

**SAMD21 ARM Cortex-M0+
Microcontroller & Harmony v3
SAM2002_{ADV} v1.01**



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



SMART | CONNECTED | SECURE

**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 (SAM D21 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 (SAM D21 Arm Cortex-M0+)
 - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
 - (2023~2024)
 - Peripheral Library (PLIB) on APP045v4.x/v5.x (SAM D21 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 (SAM D21 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 (SAM D21 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 (SAM D21 Arm Cortex-M0+)
 - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
 - (2023~2024)
 - System Service and Driver on APP045v4.x/v5.x (SAM D21 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 (SAM D21 Arm Cortex-M0+)
 - MPLAB Harmony Framework v3.x + XC32 Compiler + MPLAB X IDE
 - (2023~2024)
 - QTouch Library and USB Device Driver on APP045v4.x (SAM D21 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

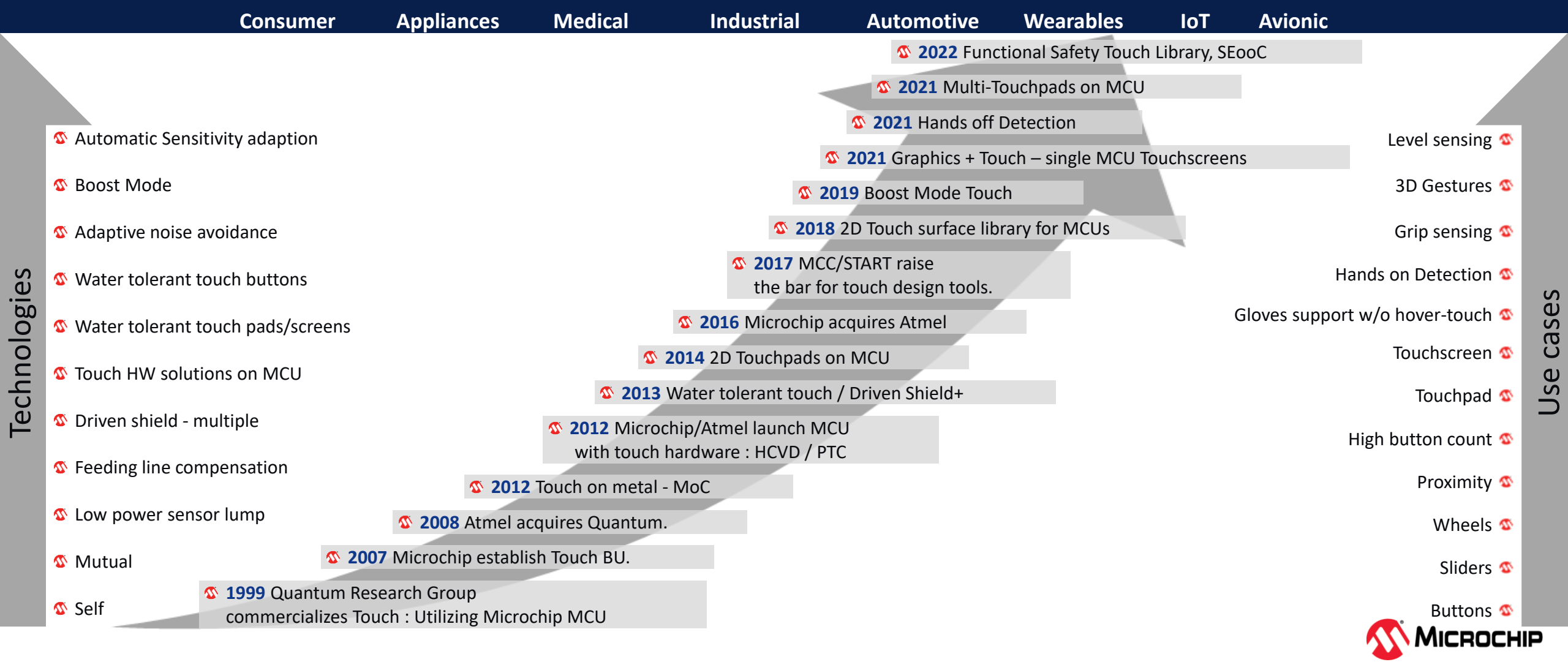
- [Introduction to QTouch® Peripheral Touch Controller \(PTC\)](#)
- [Harmony Touch Configurator](#)
- [APP045 EVB QTouch® Design and Pattern](#)
 -  [Lab1 Touch Buttons](#)
 -  [Lab2 Touch Slider](#)
 -  [Lab3 Data Visualizer for Touch](#)
- [USB Fundamentals](#)
- [USB Device Library \(USB Device Architecture\)](#)
- [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)

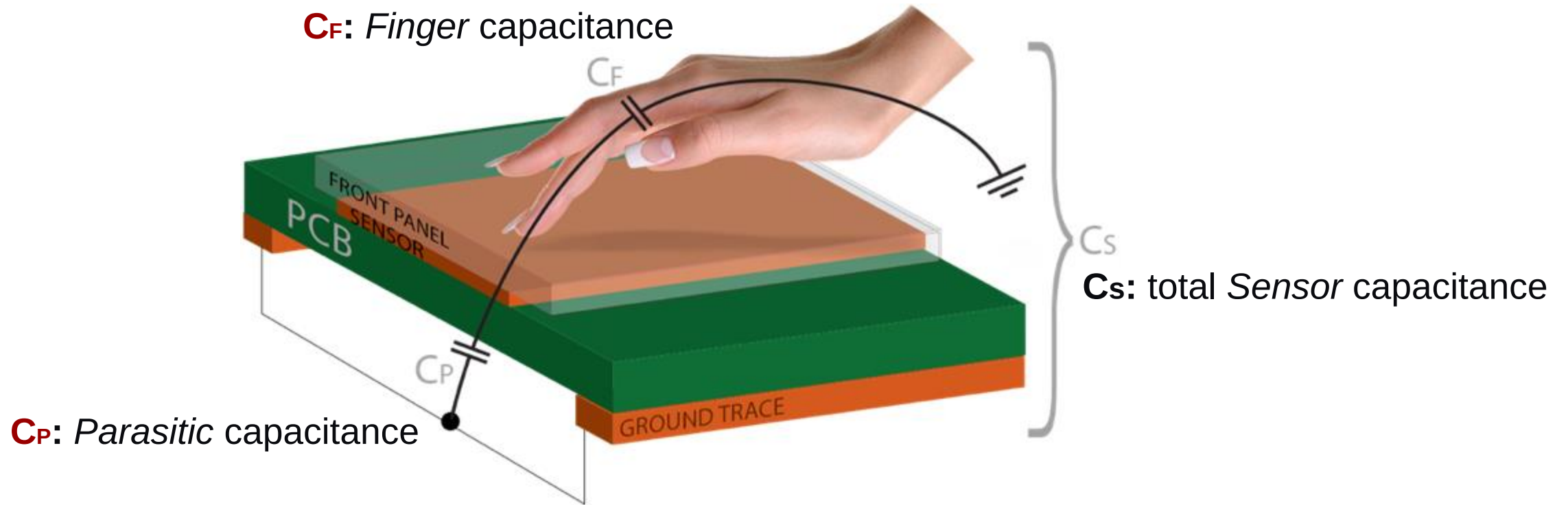
Long-Term innovative Player in Touch

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



Introduction to QTouch® Peripheral Touch Controller (PTC)

