    myprintf("AKS  1    1    2\r\n");
    ...
```

MICROCHIP

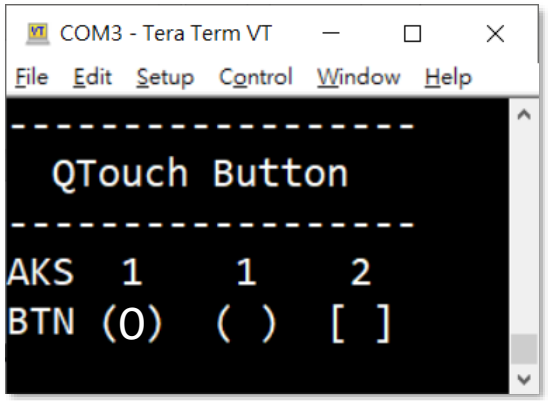
Lab1 Touch Buttons

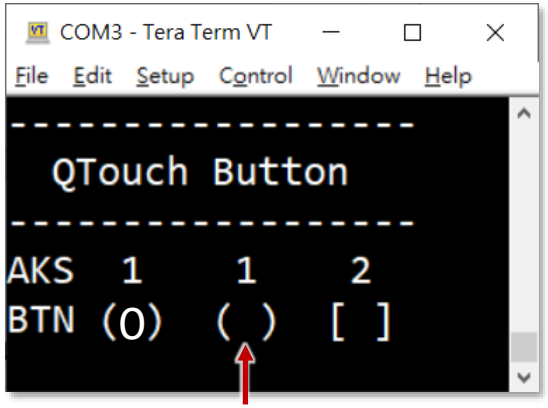
- Implement below function in while loop of main.c to make Touch Buttons works.

```
int main ( void )
{
    bool KeyStatus[3] = {false, false, false};
    ...

    while ( true )
    {
        touch_process();

        if( measurement_done_touch == 1 )
        {
            measurement_done_touch = 0;
            switch( get_sensor_state(0) )
            {
            case QTM_KEY_STATE_DETECT:  KeyStatus[0]=true;  LED1_Set();   break;
            case QTM_KEY_STATE_NO_DET:  KeyStatus[0]=false; LED1_Clear(); break;
            }

            switch( get_sensor_state(1) )
            {
            case QTM_KEY_STATE_DETECT:  KeyStatus[1]=true;  LED2_Set();   break;
            case QTM_KEY_STATE_NO_DET:  KeyStatus[1]=false; LED2_Clear(); break;
            }

            switch( get_sensor_state(2) )
            {
            case QTM_KEY_STATE_DETECT:  KeyStatus[2]=true;  LED3_Set();   break;
            case QTM_KEY_STATE_NO_DET:  KeyStatus[2]=false; LED3_Clear(); break;
            }

            myprintf("\033[5;1H");
            myprintf("BTN (%c)  (%c)  [%c]", (KeyStatus[0]?'O':' '),
                                             (KeyStatus[1]?'O':' '),
                                             (KeyStatus[2]?'X':' ') );
        }
        ...
    }
```

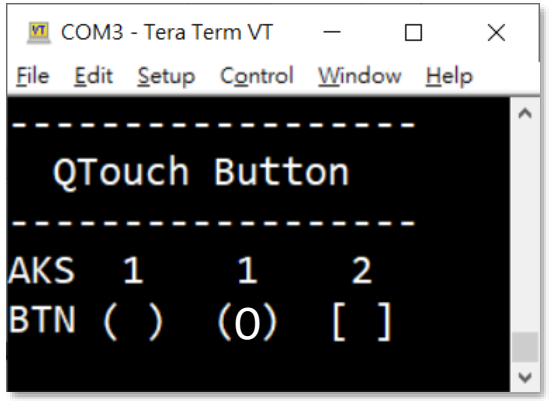
Lab1 Touch Buttons

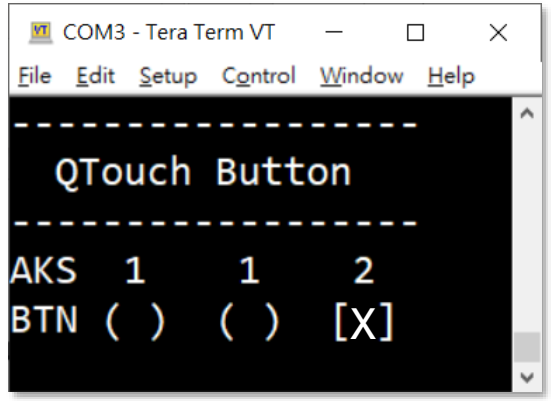
- **Result of Touch Key implementation.**

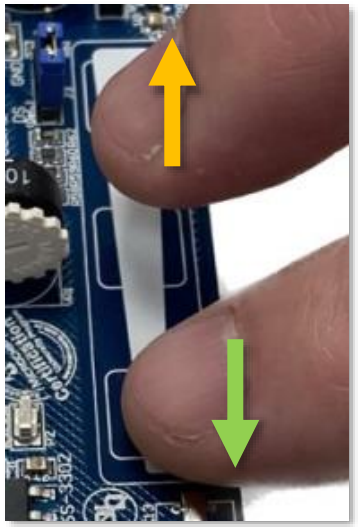


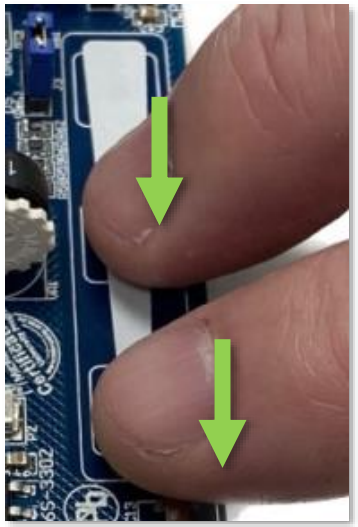


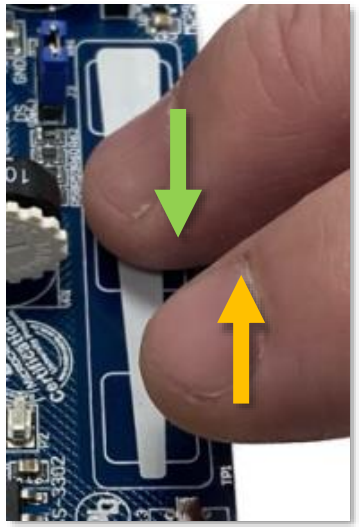
(Adjacent Keys Suppression)

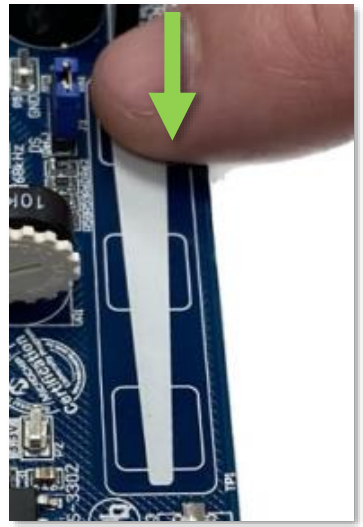












🛠 Lab1 Touch Buttons

Discuss

- **Implement manual Touch Calibration feature.**

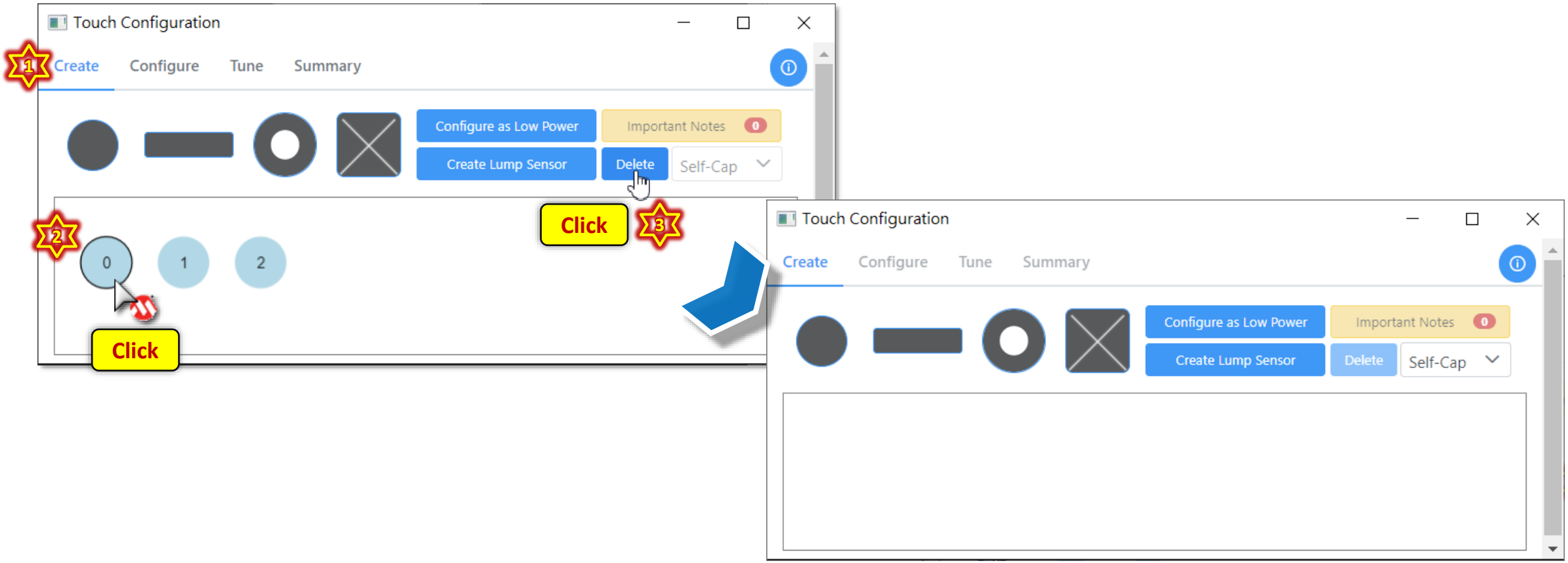
```
int main ( void )
{
    bool KeyStatus[3] = {false, false, false};
    ...

    while ( true )
    {
        touch_process();

        if(BT1_Get()==0)
        {
            while(BT1_Get()==0);
            calibrate_node(0);
            calibrate_node(1);
            calibrate_node(2);
        }

        if( measurement_done_touch == 1 )
        {
            measurement_done_touch = 0;
            switch( get_sensor_state(0) )
            {
            case QTM_KEY_STATE_DETECT:  KeyStatus[0]=true;  LED1_Set();   break;
            case QTM_KEY_STATE_NO_DET:  KeyStatus[0]=false; LED1_Clear(); break;
            }

            switch( get_sensor_state(1) )
            {
            case QTM_KEY_STATE_DETECT:  KeyStatus[1]=true;  LED2_Set();   break;
            case QTM_KEY_STATE_NO_DET:  KeyStatus[1]=false; LED2_Clear(); break;
            }
            ...
        }
    }
```

MICROCHIP

🛠 Lab2 Touch Slider

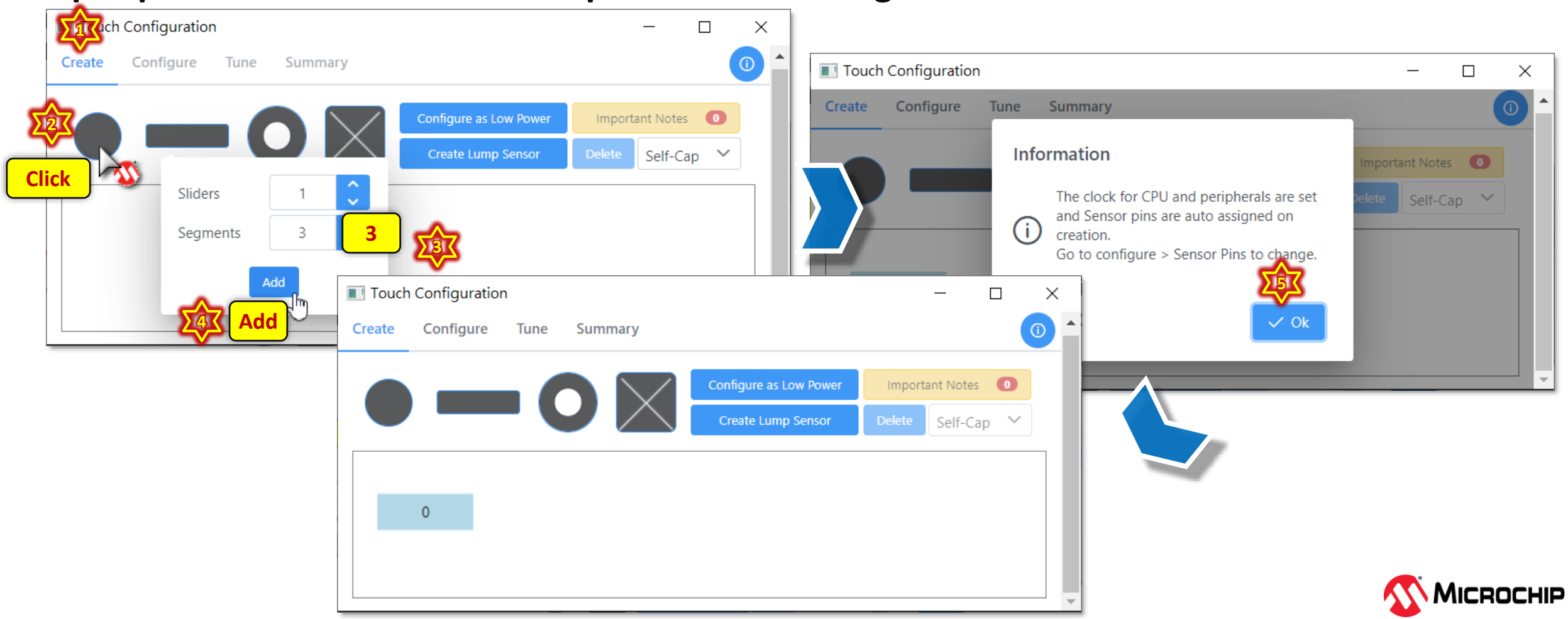
🛠 Lab2 Touch Slider

- Return Harmony Touch Configurator, if you are continuing from previous Lab1.
- In Create Tab, click on each Buttons Icon and click [Delete] to remove all of them.



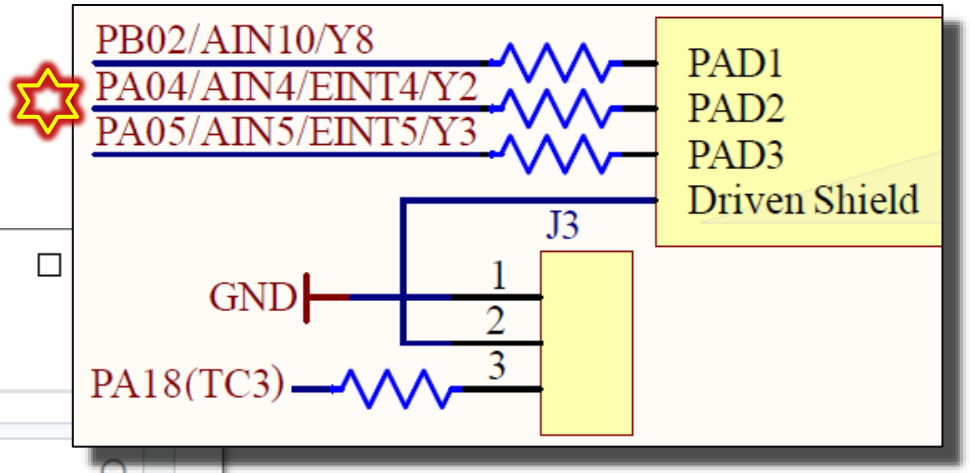
Lab2 Touch Slider

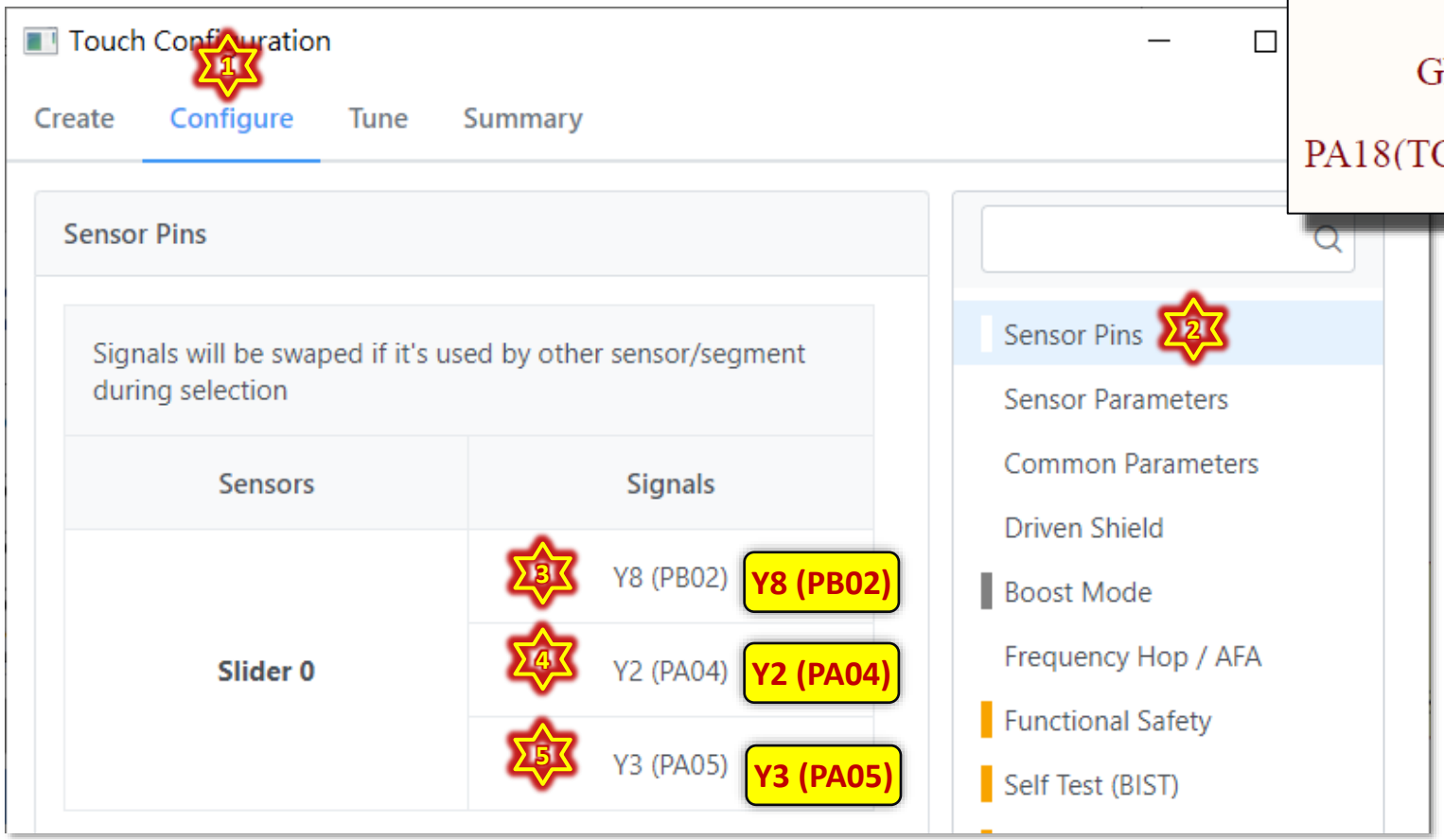
- In Create Tab, click on Slider Icon and change number Segments to 3.
- Click Add then a Warning dialog will show up to notice “The clock for CPU and peripherals are set and Sensor pins are auto assigned on creation”.



Lab2 Touch Slider

- **In Configure Tab, Click on Sensor Pins and configure Slider 0 Signals pins as below.**
  - Slider 0 (Signals) : Y8 (PB02)
  - Slider 0 (Signals) : Y2 (PA04)
  - Slider 0 (Signals) : Y3 (PA05)

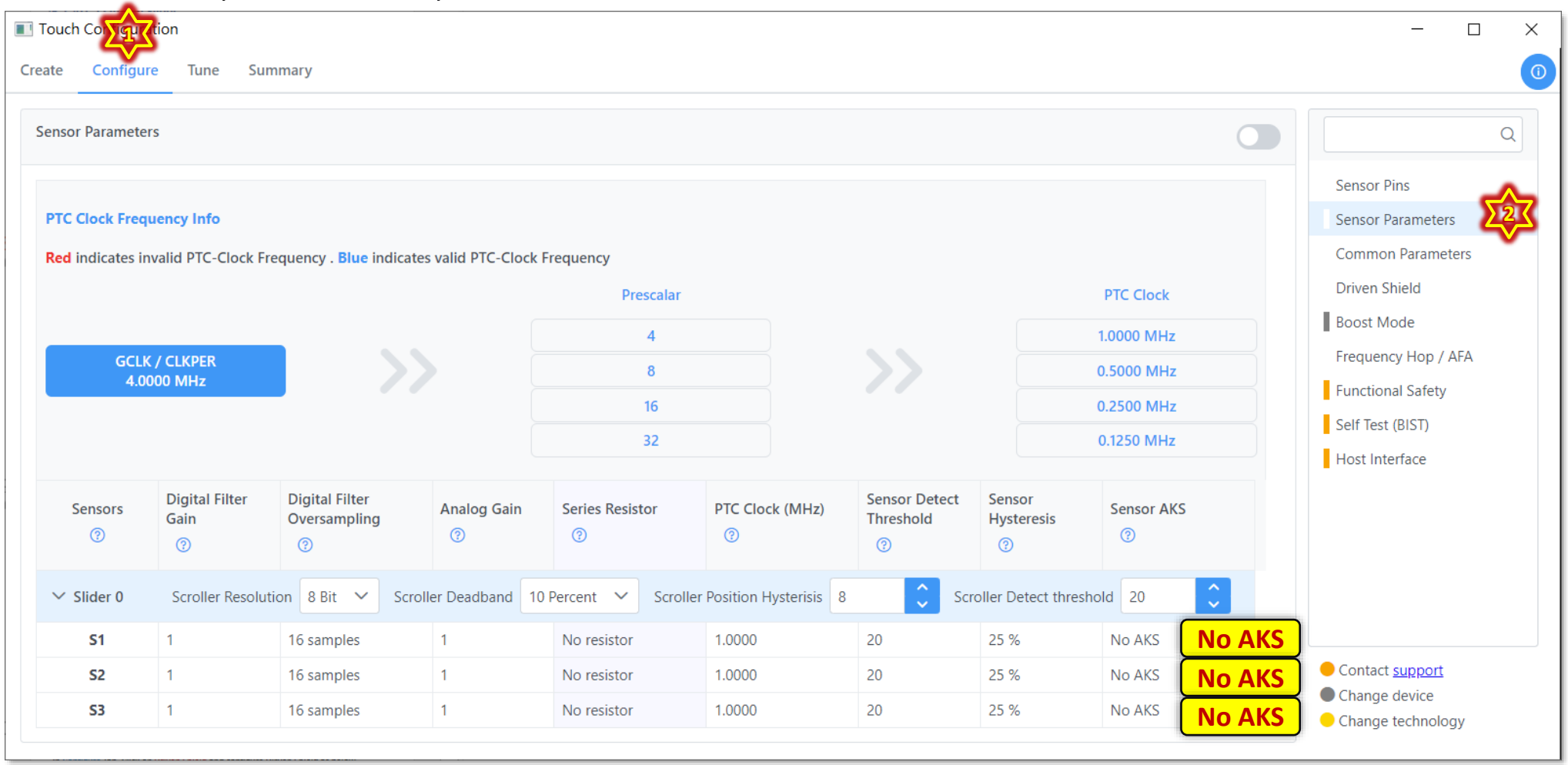




MICROCHIP

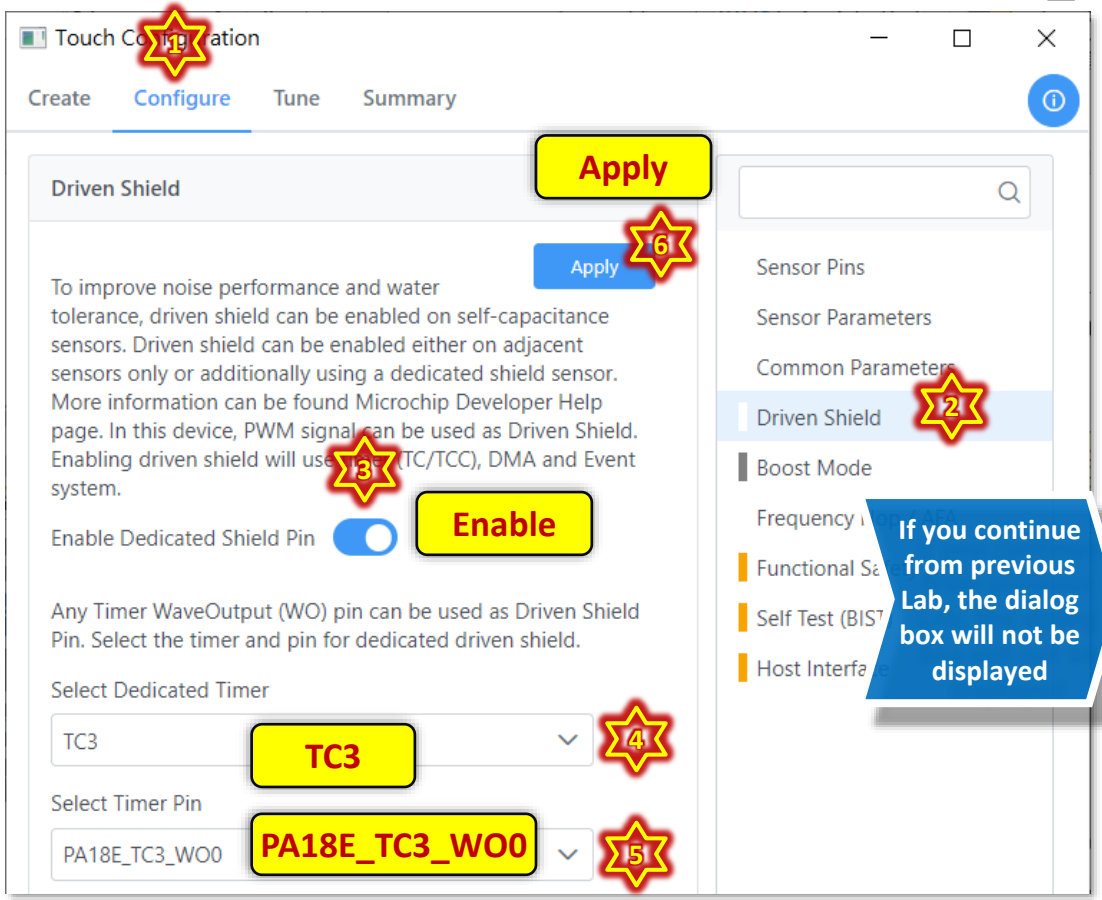
Lab2 Touch Slider

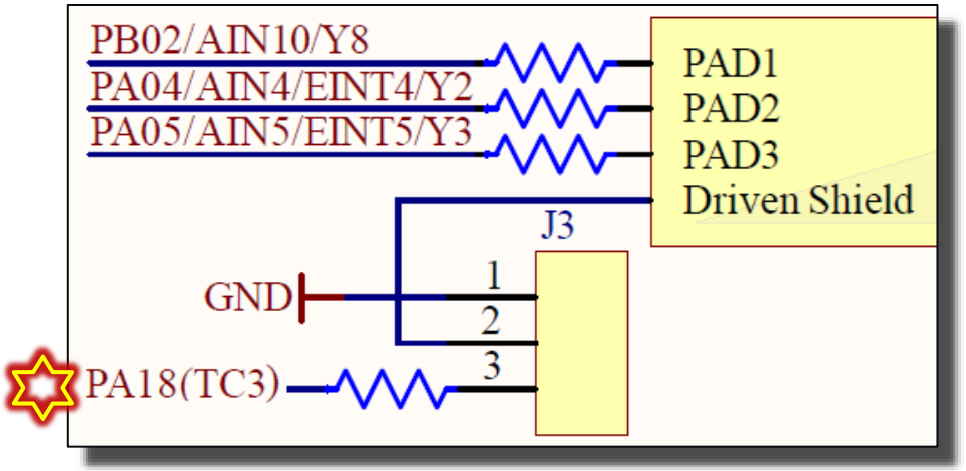
- In Configure Tab, Click on Sensor Parameters and check the configuration of Sensor AKS(Adjacent Keys Suppression) group as below.
  - S1 : no AKS, S2 : no AKS, S3 : no AKS

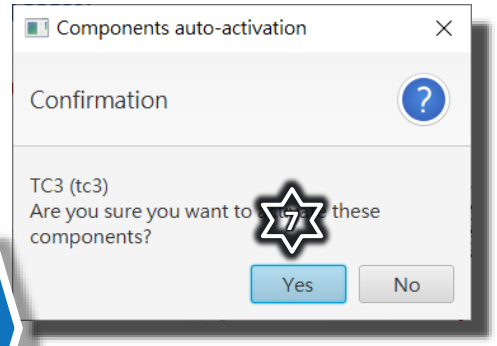


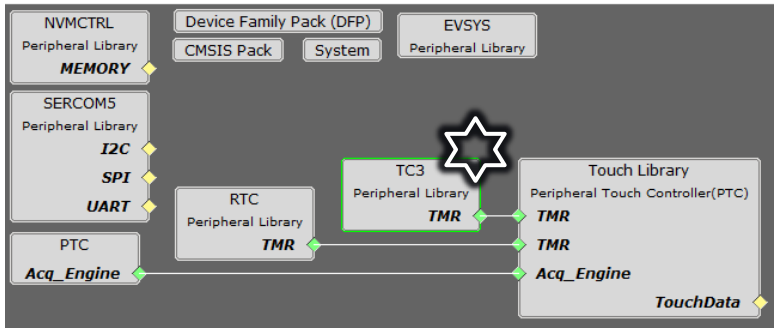
Lab2 Touch Slider

- **In Configure Tab, Click on Driven Shield and configure Dirven Shield as below.**
  - Enable Dedicated Dirven Shield : 
  - Select Dedicated Timer : TC3
  - Select Timer Pin : PA18E_TC3_WO0

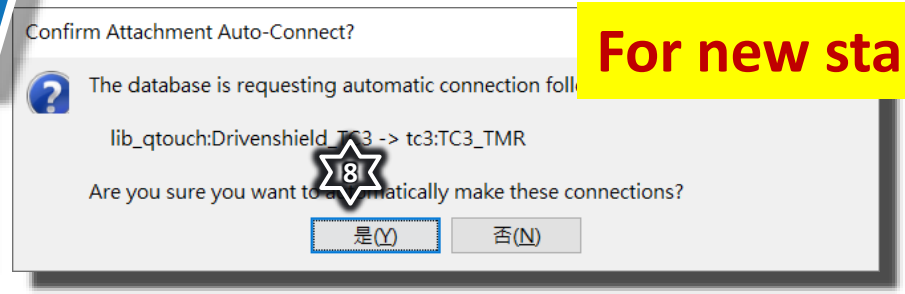








If you continue from previous Lab, the dialog box will not be displayed

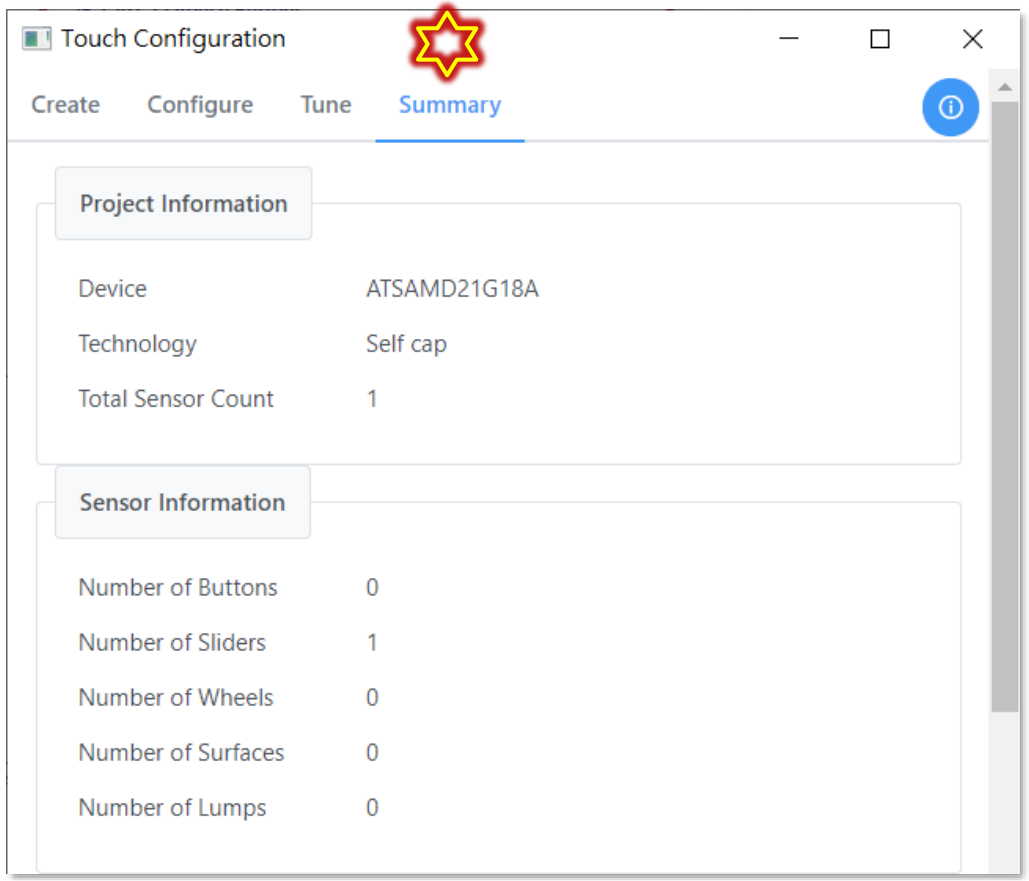


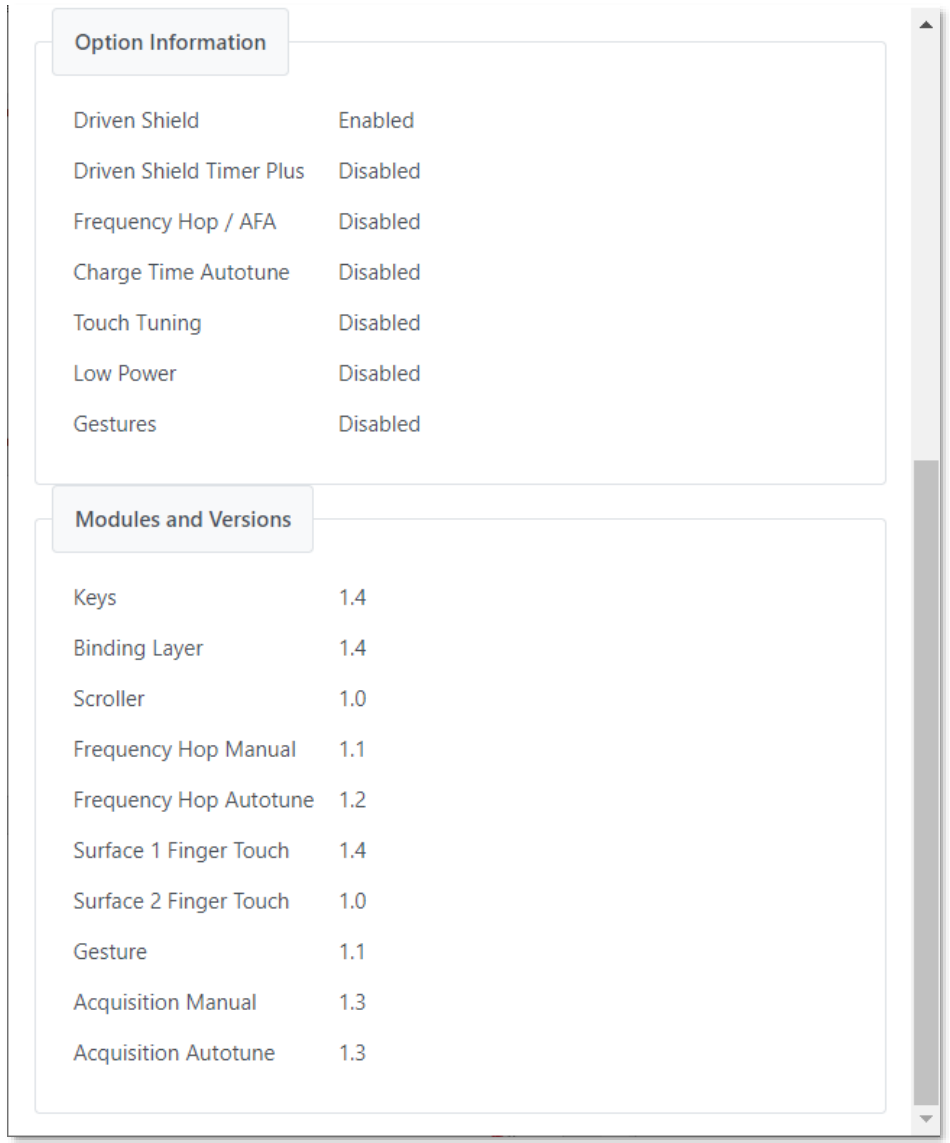
For new start project only

MICROCHIP

Lab2 Touch Slider

- In Summary Tab, to figure out current configuration of Touch.

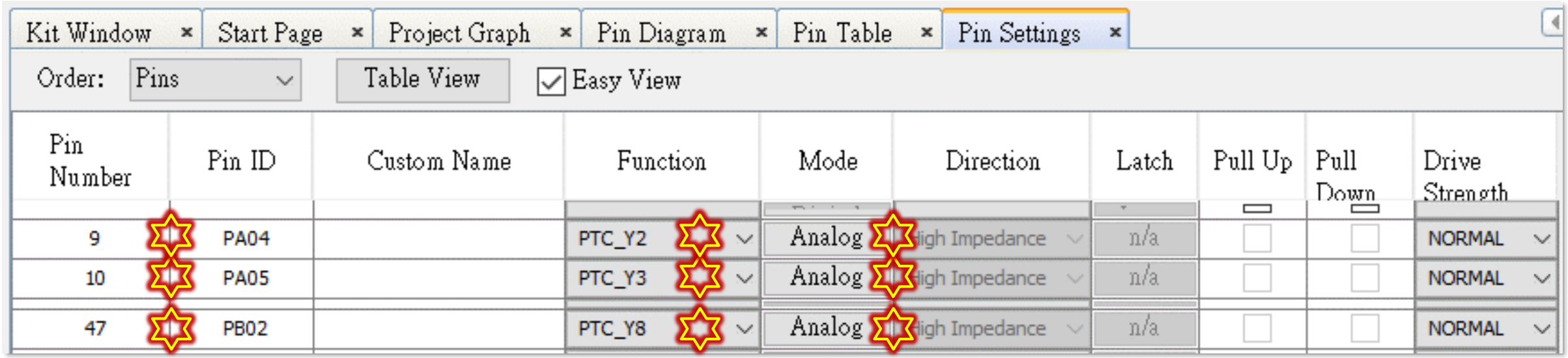




MICROCHIP

Lab2 Touch Slider

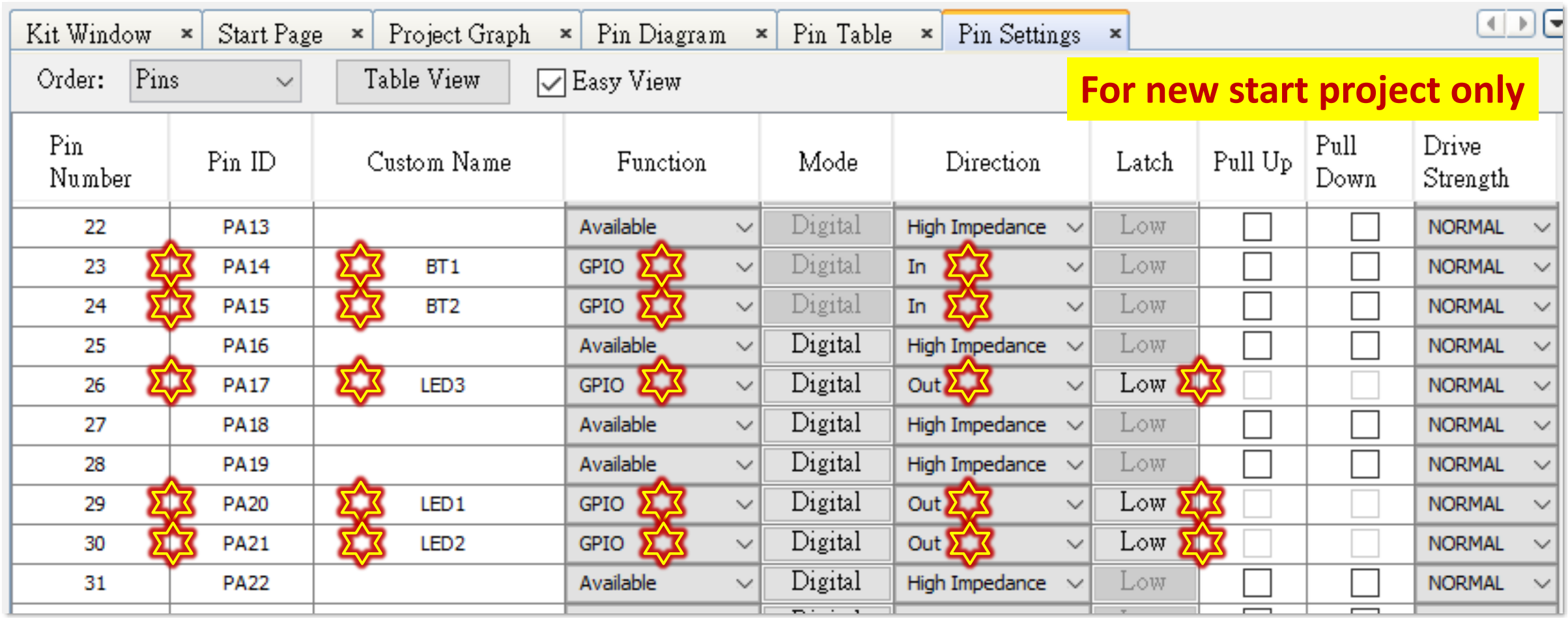
- Check the Touch Sensor Pins in Plugin **Pin Configuration** of Project Graph as below.



MICROCHIP

Lab2 Touch Slider