Capacitive Touch Sensor

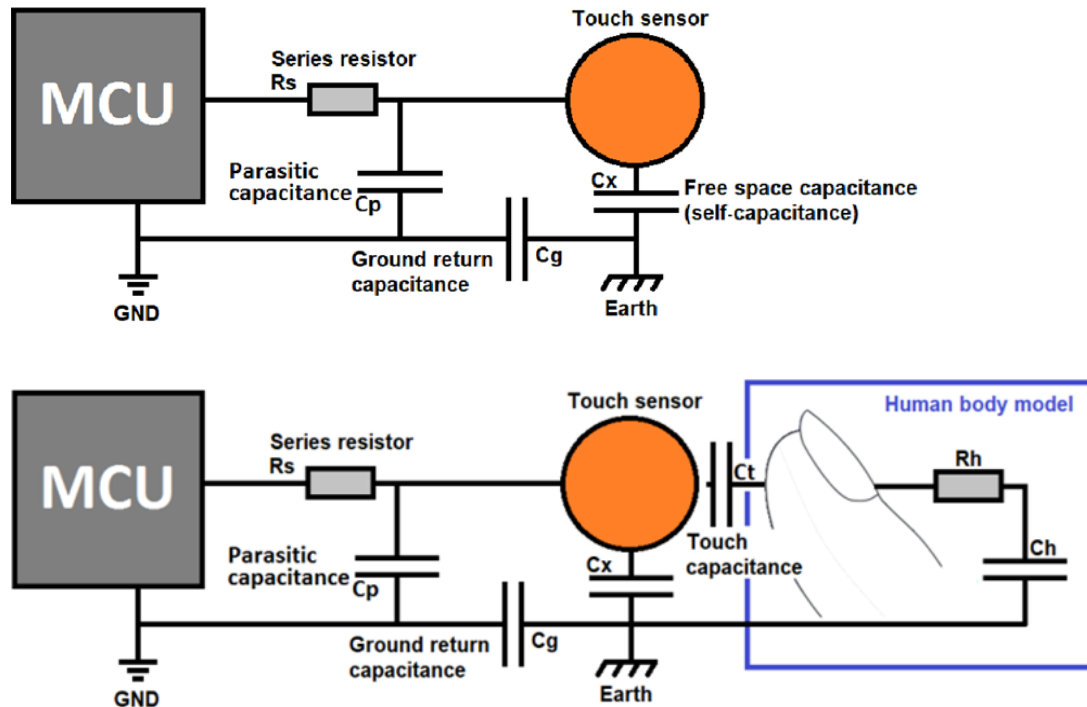


$$\text{Sensor Capacitance } (C_S) = C_P + C_F$$

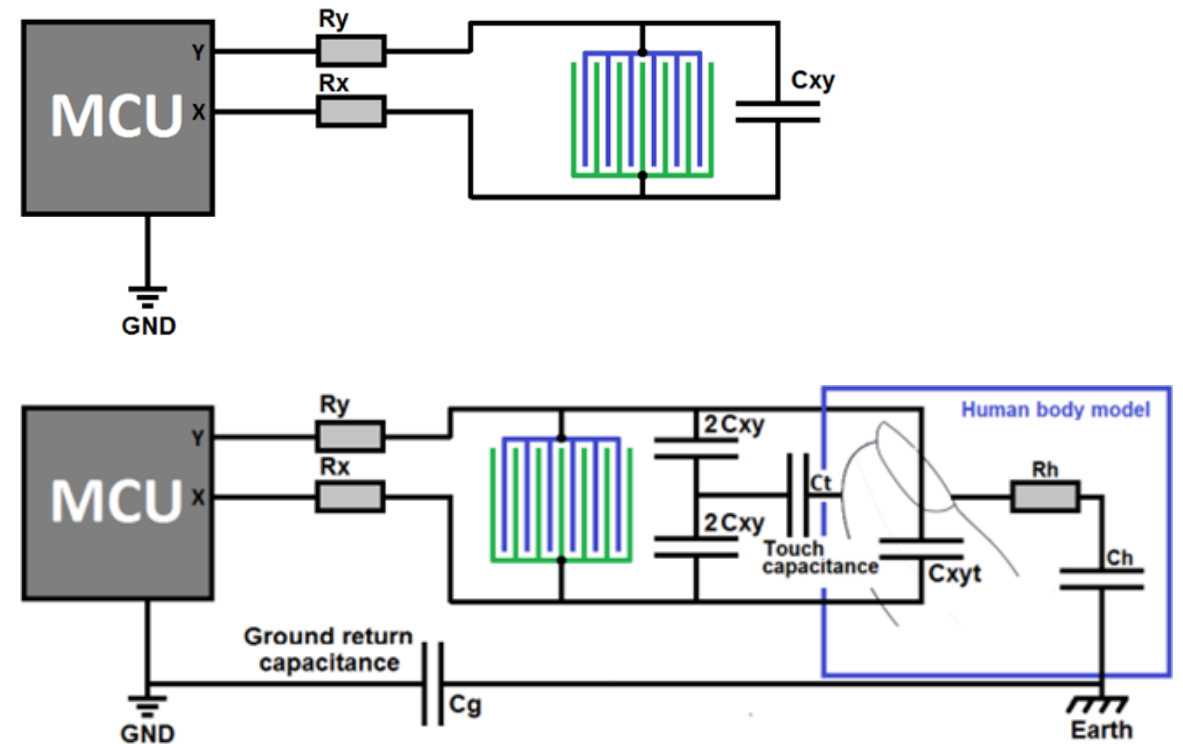
Introduction to QTouch® Peripheral Touch Controller (PTC)

Capacitance Touch Measurement

Self Capacitance Touch



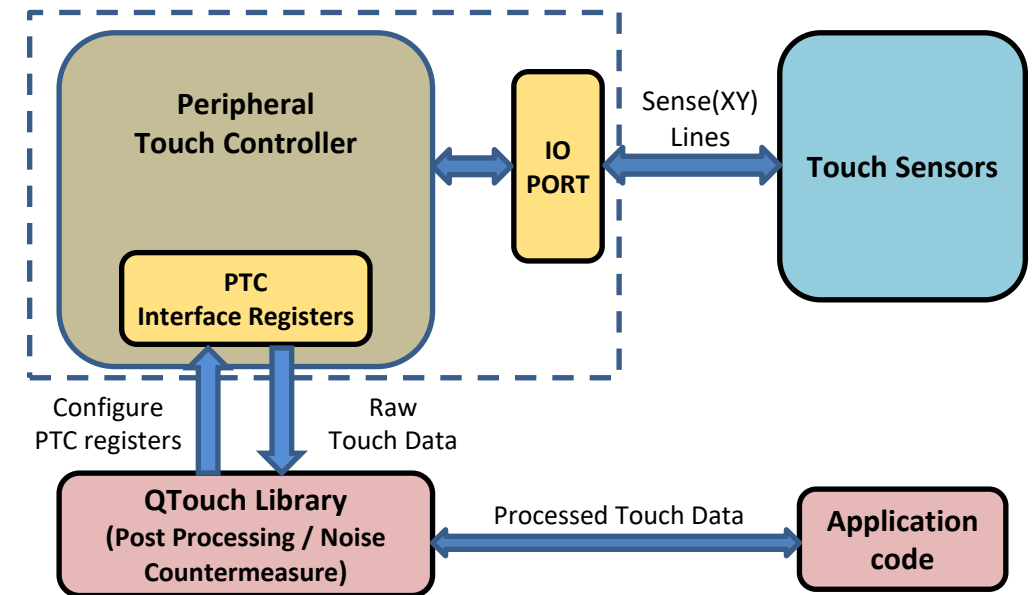
Mutual Capacitance Touch



Introduction to QTouch® Peripheral Touch Controller (PTC)

Peripheral Touch Controller (PTC)

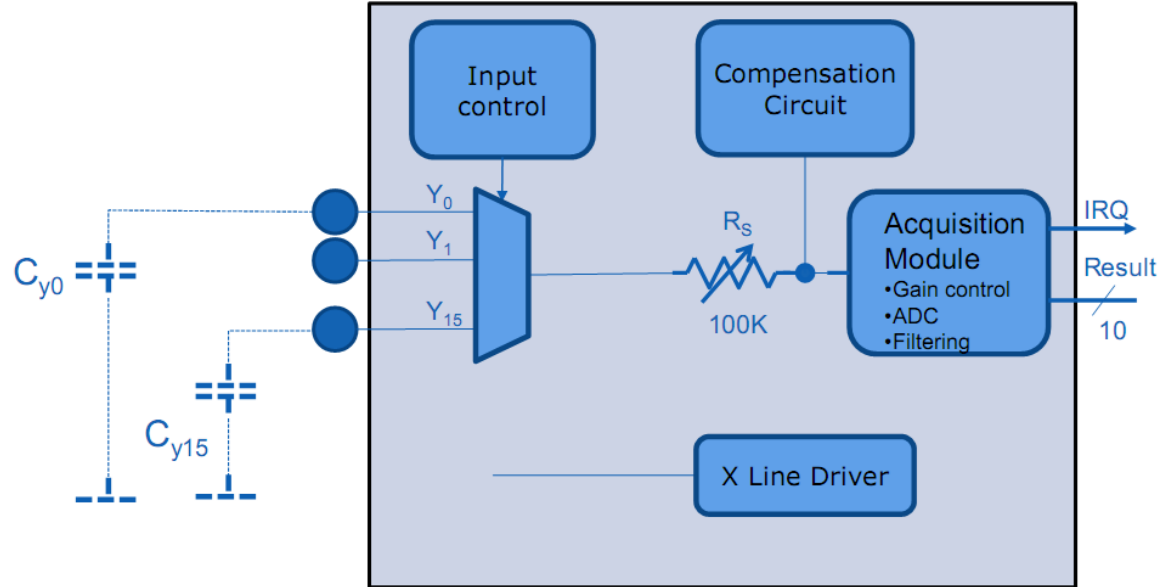
- The Peripheral Touch Controller (PTC) acquires signals in order to detect touch on capacitive sensors.
- The external capacitive touch sensor is typically formed on a PCB, and the sensor electrodes are connected to the analog front end of the PTC through the I/O pins in the device.
- The PTC supports 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.