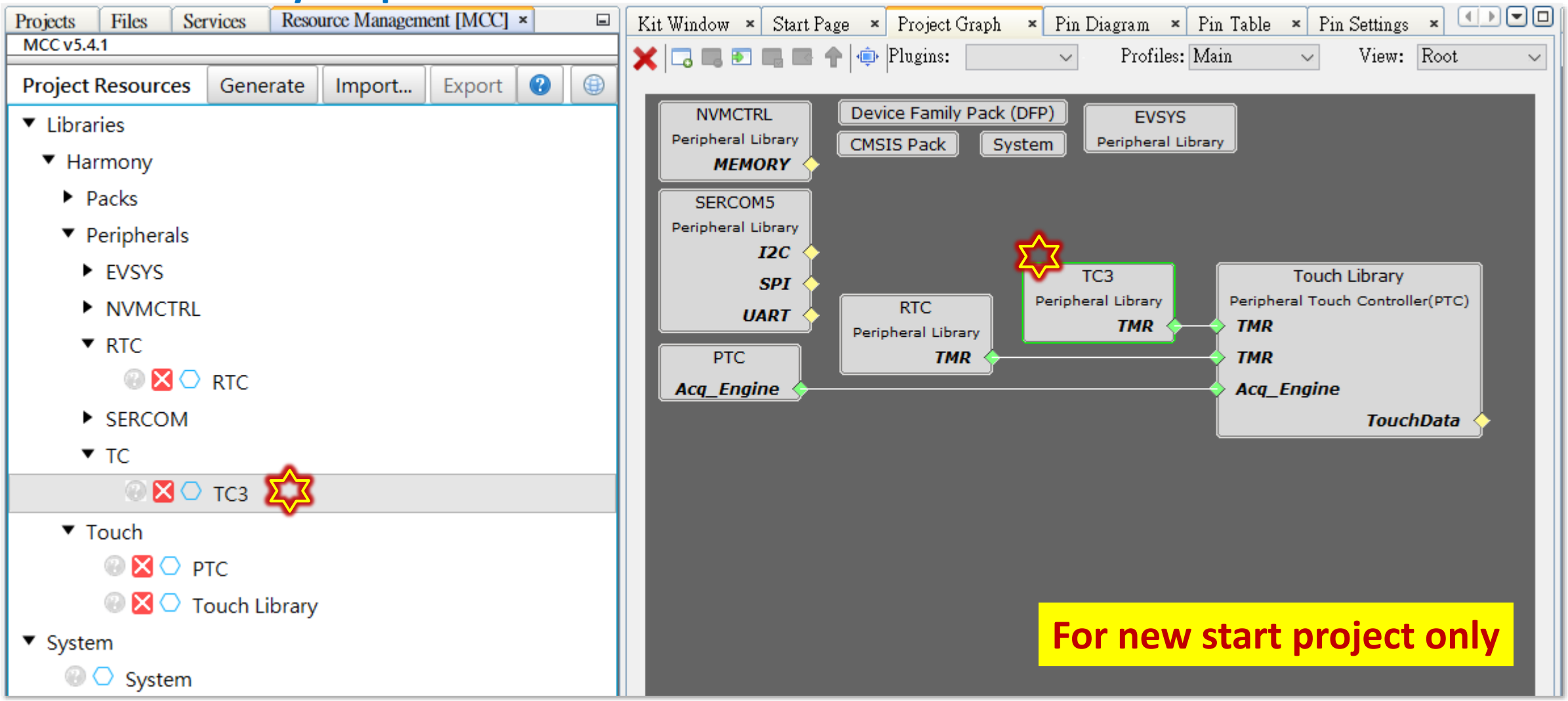
- Configure LEDs and Buttons in Plugin **Pin Configuration** of Project Graph as below.



Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
22	PA13		Available	Digital	High Impedance	Low	☐	☐	NORMAL
23	PA14	BT1	GPIO	Digital	In	Low	☐	☐	NORMAL
24	PA15	BT2	GPIO	Digital	In	Low	☐	☐	NORMAL
25	PA16		Available	Digital	High Impedance	Low	☐	☐	NORMAL
26	PA17	LED3	GPIO	Digital	Out	Low	☐	☐	NORMAL
27	PA18		Available	Digital	High Impedance	Low	☐	☐	NORMAL
28	PA19		Available	Digital	High Impedance	Low	☐	☐	NORMAL
29	PA20	LED1	GPIO	Digital	Out	Low	☐	☐	NORMAL
30	PA21	LED2	GPIO	Digital	Out	Low	☐	☐	NORMAL
31	PA22		Available	Digital	High Impedance	Low	☐	☐	NORMAL

Lab2 Touch Slider

- Check Harmony Graph to see the difference after added Driven Shield.



Lab2 Touch Slider

API functions in Touch.c

uint8_t measurement_done_touch	Touch acquisition done flag
touch_process ()	Main processing function of touch library. This function initiates the acquisition, calls post processing after the acquisition complete and sets the flag for next measurement based on the sensor status.
touch_init ()	Initialization PTC, timer, and datastreamer modules of touch library.
touch_timer_config ()	Initialization necessary RTC timer for Touch in touch_init()
touch_sensors_config ()	Initialization of touch key sensors in touch_init()
get_sensor_node_signal ()	Get sensor node aquiction signals.
update_sensor_node_signal ()	Update sensor node aquiction signals.
get_sensor_node_reference ()	Get sensor node channel reference.
update_sensor_node_reference ()	Update sensor node channel reference.
get_sensor_cc_val ()	Get sensor node compansation capacitor value.
update_sensor_cc_val ()	Update sensor node compansation capacitor value.
get_sensor_state ()	Get sensor node state
update_sensor_state ()	Update sensor node state
calibrate_node ()	Sensor node calibration
get_scroller_state	Get scroller sensor state
get_scroller_position	Get scroller position

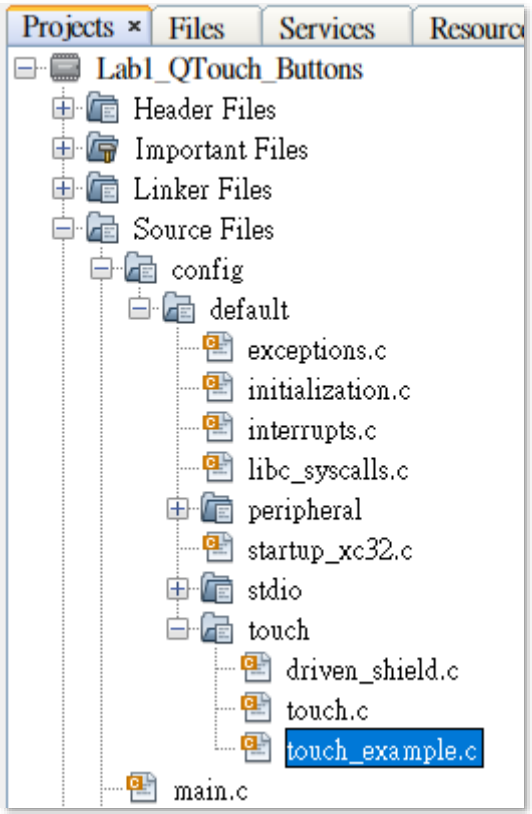
API functions in driven_shield.c

drivenshield_configure ( )	Iinitialization Driven Shield for Touch in touch_init()

MICROCHIP

Lab2 Touch Slider

- **Example functions in touch_example.c depend to sensors you to configured.**




```c
#include "touch_example.h"

void touch_mainloop_example(void){

    /* call touch process function */
    touch_process();

    if(measurement_done_touch == 1u)
    {
        measurement_done_touch = 0u;
        // process touch data
    }
}
```

```c
    scroller_status = get_scroller_state(0);
    scroller_position = get_scroller_position(0);
    //Example: 8 bit scroller resolution. Modify as
    scroller_position = scroller_position  >> 5u;
    //LED_OFF
    if ( 0u != scroller_status) {
        switch (scroller_position) {
        case 0:
            //LED0_ON
            break;
        case 1:
            //LED1_ON
            break;
...

        case 7:
            //LED7_ON
            break;
        default:
            //LED_OFF
            break;
        }
    }
}
```

```c
/*============================================================
void touch_status_display(void)
------------------------------------------------------------
----------
Purpose: Sample code snippet to demonstrate how to check the status
of the
         sensors
Input  : none
Output : none
Notes  : none
============================================================*/
void touch_status_display(void)
{
uint8_t key_status = 0u;
uint8_t scroller_status = 0u;
uint16_t scroller_position = 0u;
    key_status = get_sensor_state(0) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }

    key_status = get_sensor_state(1) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }

    key_status = get_sensor_state(2) & KEY_TOUCHED_MASK;
    if (0u != key_status) {
        //Touch detect
    } else {
        //Touch No detect
    }
```

MICROCHIP

Lab2 Touch Slider

- Implement below function in main.c to make sure your UART console is ready for use.

```
#include <string.h>
#include <stdio.h>
#include <stdarg.h>

extern volatile uint8_t measurement_done_touch;

#define SYS_CONSOLE_PRINT_BUFFER_SIZE   200
static char consolePrintBuffer[SYS_CONSOLE_PRINT_BUFFER_SIZE];
void myprintf(const char *format, ...)
{
    size_t len = 0;
    va_list args = {0};

    va_start(args, format);
    len = vsnprintf(consolePrintBuffer, SYS_CONSOLE_PRINT_BUFFER_SIZE, format, args);
    va_end(args);

    if ((len > 0) && (len < SYS_CONSOLE_PRINT_BUFFER_SIZE))
    {
        consolePrintBuffer[len] = '\0';
        SERCOM5_USART_Write((uint8_t*)consolePrintBuffer, len);
        while (SERCOM5_USART_WriteIsBusy());
    }
}

int main ( void )
{
    SYS_Initialize ( NULL );

    myprintf("\033[2J");
    myprintf("\033[1;1H");
    myprintf("\033[?25l");
    myprintf("------------------\r\n");
    myprintf("  Touch Slider\r\n");
    myprintf("------------------\r\n");
    myprintf("____________________");
    ...
```

Lab2 Touch Slider

- **Implement below function in while loop of main.c to make Touch Slider works.**

```
int main ( void )
{
    uint16_t ScrollPos, PreScrollPos=256;

    SYS_Initialize ( NULL );
    ...

    while ( true )
    {
        touch_process();

        if( measurement_done_touch==1 && get_scroller_state(0) )
        {
            measurement_done_touch = 0;
            ScrollPos = get_scroller_position(0);
            if( ScrollPos != PreScrollPos )
            {
                PreScrollPos = ScrollPos;
                myprintf("\033[4;1H");
                myprintf("__________________");
                myprintf("\033[4;%dH", (ScrollPos*17/255)+1);
                myprintf("O");
                myprintf("\033[5;1H");
                myprintf("Slider Pos = %03d", ScrollPos);

                switch( ScrollPos*4/255 )
                {
                case 0: LED1_Set();   LED2_Clear(); LED3_Clear(); break;
                case 1: LED1_Set();   LED2_Set();   LED3_Clear(); break;
                case 2: LED1_Clear(); LED2_Set();   LED3_Clear(); break;
                case 3: LED1_Clear(); LED2_Set();   LED3_Set();   break;
                case 4: LED1_Clear(); LED2_Clear(); LED3_Set();   break;
                }
            }
        }
        ...
    }
```

Lab2 Touch Slider

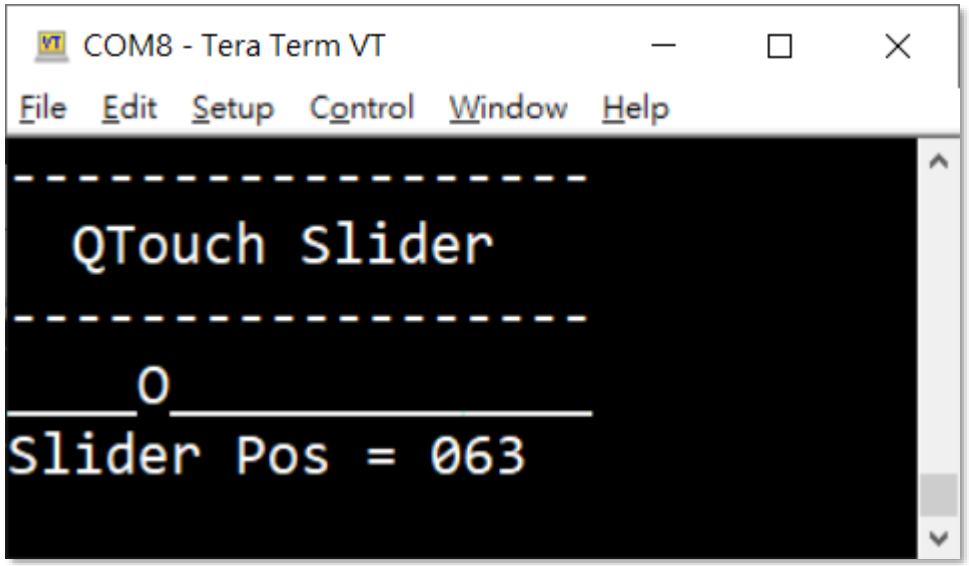
- **Implement manual Touch Calibration feature.**

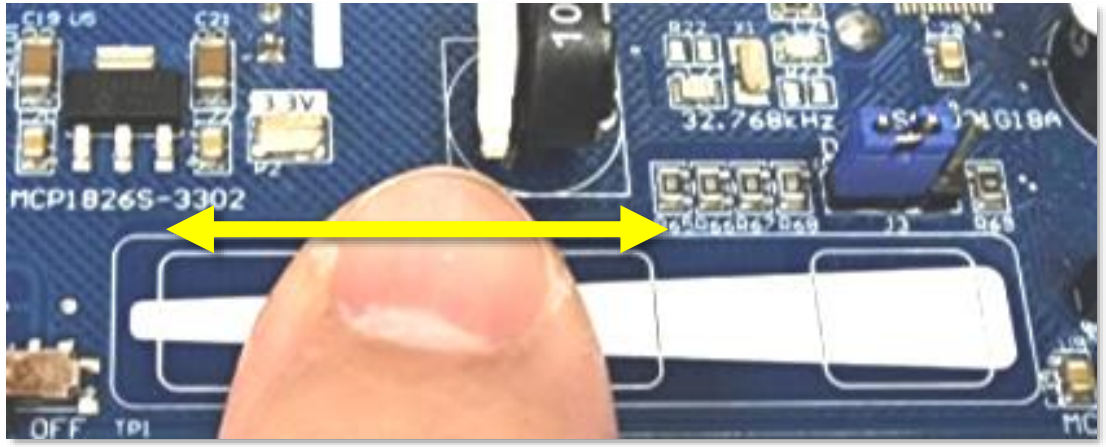
```
int main ( void )
{
    uint16_t ScrollPos, PreScrollPos=256;
    ...

    while ( true )
    {
        touch_process();

        if(BT1_Get()==0)
        {
            while(BT1_Get()==0);
            calibrate_node(0);
            calibrate_node(1);
            calibrate_node(2);
        }

        if( measurement_done_touch==1 && (get_scroller_state(0) & KEY_TOUCHED_MASK) )
        {
            measurement_done_touch = 0;
            ScrollPos = get_scroller_position(0);
            if( ScrollPos != PreScrollPos )
            {
                PreScrollPos = ScrollPos;
                myprintf("\033[4;1H");
                myprintf("__________________");
                myprintf("\033[4;%dH", (ScrollPos*17/255)+1);
                myprintf("O");
                myprintf("\033[5;1H");
                myprintf("Slider Pos = %03d", ScrollPos);