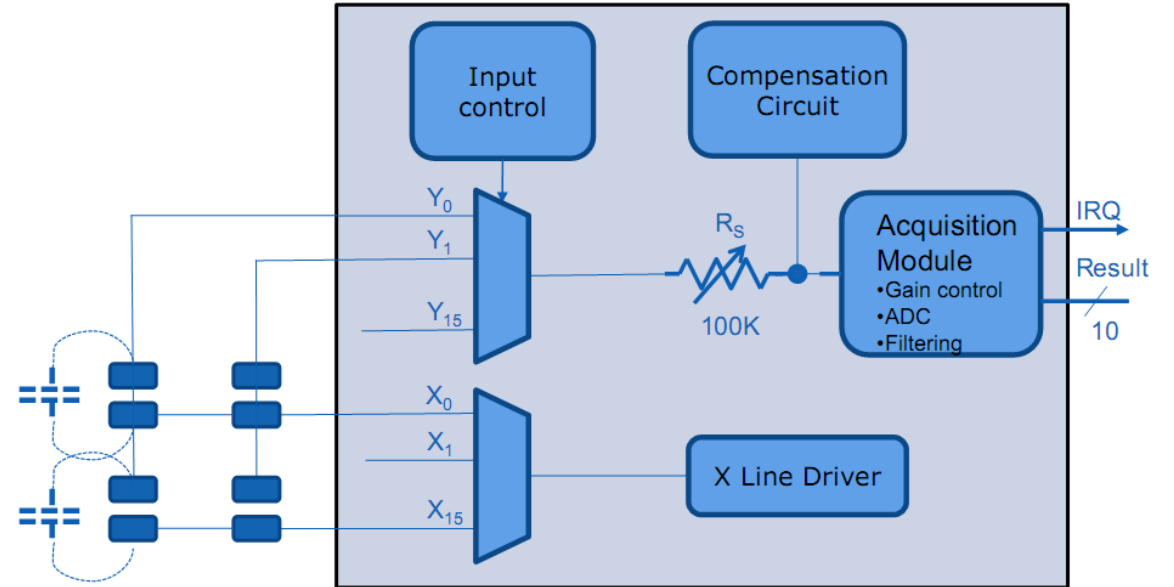
Introduction to QTouch® Peripheral Touch Controller (PTC)

PTC Block Diagram

Self Capacitance Touch



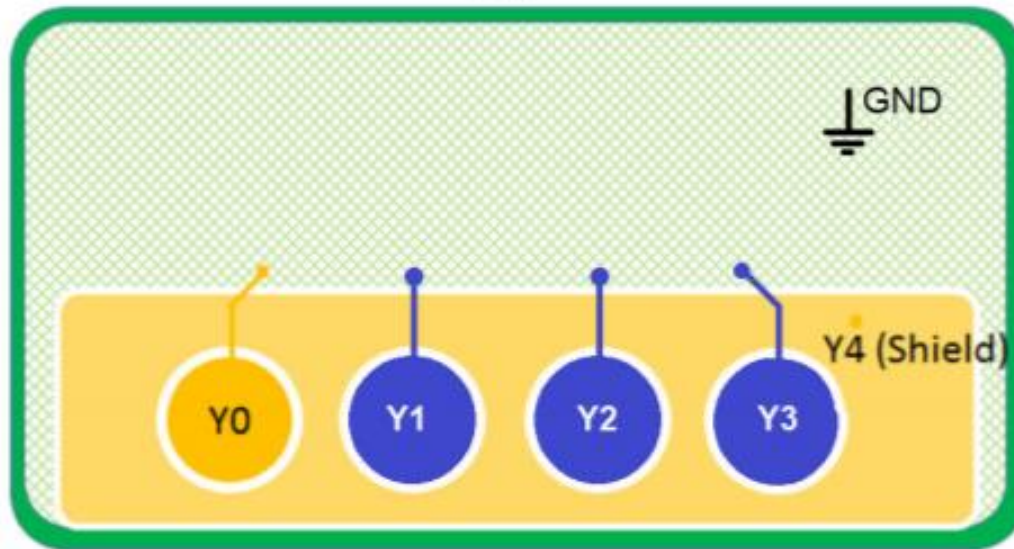
Mutual Capacitance Touch



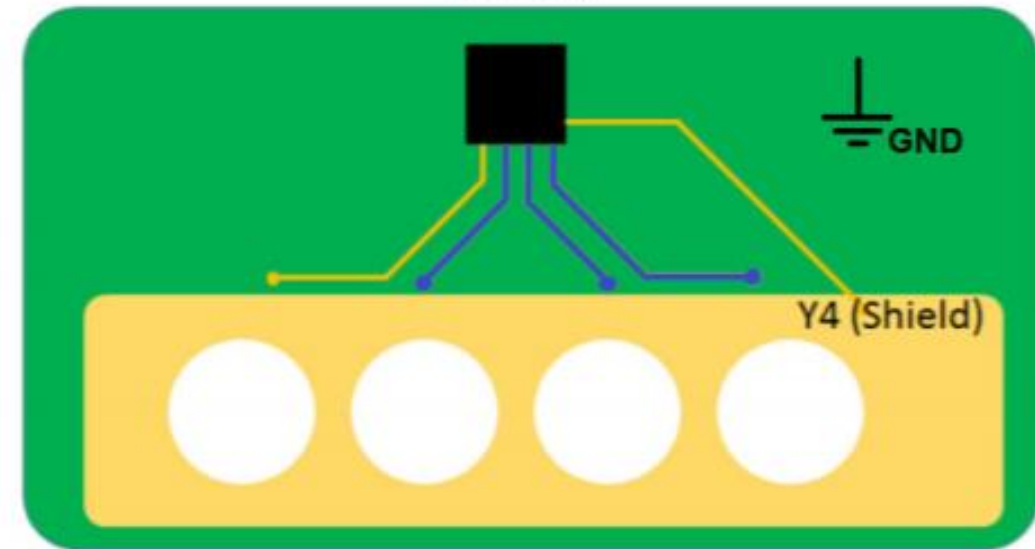
Introduction to QTouch® Peripheral Touch Controller (PTC)