                switch( ScrollPos*4/255 )
                {
                case 0: LED1_Set();   LED2_Clear(); LED3_Clear(); break;
                case 1: LED1_Set();   LED2_Set();   LED3_Clear(); break;
                case 2: LED1_Clear(); LED2_Set();   LED3_Clear(); break;
                ...
```

MICROCHIP

🛠 Lab2 Touch Slider

- **Result of Touch Slider(Scroller) implementation.**



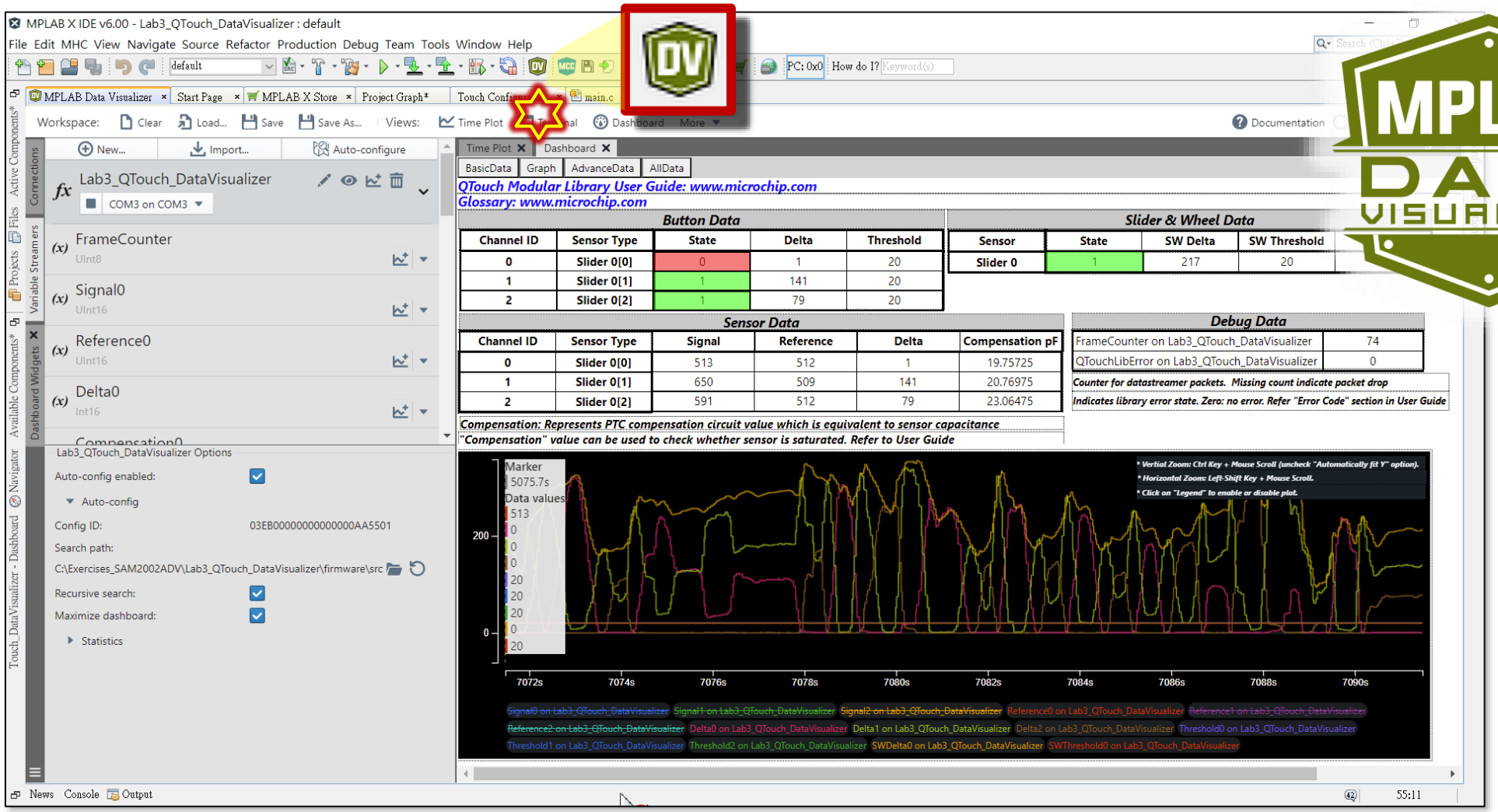


MICROCHIP

🛠 Lab3 Data Visualizer for Touch

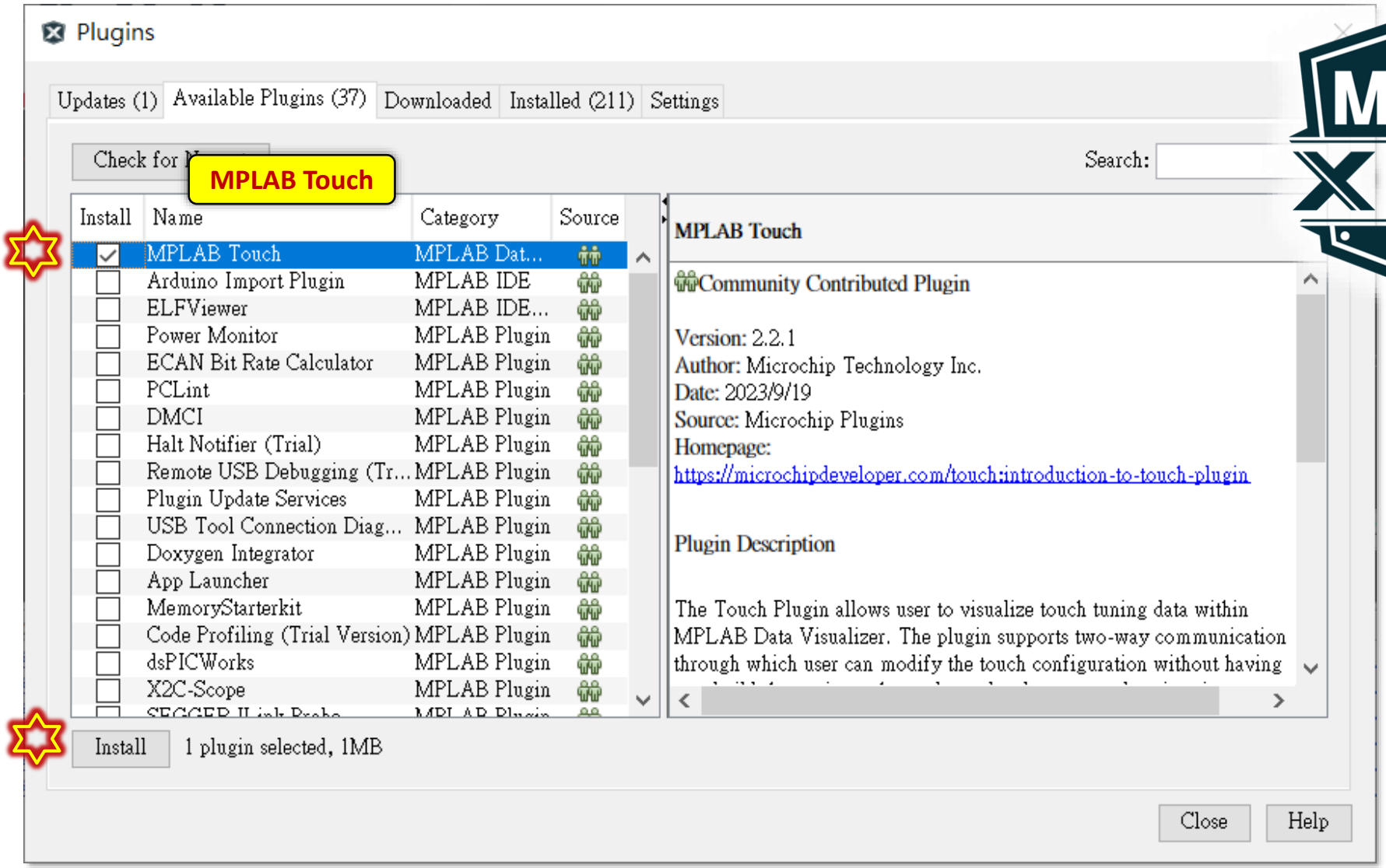
Lab3 Data Visualizer for Touch

- **MPLAB® Data Visualizer could support to Touch Visualization and Tuning through UART.**



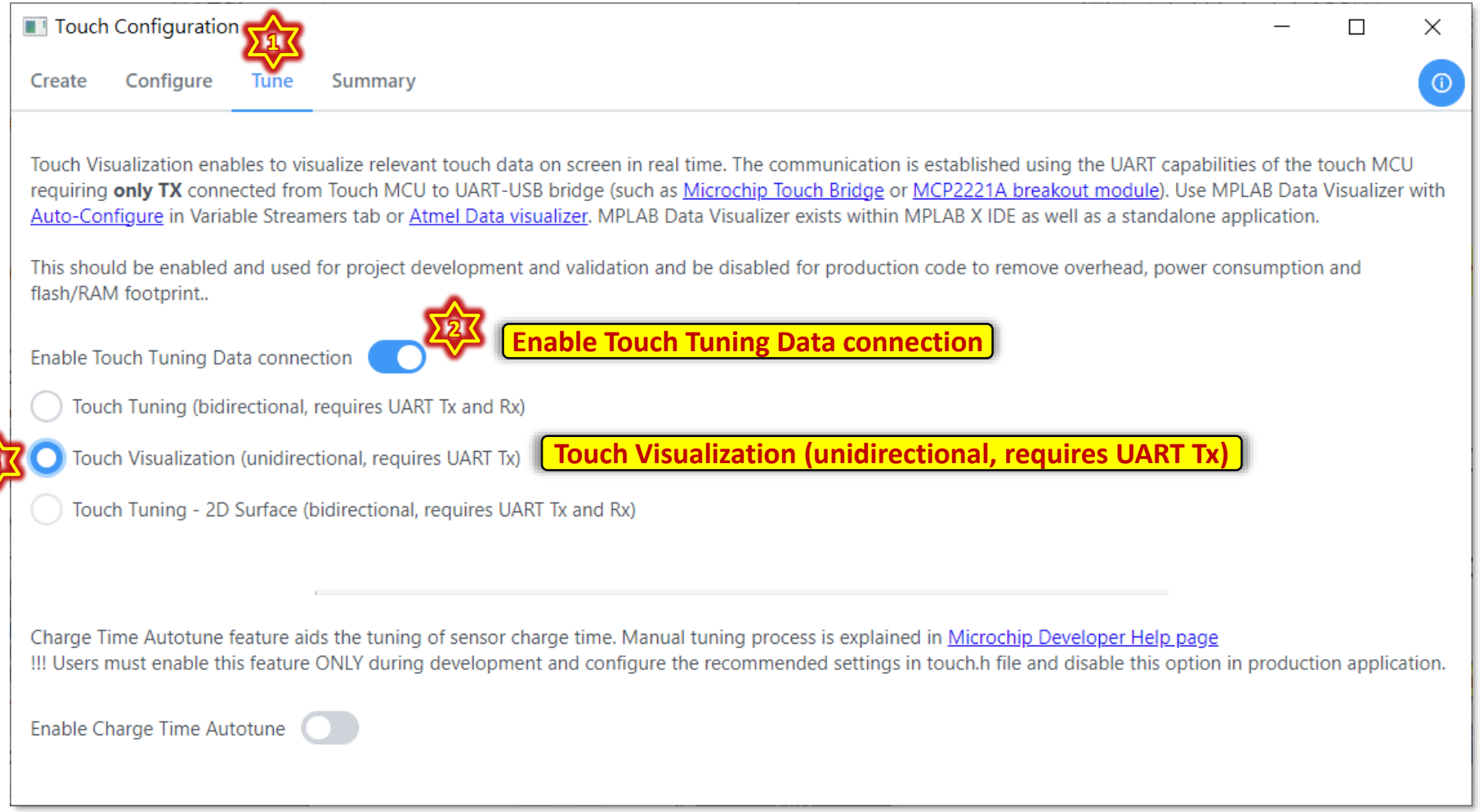
Lab3 Data Visualizer for Touch

- **Before use Data Visualizer for touch tuning, it's need install extra Touch plugin firstly.**



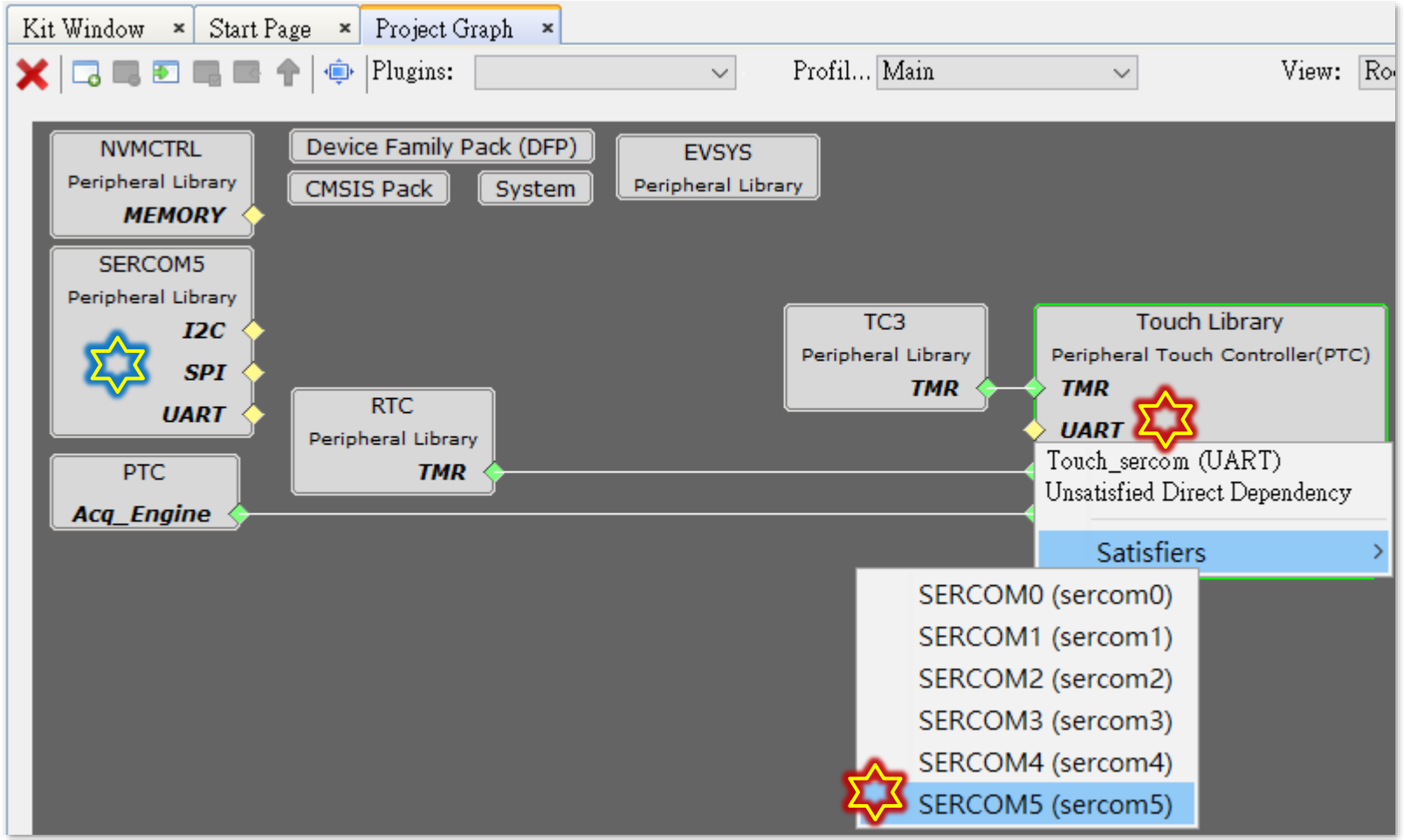
🛠 Lab3 Data Visualizer for Touch

- **Return Harmony Touch Configurator, if you are continuing from previous Lab2.**
  In Tune Tab, Please switch to Enable Touch Tuning Data connection and choice Touch Visualization (unidirectional, requires UART Tx)



🛠 Lab3 Data Visualizer for Touch

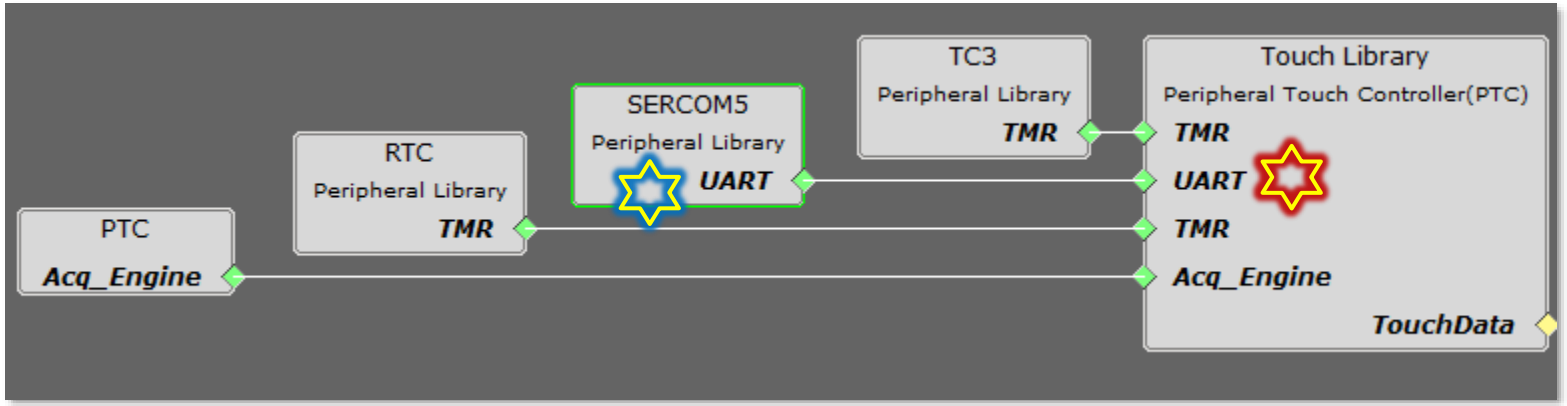
- A new **UART** connection interface will appear on **Touch Library** module after enabled the Tuning feature in Touch Configurator.
- Assign PLIB **SERCOM5** to **UART** interface of **Touch Library**.



MICROCHIP

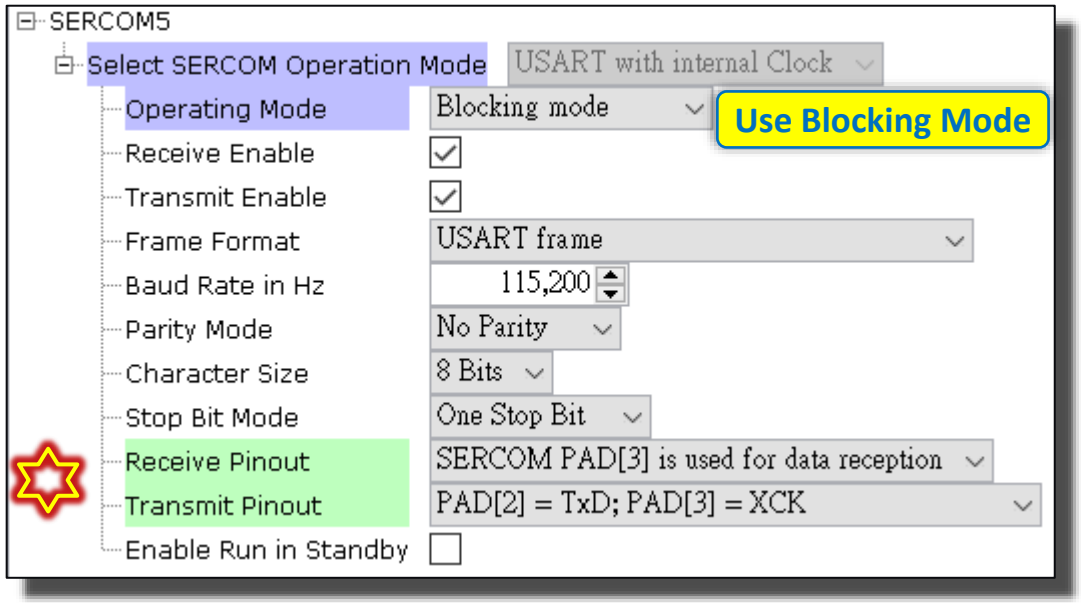
Lab3 Data Visualizer for Touch

- **Configure PLIB SERCOM5-UART as below, if you are not continuing from previous Lab2.**



For new start project only

Pin Number	Pin ID	Custom Name	Function
37	PB22		SERCOM5_PAD2
38	PB23		SERCOM5_PAD3



🛠 Lab3 Data Visualizer for Touch

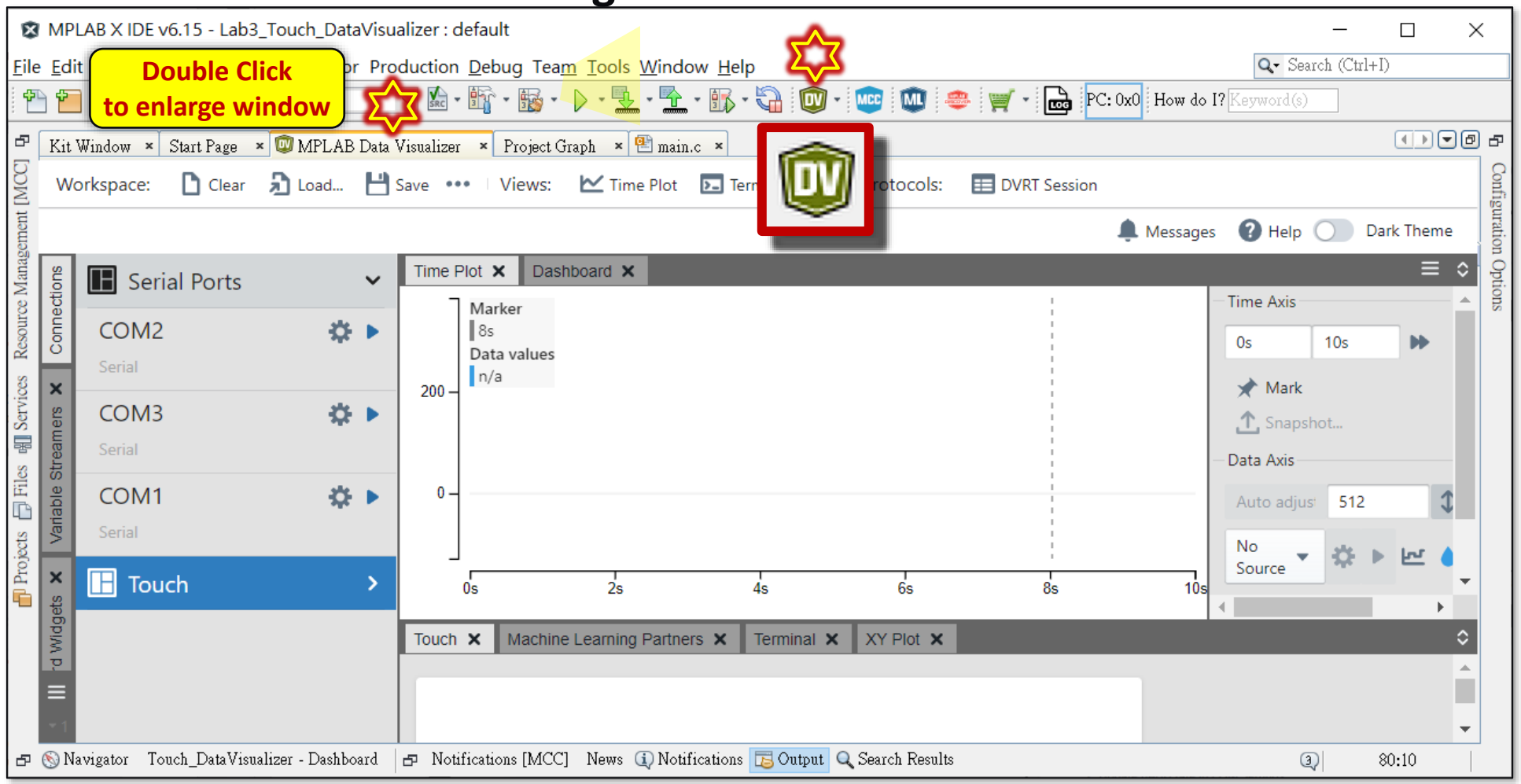
- Modify myprintf() to use TransmitComplete() function of SERCOM Blocking Mode.
- Common out all debug myprintf() in while loop of main.c.