Driven Shield

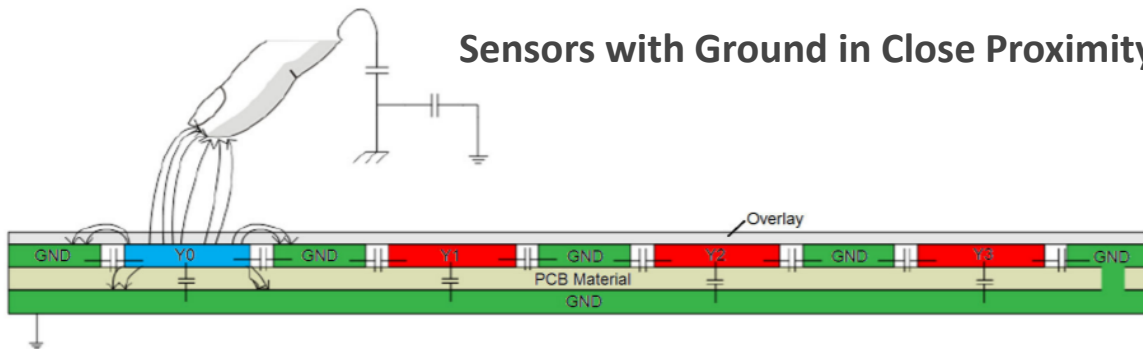
Top



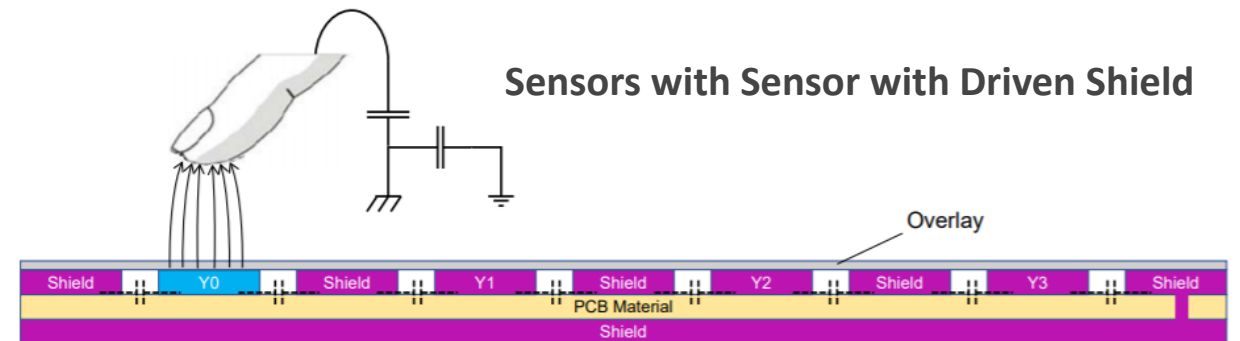
Bottom



Sensors with Ground in Close Proximity



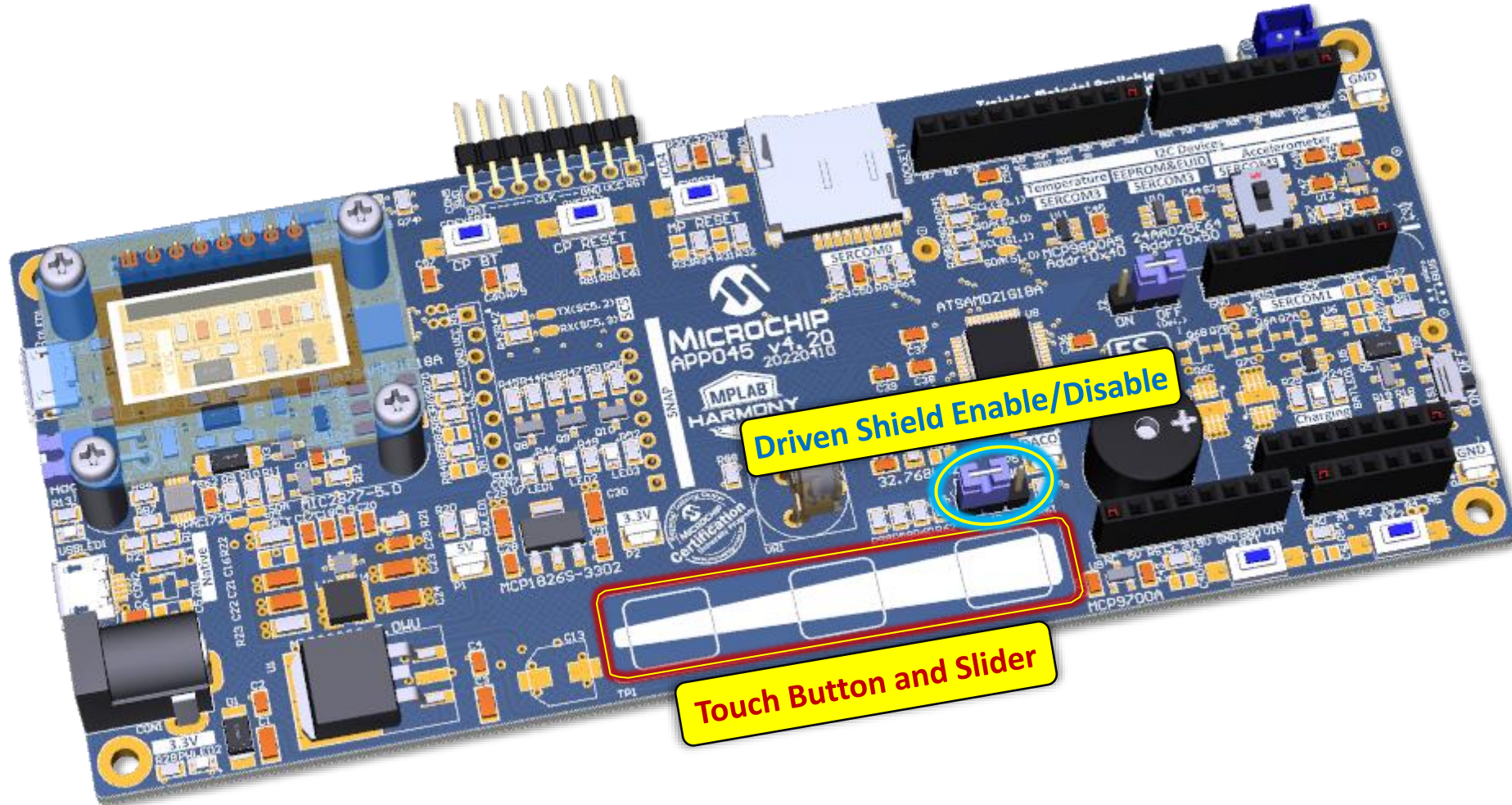
Sensors with Sensor with Driven Shield



APP045 EVB QTouch® Design and Pattern

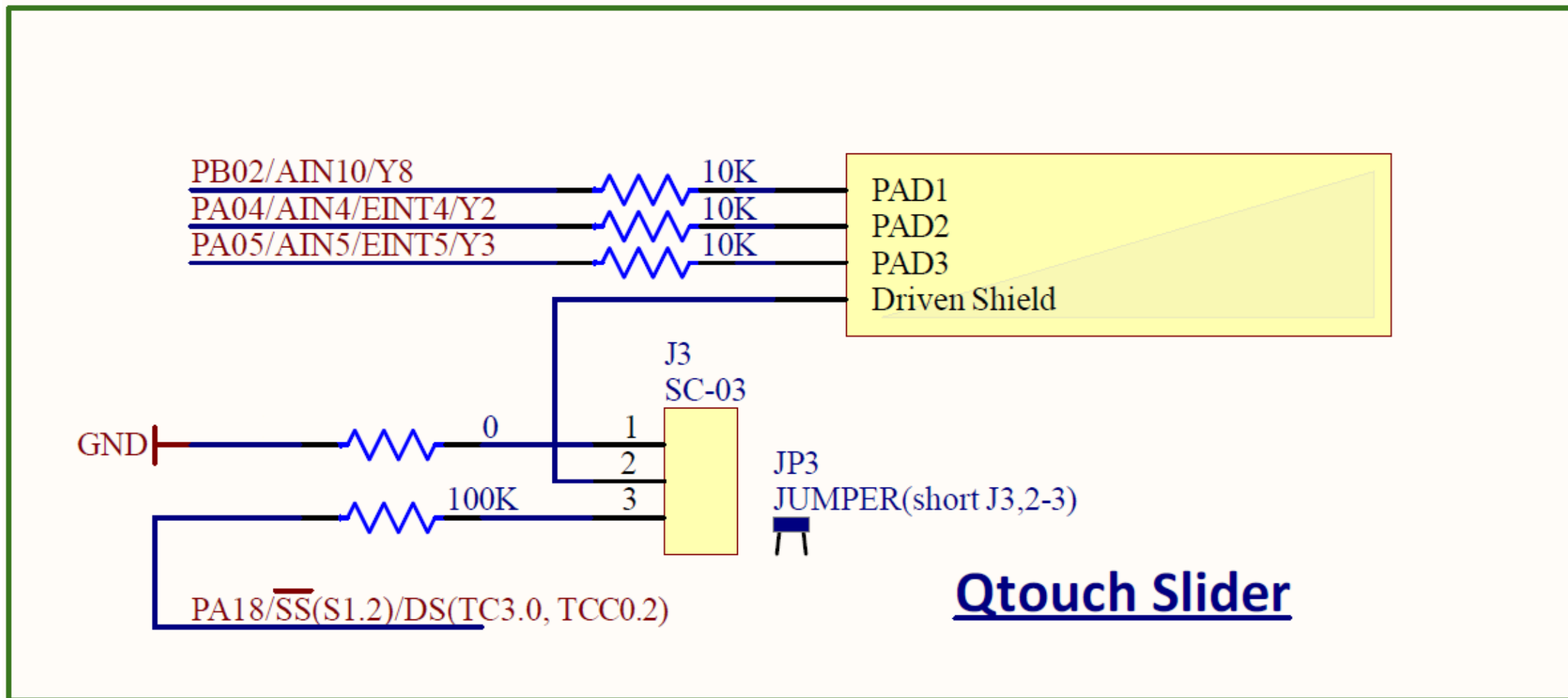
APP045 EVB QTouch® Design and Pattern

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



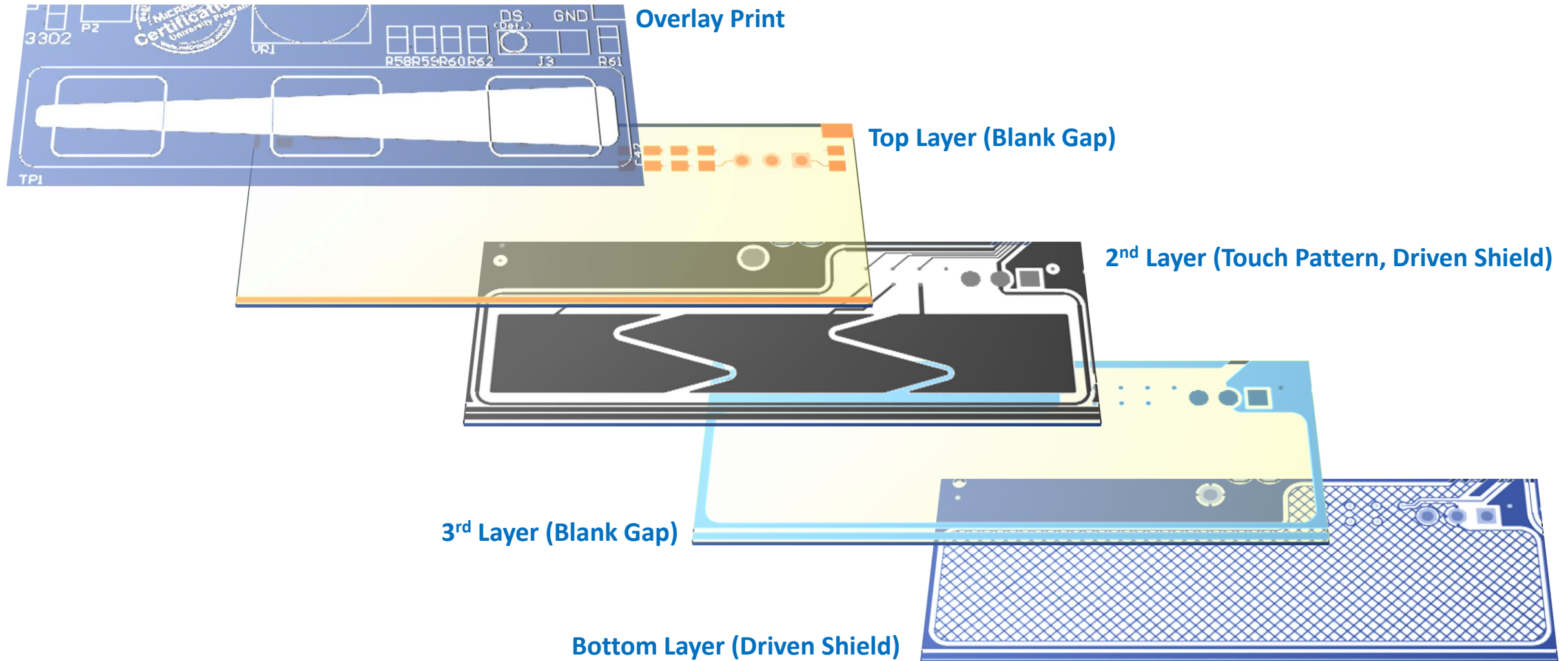
APP045 EVB QTouch® Design and Pattern

Schematic



APP045 EVB QTouch® Design and Pattern

PCB Stack



Harmony Touch Configurator

Harmony Touch Configurator

Create : Select Sensing Type and add Touch Sensors

Touch Configuration

Create Configure Tune Summary

Buttons Sliders Wheels Horizontal segments Vertical segments Gesture Type

Buttons: 1 Add

Sliders: 1 Segments: 4 Add

Wheels: 1 Segments: 4 Add

Horizontal segments: 4 Vertical segments: 4 Gesture Type: No gestures Add

Configure as Low Power Create Lump Sensor Delete Self-Cap Self-Cap Mutual-Cap

Important Notes 0

Information: The clock for CPU and peripherals are set and Sensor pins are auto assigned on creation. Go to configure > Sensor Pins to change. Ok

Touch Configuration: Create Configure Tune Summary

Buttons Sliders Wheels Horizontal segments Vertical segments Gesture Type

Buttons: 0 Sliders: 0 Segments: 0

Horizontal segments: 4 Vertical segments: 4 Gesture Type: No gestures Add

Configure as Low Power Create Lump Sensor Delete Self-Cap Self-Cap Mutual-Cap

Important Notes 0

Microchip

Harmony Touch Configurator

Configure – Sensor Pins : Sensors Pins signal configuration

Touch Configuration

Create **Configure** Tune Summary

Sensor Pins

☒ Pins view

☐ Surface view

Signals will be swaped if it's used by other sensor/segment during sel

Sensors	Signals
Button 0	Y0 (PA02)
Slider 0	Y1 (PA03)
	Y2 (PA04)
	Y3 (PA05)
	Y4 (PA06)
Wheel 0	Y5 (PA07)
	Y8 (PB02)
	Y9 (PB03)
	Y14 (PB08)

Touch Configuration

Create **Configure** Tune Summary

Sensor Pins

☐ Pins view

☒ Surface view

Note: Please click on the coordinates in any one of the corner or matrix to change the orientation of the surface.

(0,255) S5 S6 S7 S8 (255,255)

S4

S3

S2

S1

(0,0)

(255,0)

Touch Configuration

Create **Configure** Tune Summary

Sensor Pins

☒ Pins view

☐ Surface view

Drag and drop sensor to modify its lines. Change the multiplexed x line to y line and vice versa by clicking it. The recommended pin configuration for a mutual cap slider/wheel is a common Y line with multiple X lines.
B1 = Button 1, S1:2 = Slider 1: Segment 2.

	X0	X1	X2	X3	X4	X5	X6	X7	X8
Y0	B0	S0:1	S0:2	S0:3	S0:4	W0:1	W0:2	W0:3	W0:4
Y1	SUR	SUR	SUR	SUR					
Y2	SUR	SUR	SUR	SUR					
Y3	SUR	SUR	SUR	SUR					
Y4	SUR	SUR	SUR	SUR					

Sensor Pins

Sensor Parameters

Common Parameters

Gesture Parameters

Driven Shield

Boost Mode

Frequency Hop / AFA

Functional Safety

Self Test (BIST)

Host Interface

● Contact [support](#)

● Change device

● Change technology

Harmony Touch Configurator

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

The screenshot displays the 'Configure' tab of the Harmony Touch Configurator. The 'Sensor Parameters' section is active, showing configuration options for various sensors. The 'PTC Clock Frequency Info' section indicates that the GCLK / CLKPER is 4.0000 MHz, and the PTC Clock is 1.0000 MHz. The 'Sensor Parameters' table lists configurations for Button 0, Slider 0, and Wheel 0. The 'Sensor Parameters' sidebar on the right is also visible, showing the 'Sensor Parameters' tab selected.

PTC Clock Frequency Info

Red indicates invalid PTC-Clock Frequency . Blue indicates valid PTC-Clock Frequency

Prescalar: 4, 8, 16, 32

PTC Clock: 1.0000 MHz, 0.5000 MHz, 0.2500 MHz, 0.1250 MHz

GCLK / CLKPER: 4.0000 MHz

Sensors	Digital Filter Gain	Digital Filter Oversampling	Analog Gain	Series Resistor	PTC Clock (MHz)	Sensor Detect Threshold	Sensor Hysteresis	Sensor AKS
Button 0	1	16 samples	1	No resistor	1.0000	20	25 %	No AKS
Slider 0								
Scroller Resolution		8 Bit	Scroller Deadband		10 Percent	Scroller Position Hysteresis		8
Detect threshold		20						
S1	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
S2	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
S3	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
S4	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
Wheel 0								
Scroller Resolution		8 Bit	Scroller Deadband		10 Percent	Scroller Position Hysteresis		8

Sensor Parameters

- Sensor Pins
- Sensor Parameters
- Common Parameters
- Gesture Parameters
- Driven Shield
- Boost Mode
- Frequency Hop / AFA
- Functional Safety
- Self Test (BIST)
- Host Interface

● Contact [support](#)

● Change device

● Change technology

Harmony Touch Configurator

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

The screenshot displays the 'Touch Configuration' window with the 'Configure' tab selected. The 'Common Parameters' section is highlighted in the sidebar. The main area is divided into two sections: 'Acquisition' and 'Sensor'.

Acquisition Parameters:

Parameter	Value	Control
Scan Rate (ms)	20	Up/Down arrows
PTC Interrupt Priority	3	Up/Down arrows
Acquisition Frequency	FREQ_SEL_0	Dropdown arrow

Sensor Parameters:

Parameter	Value	Control
Detect Integration	4	Up/Down arrows
Away from Touch Recal Intergration Count	5	Up/Down arrows
Away from Touch Recal Threshold	100 percent of Touch threshold	Dropdown arrow
Touch Drift Rate	20	Up/Down arrows
Away from Touch Drift Rate	5	Up/Down arrows
Drift Hold Time	20	Up/Down arrows
Re-burst mode	Reburst sensors only in process of calibration / filter in / filter out and AKS groups	Dropdown arrow
Max ON Duration	0	Up/Down arrows

Sidebar Navigation:

- Sensor Pins
- Sensor Parameters
- Common Parameters**
- Gesture Parameters
- Driven Shield
- Boost Mode
- Frequency Hop / AFA
- Functional Safety
- Self Test (BIST)
- Host Interface

Footer:

- Contact [support](#)
- Change device
- Change technology

Harmony Touch Configurator

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

The screenshot displays the 'Configure' tab of the Harmony Touch Configurator. The 'Gesture Parameters' section is active, as indicated by the red box around the 'Configure' tab and the yellow box around the 'Gesture Parameters' item in the right-hand sidebar. The 'Enable Gestures - 1 Finger' toggle is turned on, also highlighted with a yellow box. Below this, a note states: 'Note: Enabling only 1 Finger Gestures Support. If 1&2 Finger Gestures support is needed, delete the existing Surface sensor and add a new Surface sensor with 1&2 Finger Gestures option type.'

The 'Gesture - Tap' section contains the following settings:

Parameter	Value
Tap Release timeout	20
Tap Hold timeout	100
Touch Drift Rate	20
Seq Tap distance threshold	50

The 'Gesture - Swipe' section contains the following settings:

Parameter	Value
Edge Boundary	0
Swipe Timeout	70
Horizontal Swipe distance threshold	30
Vertical Swipe distance threshold	40

The 'Gesture - Wheel' section contains the following settings:

Parameter	Value
Wheel Post-scaler	1

The right-hand sidebar lists various configuration categories: Sensor Pins, Sensor Parameters, Common Parameters, Gesture Parameters (highlighted), Driven Shield, Boost Mode, Frequency Hop / AFA, Functional Safety, Self Test (BIST), and Host Interface. At the bottom right, there are three icons with labels: a yellow circle for 'Contact support', a grey circle for 'Change device', and a yellow circle for 'Change technology'.

Harmony Touch Configurator

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

Touch Configuration

Create **Configure** Tune Summary

Driven Shield

To improve noise performance and water tolerance, driven shield can be enabled on self-capacitance sensors. Driven shield can be enabled either on adjacent sensors only or additionally using a dedicated shield sensor. More information can be found [Microchip Developer Help page](#). In this device, PWM signal can be used as Driven Shield. Enabling driven shield will use timer (TC/TCC), DMA and Event system.

Apply

Enable Dedicated Shield Pin ☒

Any Timer WaveOutput (WO) pin can be used as Driven Shield Pin. Select the timer and pin for dedicated driven shield.

Select Dedicated Timer TC3

Select Timer Pin PA18E_TC3_WO0

"Driven shield plus" feature configures the PTC pins which are currently not being measured as shield pins. Enabling this feature shows the timer multiplexing option available for each Y lines used in this application. Selecting the timer will add the timer to the project and will be used for touch. It is possible to select timer only for few channels.