```
...

void myprintf(const char *format, ...)
{
    ...
    if ((len > 0) && (len < SYS_CONSOLE_PRINT_BUFFER_SIZE))
    {
        consolePrintBuffer[len] = '\0';
        SERCOM5_USART_Write((uint8_t*)consolePrintBuffer, len);
        //while (SERCOM5_USART_WriteIsBusy());
        while( !SERCOM5_USART_TransmitComplete() );
    }
}

int main ( void )
{   ...

    while ( true )
    {
        touch_process();

        if( measurement_done_touch==1 && get_scroller_state(0) )
        {   ...
            if( ScrollPos != PreScrollPos )
            {
                PreScrollPos = ScrollPos;
                //myprintf("\033[4;1H");
                //myprintf("____________________");
                //myprintf("\033[4;%dH", (ScrollPos*17/255)+1);
                //myprintf("O");
                //myprintf("\033[5;1H");
                //myprintf("Slider Pos = %03d", ScrollPos);

                switch( ScrollPos*4/255 )
```

MICROCHIP

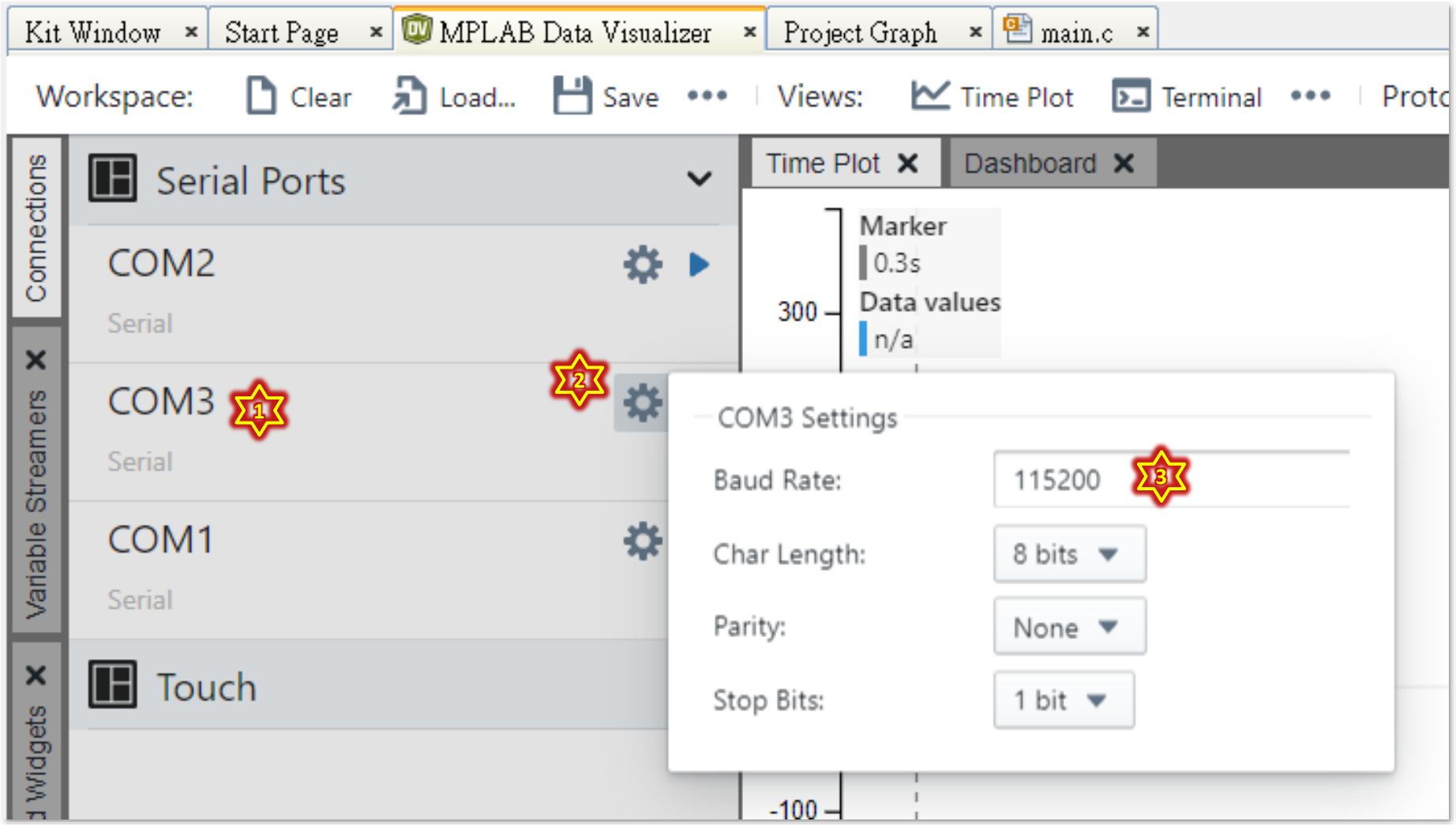
Lab3 Data Visualizer for Touch

- Open MPLAB® Data Visualizer by simple click on icon , you could doble click on MPLAB Data Visualizer tab to enlarge it to full window as below.



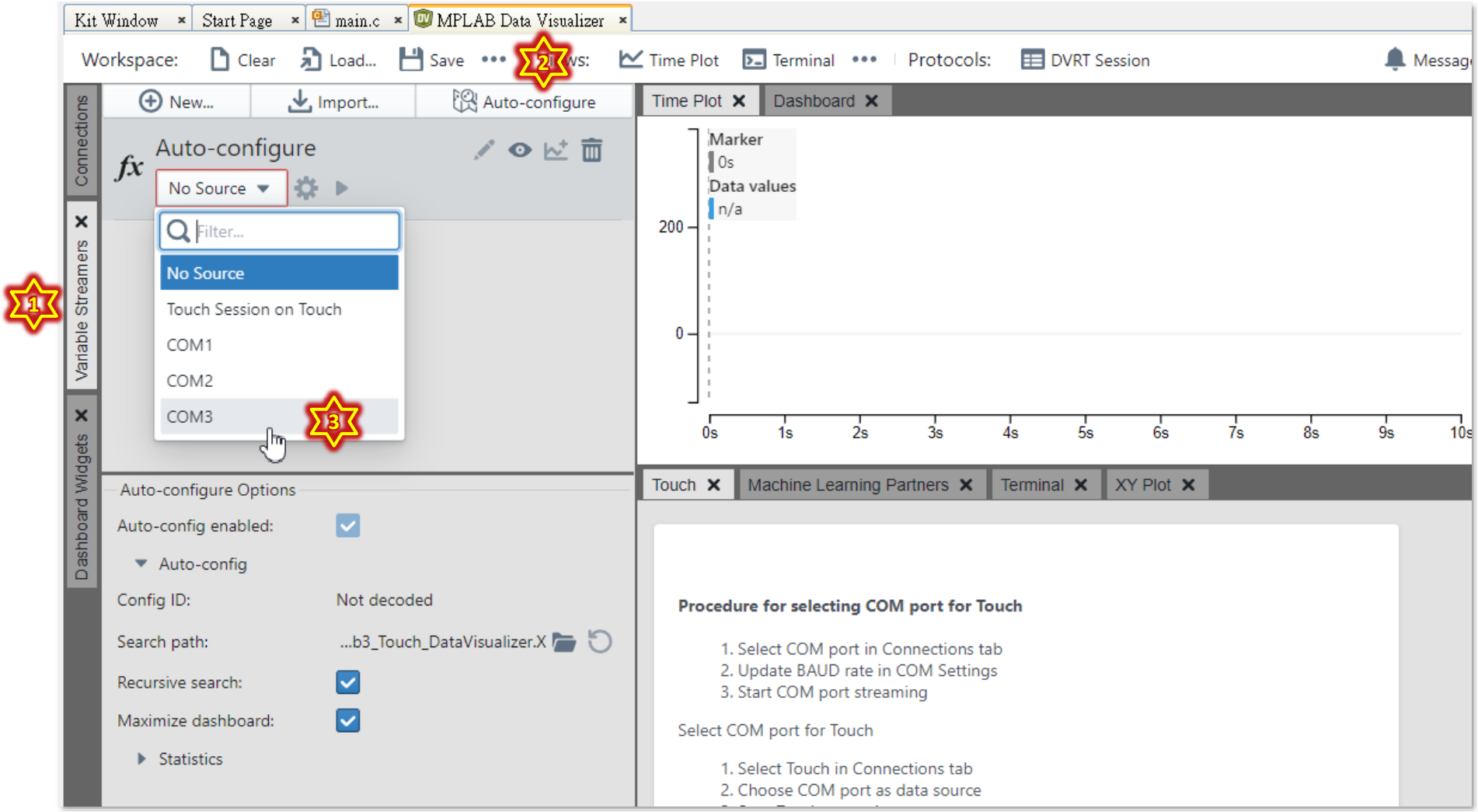
🛠 Lab3 Data Visualizer for Touch

- **In Connections Tab, click on the UART COM Port you to output debug message from SERCOM5 and configure to 115200-n-8-1.**



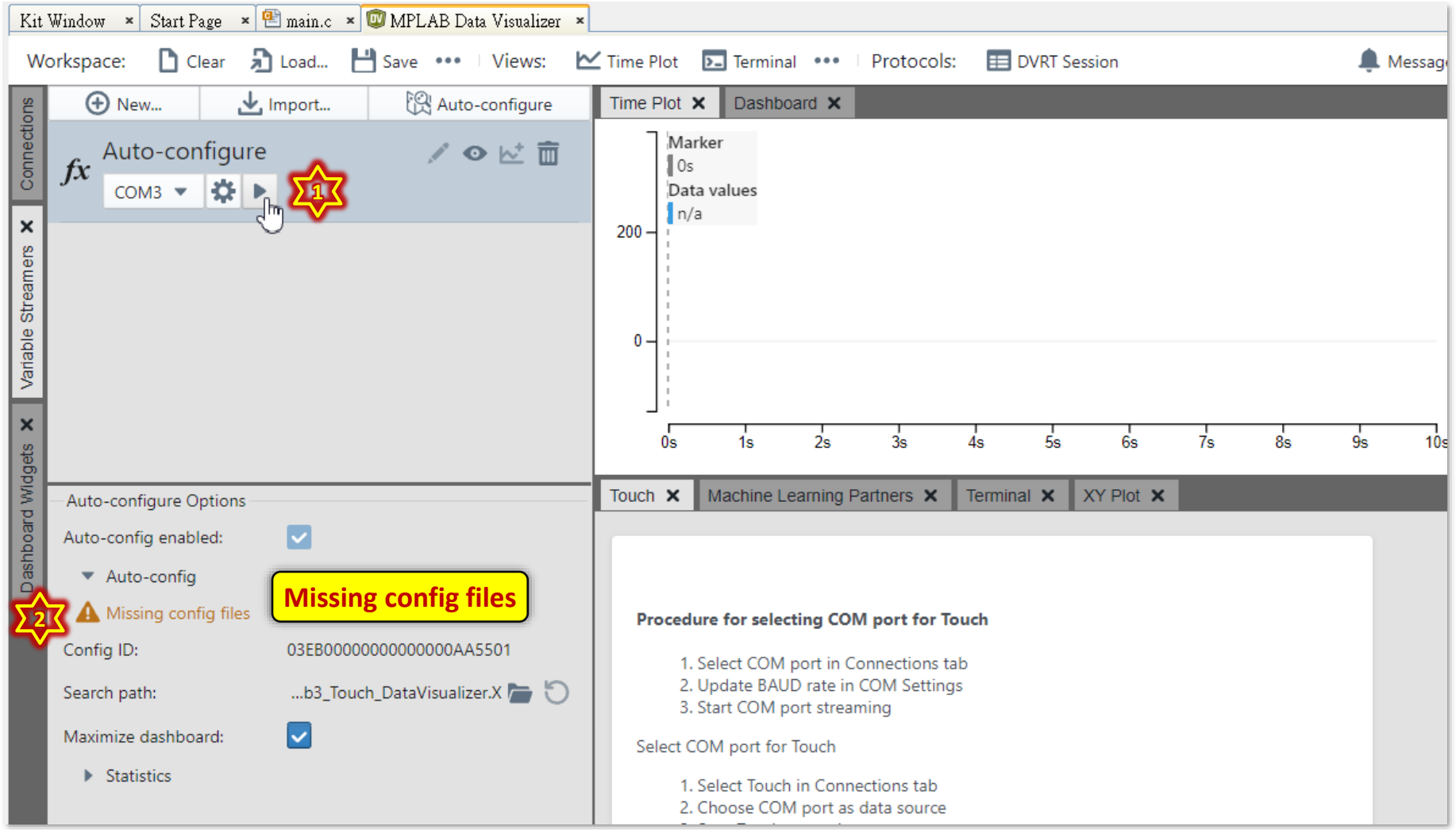
🛠 Lab3 Data Visualizer for Touch

- Click on Variable Streamers tab, then click the Auto-configure to add a new Streamer.
- Select the same COM Port as Auto-configuration source.



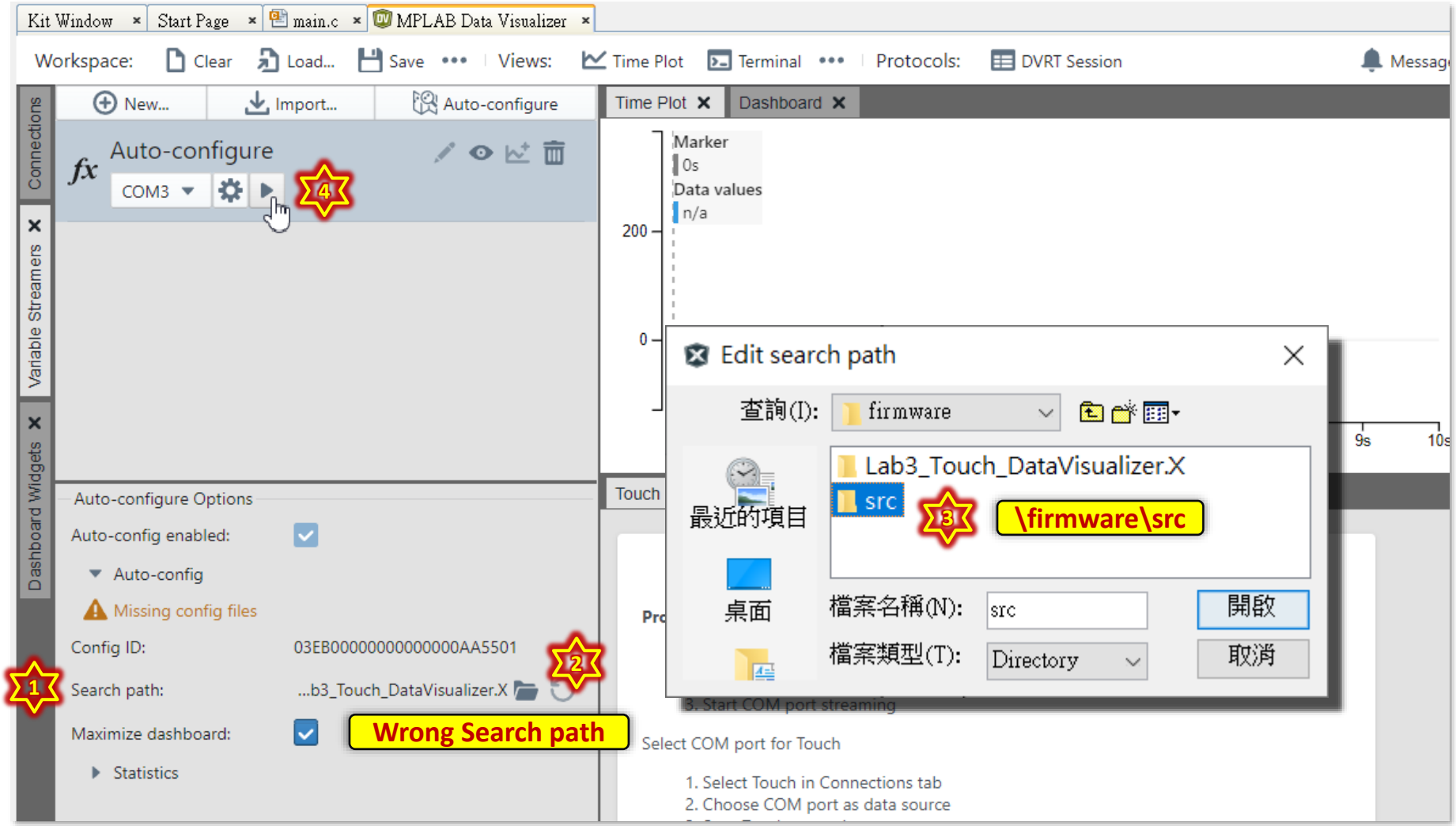
Lab3 Data Visualizer for Touch

- Click on  to start communicate then you will see a **Miss configure files** message.



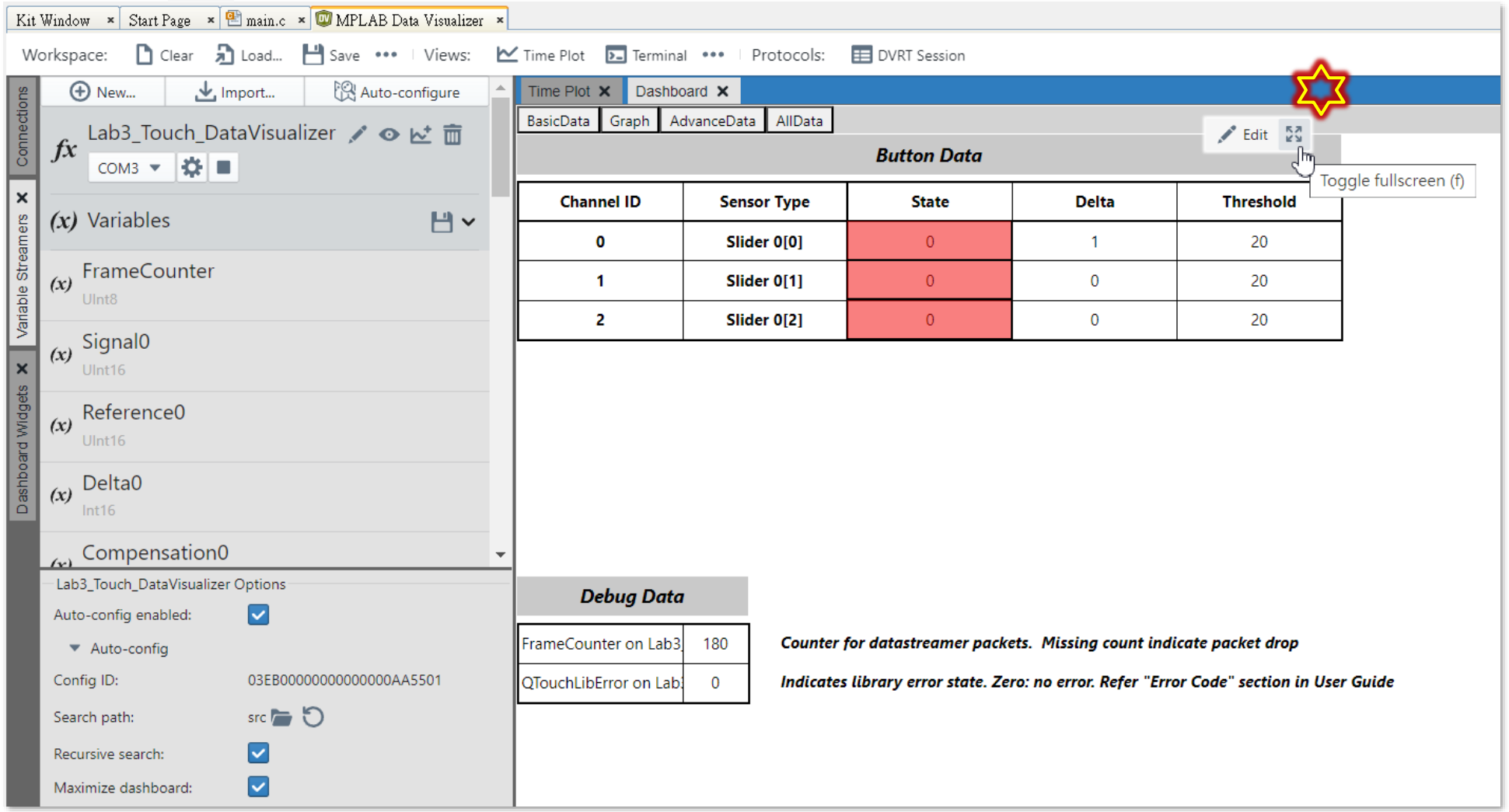
Lab3 Data Visualizer for Touch