Enable Driven Shield Plus ☒

Sensors	Timers
Button 0	---

Slider 0	---

Search

- Sensor Pins
- Sensor Parameters
- Common Parameters
- Gesture Parameters
- Driven Shield**
- Boost Mode
- Frequency Hop / AFA
- Functional Safety
- Self Test (BIST)
- Host Interface

● Contact [support](#)

● Change device

● Change technology

Harmony Touch Configurator

Configure – Frequency Hop / AFA

Touch Configuration

Create **Configure** Tune Summary

Frequency Hop / AFA

Frequency Hop is a mechanism used in touch measurement to avoid noisy signal value. In Frequency Hop, more than one bursting frequency (user configurable) is used. Refer [Touch Modular Library Userguide](#) for more details on Frequency Hop.

Enable Frequency Hop / AFA

Enable Frequency Auto Tuning

AutoTune Parameters

Maximum Variance

25

Maximum Tune-in count

6

Channel Configuration

Frequency Hop Steps

3

Hop Frequency 0

Frequency 0

Hop Frequency 1

Frequency 1

Hop Frequency 2

Frequency 2

Frequency Hop / AFA

Functional Safety

Self Test (BIST)

Host Interface

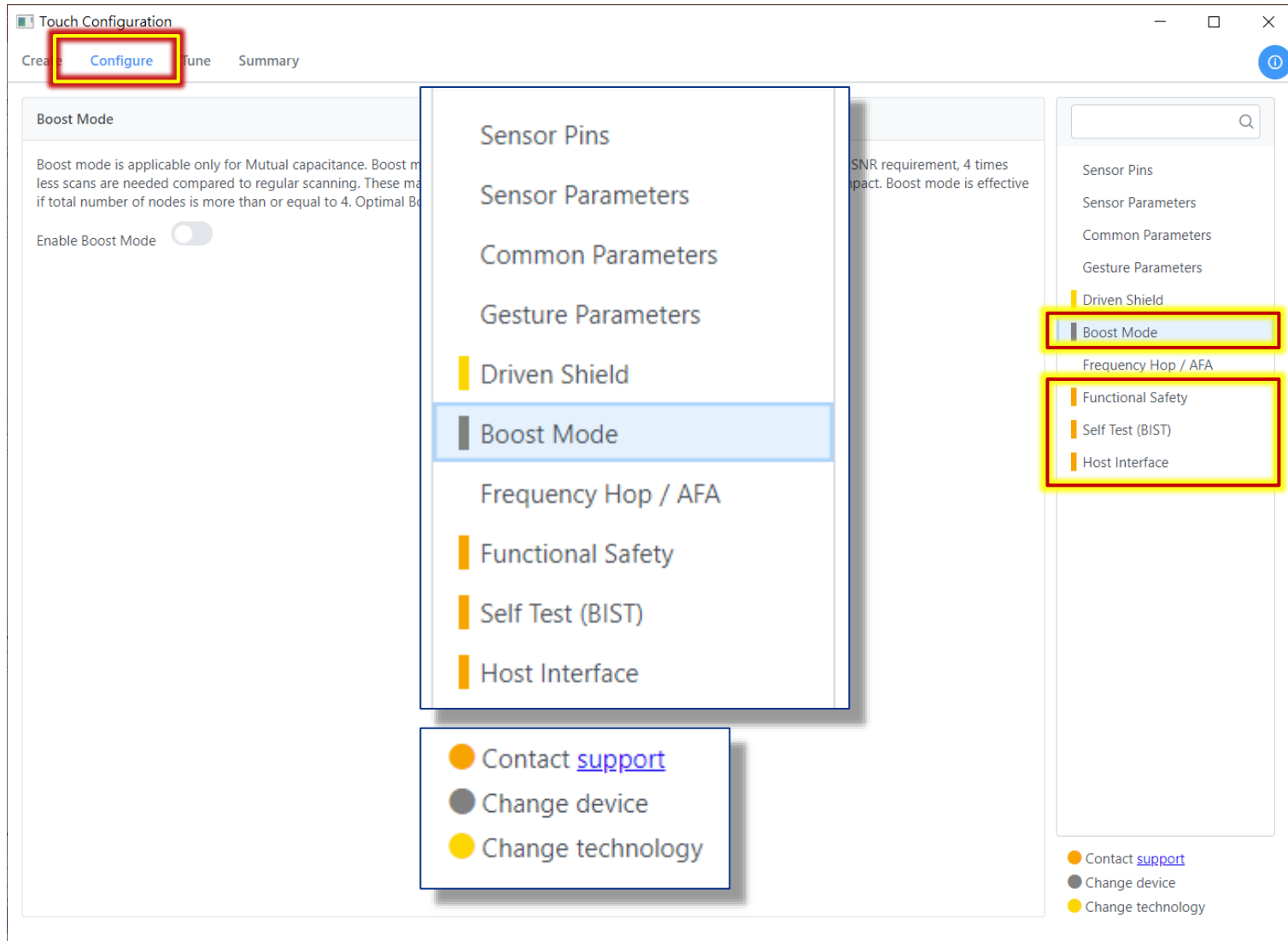
Contact [support](#)

Change device

Change technology

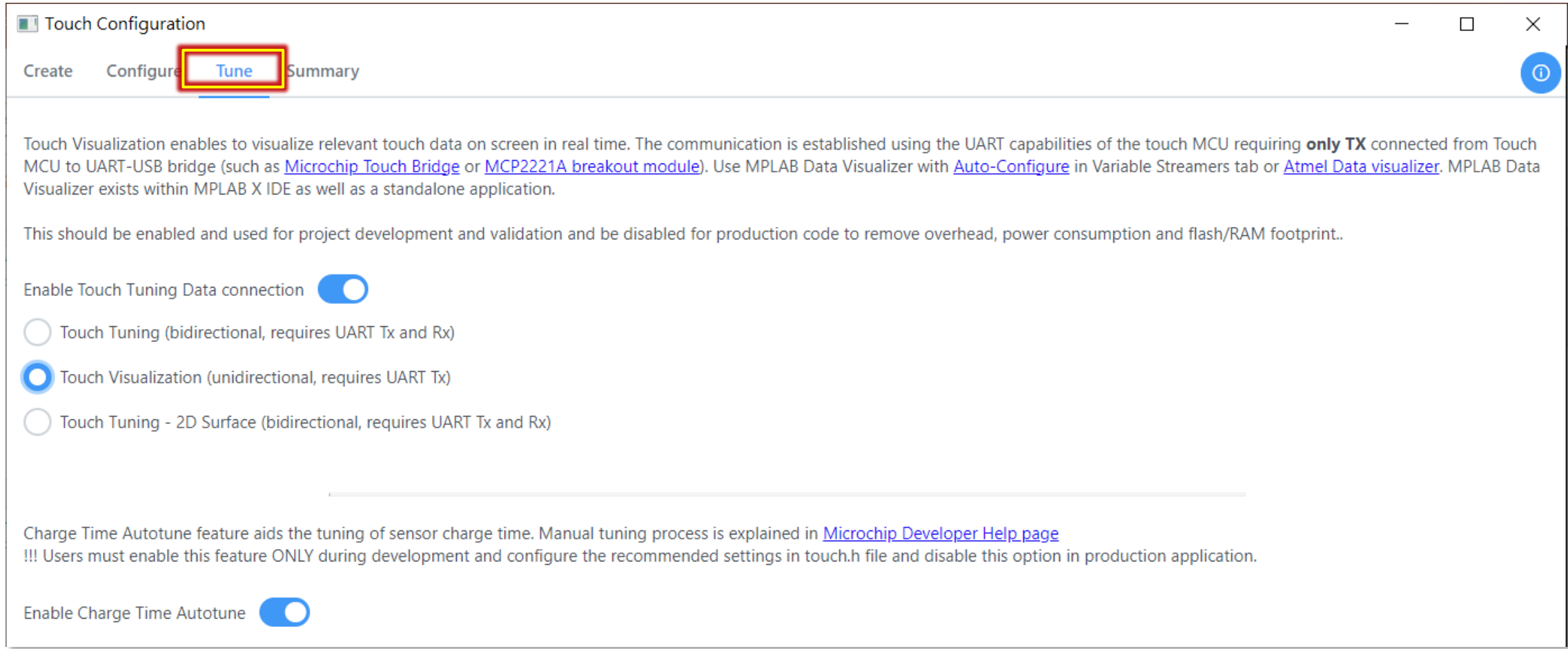
Harmony Touch Configurator

Configure – Others



Harmony Touch Configurator

Tune – Enable the debug output for Touch Tuning



The screenshot shows the 'Touch Configuration' window with the 'Tune' tab selected. The 'Tune' tab is highlighted with a red and yellow border. The window contains the following content:

Touch Visualization enables to visualize relevant touch data on screen in real time. The communication is established using the UART capabilities of the touch MCU requiring **only TX** connected from Touch MCU to UART-USB bridge (such as [Microchip Touch Bridge](#) or [MCP2221A breakout module](#)). Use MPLAB Data Visualizer with [Auto-Configure](#) in Variable Streamers tab or [Atmel Data visualizer](#). MPLAB Data Visualizer exists within MPLAB X IDE as well as a standalone application.

This should be enabled and used for project development and validation and be disabled for production code to remove overhead, power consumption and flash/RAM footprint..