- **The default Auto-configuration Options Search Path is wrong, please click on Icon to change folder to \firmware\src then click again.**



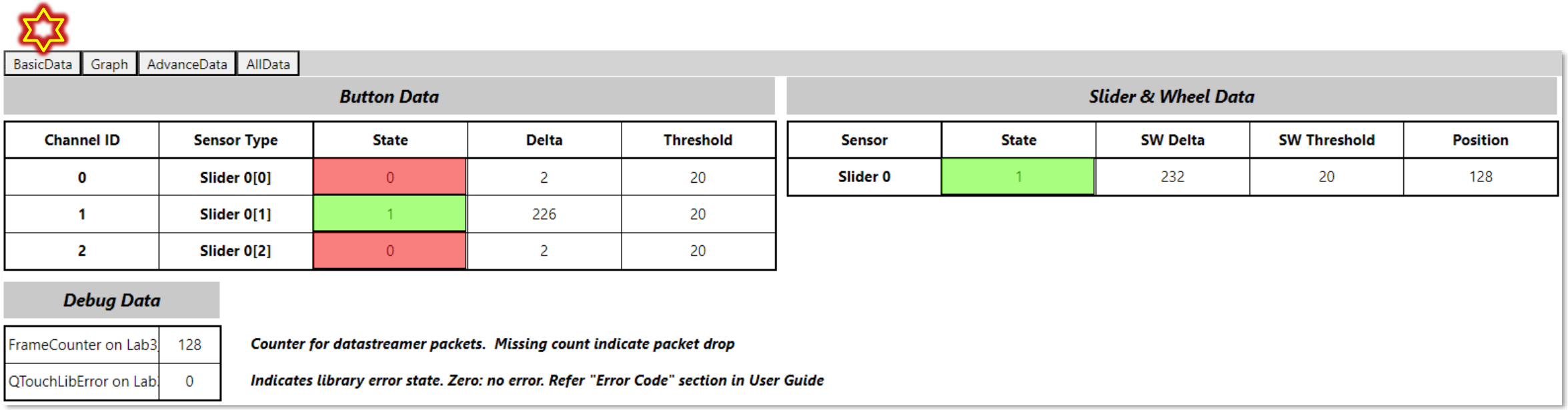
Lab3 Data Visualizer for Touch

- If all configuration are correctly, you will see a Touch Dashboard to start up.
- Click on Icon to enlarge Dashboard window.



Lab3 Data Visualizer for Touch

- **Basic Data** window of Touch Dashboard.



BasicData | Graph | AdvanceData | AllData

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Slider 0[0]	0	2	20
1	Slider 0[1]	1	226	20
2	Slider 0[2]	0	2	20

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	1	232	20	128

Debug Data

FrameCounter on Lab3	128
QTouchLibError on Lab	0

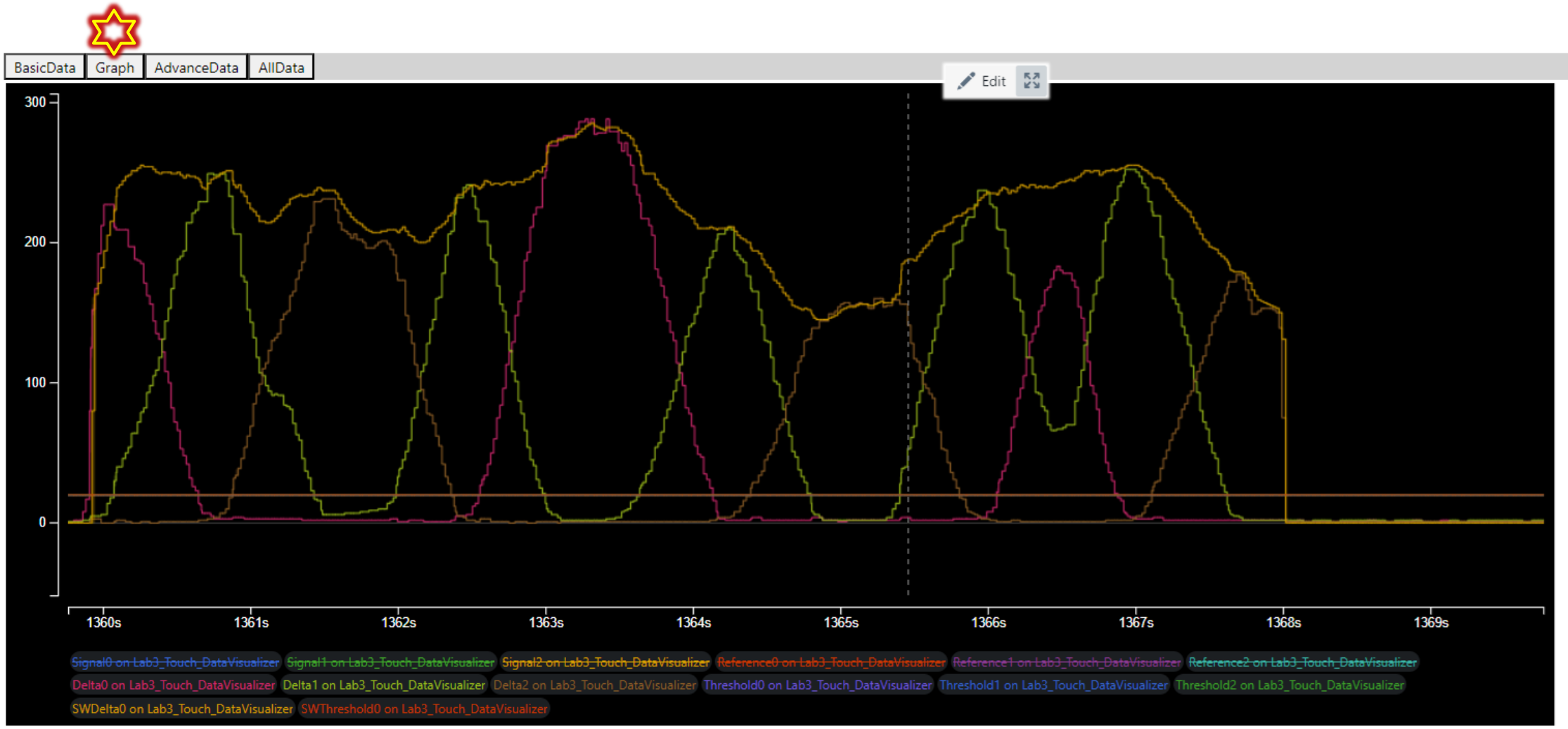
Counter for datastreamer packets. Missing count indicate packet drop

Indicates library error state. Zero: no error. Refer "Error Code" section in User Guide

MICROCHIP

Lab3 Data Visualizer for Touch

- **Graph** window of Touch Dashboard.



** Vertial Zoom: Ctrl Key + Mouse Scroll (uncheck "Automatically fit Y" option).*

** Horizontal Zoom: Left-Shift Key + Mouse Scroll.*

** Click on "Legend" to enable or disable plot.*

MICROCHIP

Lab3 Data Visualizer for Touch

- **Advance Data window of Touch Dashboard.**

BasicData | Graph | AdvanceData | AllData

Edit

Sensor Data

Channel ID	Sensor Type	Signal	Reference	Delta	Compensation pF
0	**Slider 0[0]**	508	508	0	17.39475
1	**Slider 0[1]**	507	506	1	17.93475
2	**Slider 0[2]**	507	507	0	21.78225

Compensation: Represents PTC compensation circuit value which is equivalent to sensor capacitance

"Compensation" value can be used to check whether sensor is saturated. Refer to User Guide

Frequency Hop Data

CurrentFrequency on L	10	HopFrequency0 on Lab	9	HopFrequency1 on Lab	10	HopFrequency2 on Lab	0

Displayed frequencies are auto-tuned by QTouch Library based on noise levels

MICROCHIP

Lab3 Data Visualizer for Touch

- **All Data** window of Touch Dashboard.

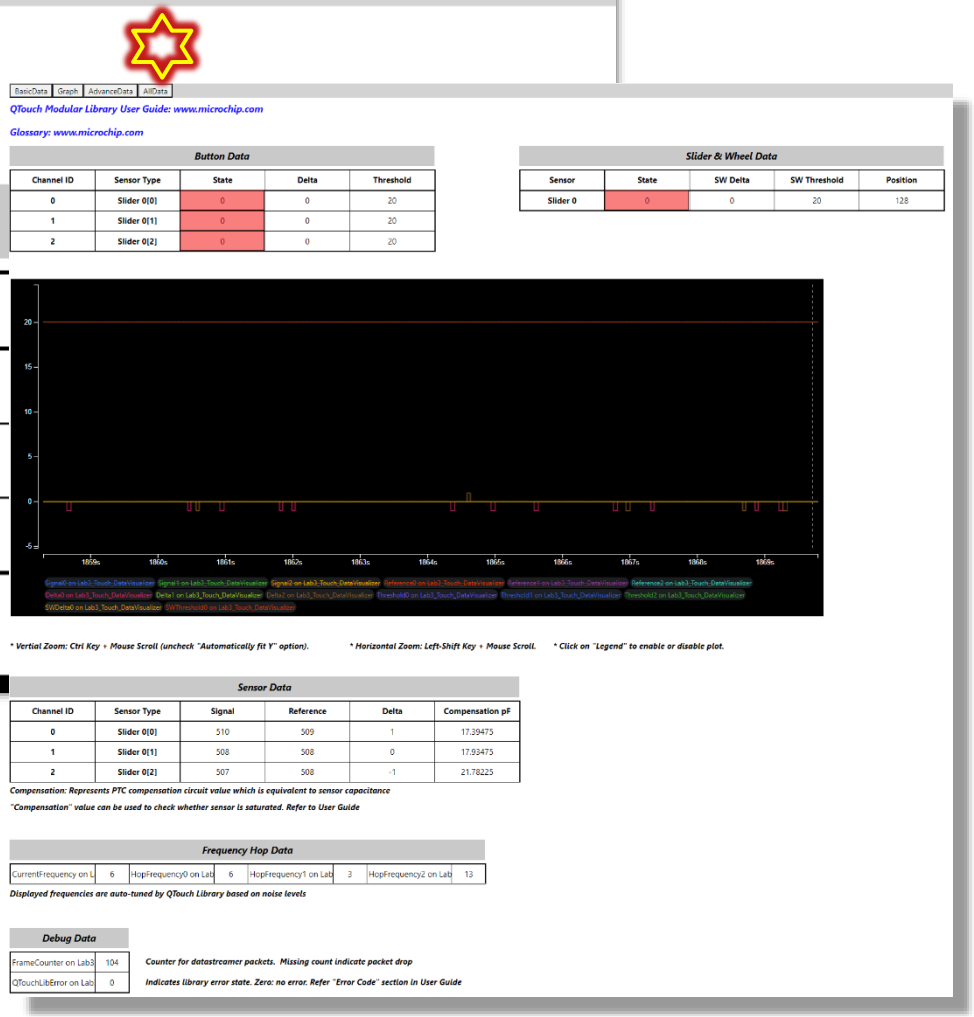
BasicData | Graph | AdvanceData | AllData

QTouch Modular Library User Guide: www.microchip.com

Glossary: www.microchip.com

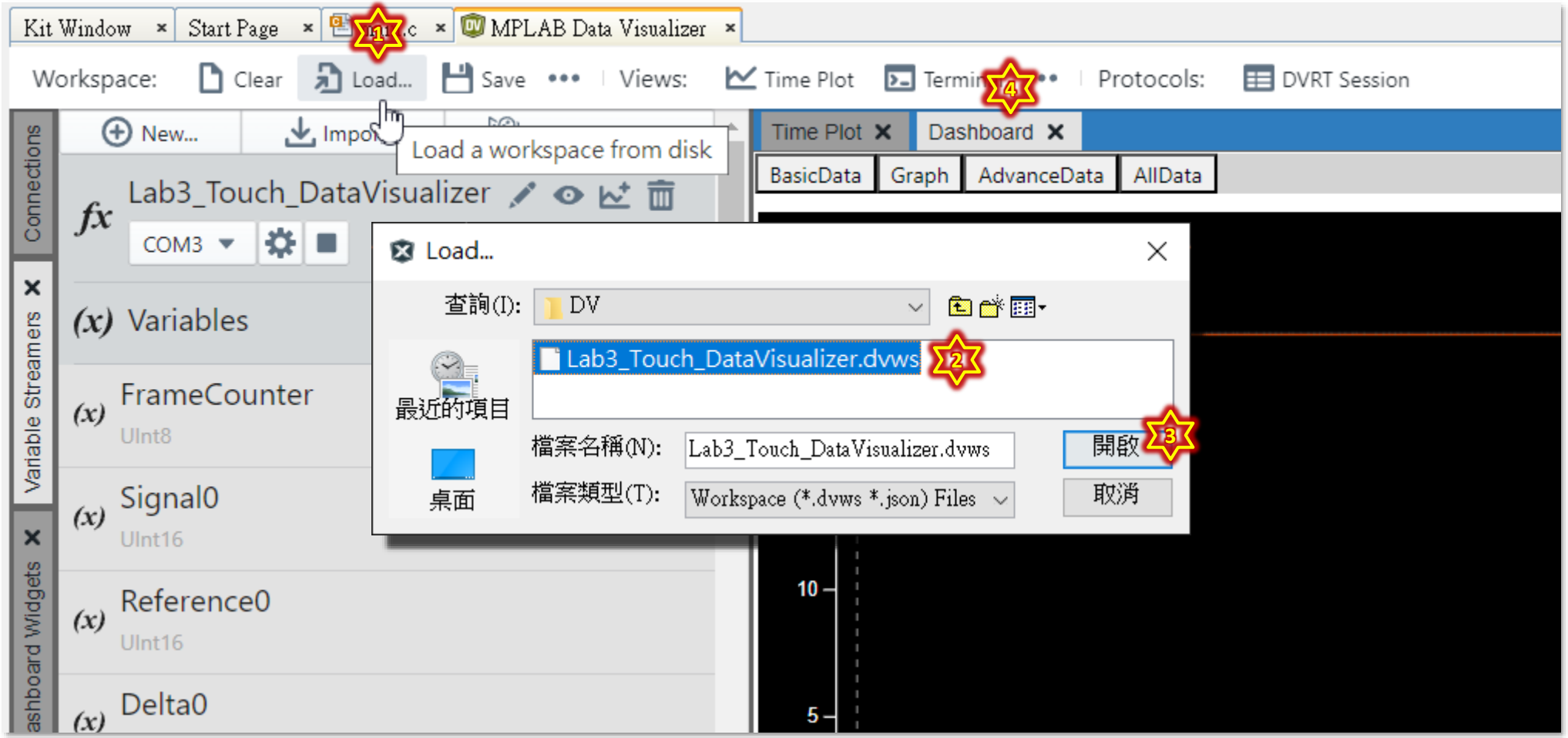
Button Data

Channel ID	Sensor Type	State
0	Slider 0[0]	0
1	Slider 0[1]	0
2	Slider 0[2]	0



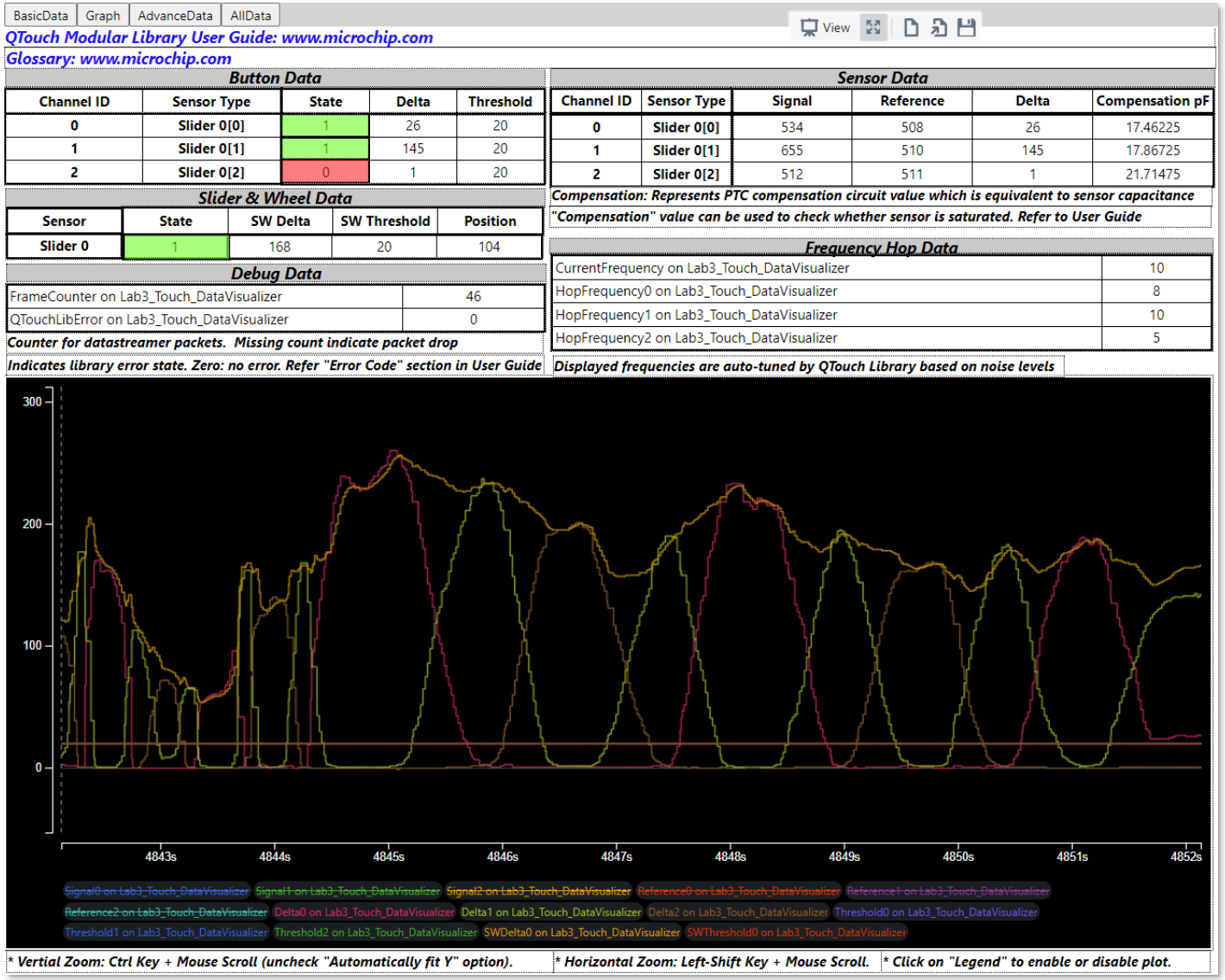
Lab3 Data Visualizer for Touch

- **Load a customized Dashboard Lab3_Touch_DataVisualizer.dvws from \DV older of project to show compact All Data.**



Lab3 Data Visualizer for Touch

- The customized **All Data** of Dashboard



Microchip Trademarks

Overview

Microchip Technology Incorporated's products are marketed and sold under one or more of the following trademarks belonging to Microchip or one of Microchip's subsidiaries. Questions regarding use of these trademarks should be addressed to the Microchip's Legal Department via email: legal.department@microchip.com.

The following are registered trademarks of Microchip in the U.S.A. and other countries:

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA

The following are registered trademarks of Microchip in the U.S.A:

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL

The following are registered trademarks of Microchip in other countries: BeaconThings, GestIC, Silicon Storage Technology

The following are trademarks of Microchip in the U.S.A. and other countries:

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA

The following is a service mark of Microchip in the U.S.A.: SQTP

The following are logos of Microchip in the U.S.A. and other countries: The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time

The following is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries: GestIC

MICROCHIP