Enable Touch Tuning Data connection ☒

☐ Touch Tuning (bidirectional, requires UART Tx and Rx)

☒ Touch Visualization (unidirectional, requires UART Tx)

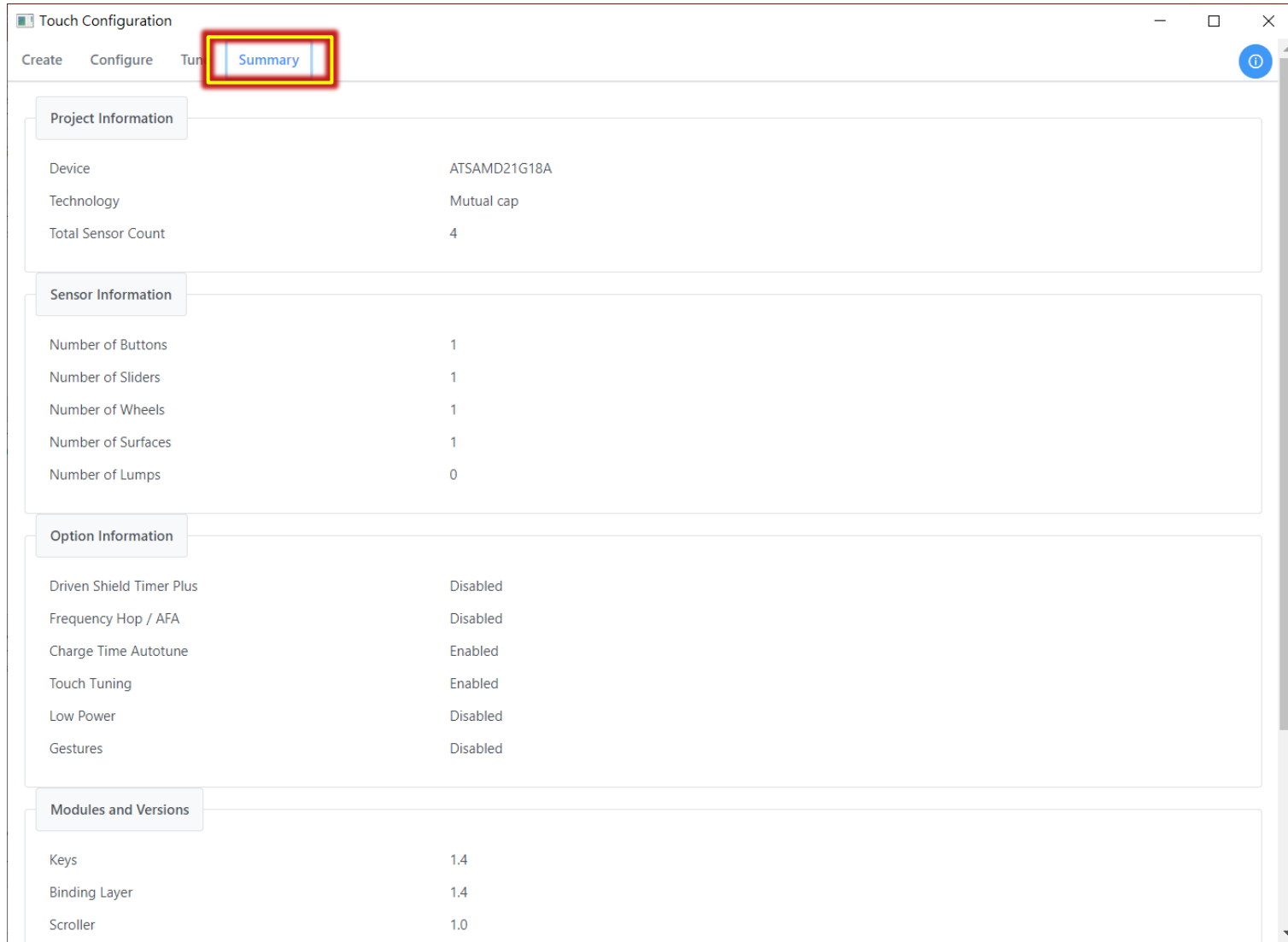
☐ Touch Tuning - 2D Surface (bidirectional, requires UART Tx and Rx)

Charge Time Autotune feature aids the tuning of sensor charge time. Manual tuning process is explained in [Microchip Developer Help page](#)
!!! Users must enable this feature ONLY during development and configure the recommended settings in touch.h file and disable this option in production application.

Enable Charge Time Autotune ☒

Harmony Touch Configurator

Summary – Configuration summary



The screenshot shows the 'Summary' tab of the Harmony Touch Configurator. The 'Summary' tab is highlighted with a red and yellow border. The interface displays four sections: Project Information, Sensor Information, Option Information, and Modules and Versions. Each section contains a table of configuration details.

Project Information

Device	ATSAMD21G18A
Technology	Mutual cap
Total Sensor Count	4

Sensor Information

Number of Buttons	1
Number of Sliders	1
Number of Wheels	1
Number of Surfaces	1
Number of Lumps	0

Option Information

Driven Shield Timer Plus	Disabled
Frequency Hop / AFA	Disabled
Charge Time Autotune	Enabled
Touch Tuning	Enabled
Low Power	Disabled
Gestures	Disabled

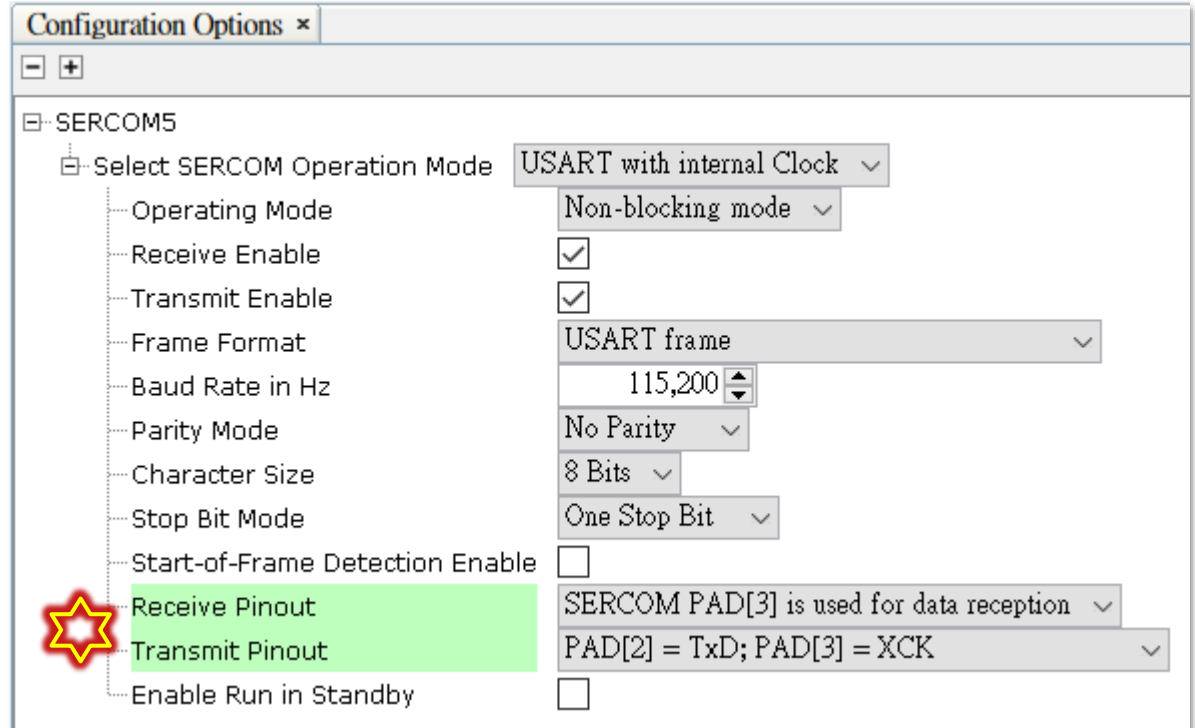
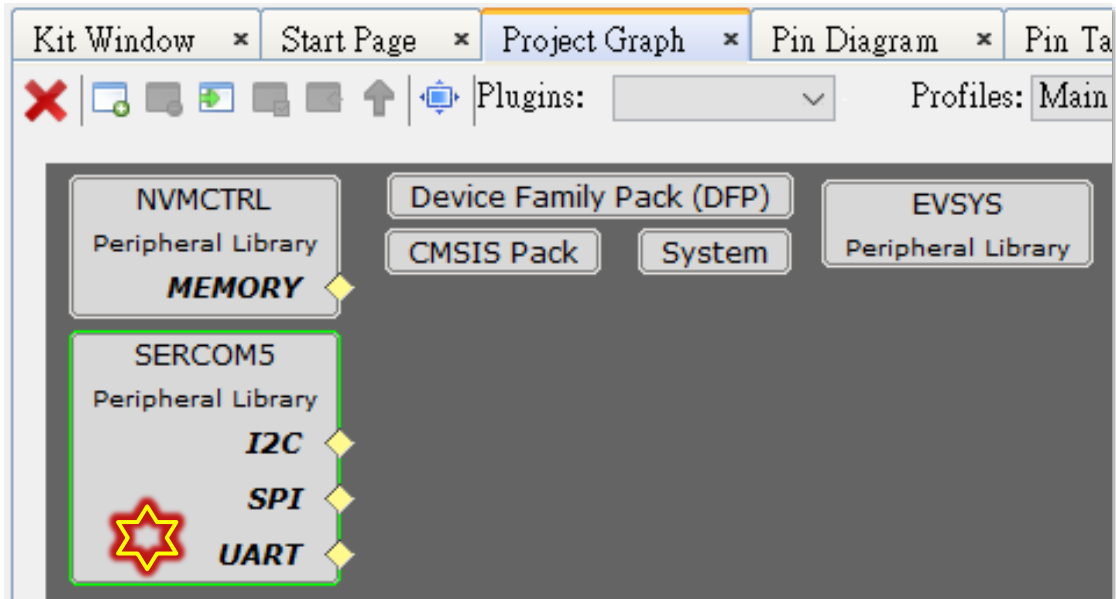
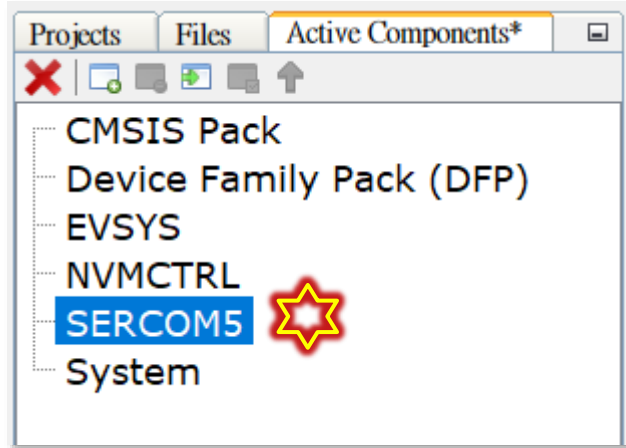
Modules and Versions

Keys	1.4
Binding Layer	1.4
Scroller	1.0

Lab1 Touch Buttons

✖ Lab1 Touch Buttons

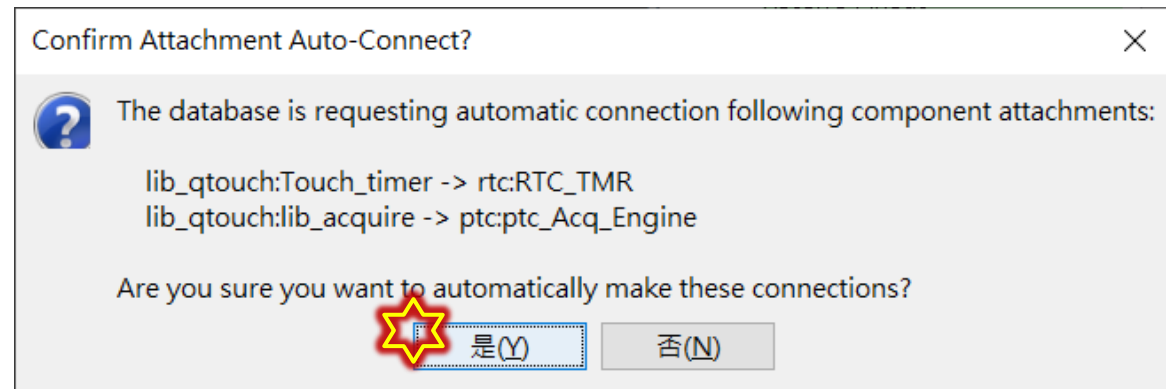
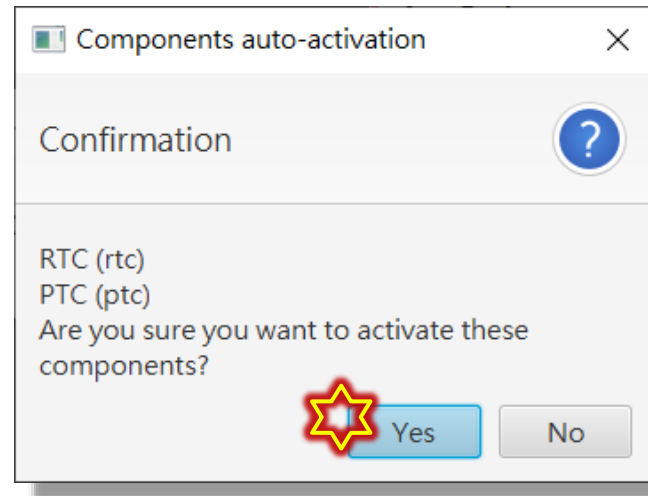
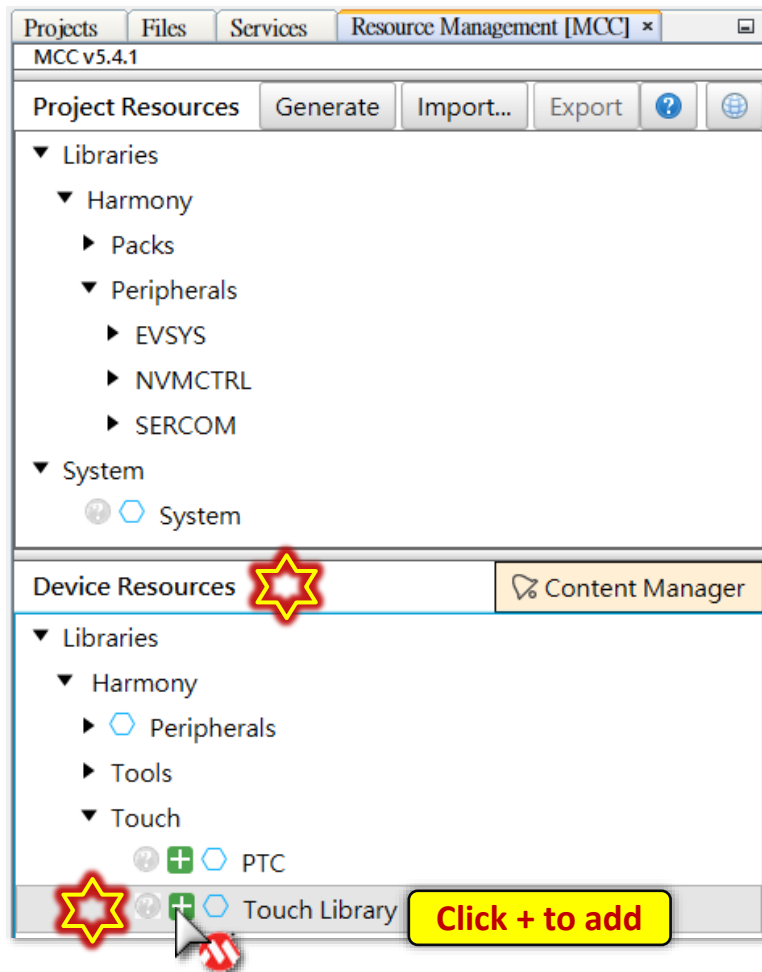
- Create a new project and add SERCOM5 in MHC for UART debug message.



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

⚙️ 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**.



⚙️ Lab1 Touch Buttons

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

MPLAB X IDE v6.15 - Lab1_QTouch_Buttons : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

default PC: 0x0 How do I? Keyword(s)

Projects Files Services Resource Management [MCC] x

MCC v5.4.1

Project Resources Generate Import... Export ?

Libraries

- Harmony
 - Packs
- Peripherals
 - EVSYS
 - NVMCTRL
 - SERCOM
 - SERCOM5
- System
 - System

Device Resources

- DAC
- DSU
- EIC
- PM
- RAM
- RTC

Content Manager

Click Content Manager

MCC Content Manager x Kit Window x Start Page x Project Graph x Pin Diagram x Pin Table x Pin Settings x

Select Export/Import ... Submit Load Manifest Select Latest Version(s) Apply Cancel ?

Click Apply to download

Content Libraries MCC Core Versions

Type to Search Globally... Input "Touch" to search

touch

Component	Version	Status	Update progress	Description	License Agreement
Harmony 3 - Capacitive Touch solutions					
touch	v3.15.0				MPLAB Harmony license
touch_apps	v3.15.0 - [remote]				MPLAB Harmony license
touch_host_driver	v3.14.0 - [remote]				MPLAB Harmony license
	v3.13.1 - [remote]				MPLAB Harmony license
	v3.13.0 - [remote]				
	v3.12.1 - [remote]				

Select latest version

⚙️ 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**.

The screenshot displays the Microchip Studio IDE interface. The **Project Graph** window shows the project components, including the **Touch Library** (Peripheral Touch Controller (PTC)) and **Acq_Engine**. The **Configuration Options** window shows the **RTC** settings, including **Hardware Settings**, **RTC Operation Mode**, and **RTC MODE 0 Configuration**. The **RTC** settings are highlighted with a red starburst icon. The **Configuration Options** window also shows the **Touch Library** settings, which are also highlighted with a red starburst icon. The **PTC** settings are also highlighted with a red starburst icon. The **Configuration Options** window shows the **RTC** settings, including **Hardware Settings**, **RTC Operation Mode**, and **RTC MODE 0 Configuration**. The **RTC** settings are highlighted with a red starburst icon. The **Configuration Options** window also shows the **Touch Library** settings, which are also highlighted with a red starburst icon. The **PTC** settings are also highlighted with a red starburst icon. The **Configuration Options** window shows the **RTC** settings, including **Hardware Settings**, **RTC Operation Mode**, and **RTC MODE 0 Configuration**. The **RTC** settings are highlighted with a red starburst icon. The **Configuration Options** window also shows the **Touch Library** settings, which are also highlighted with a red starburst icon. The **PTC** settings are also highlighted with a red starburst icon.

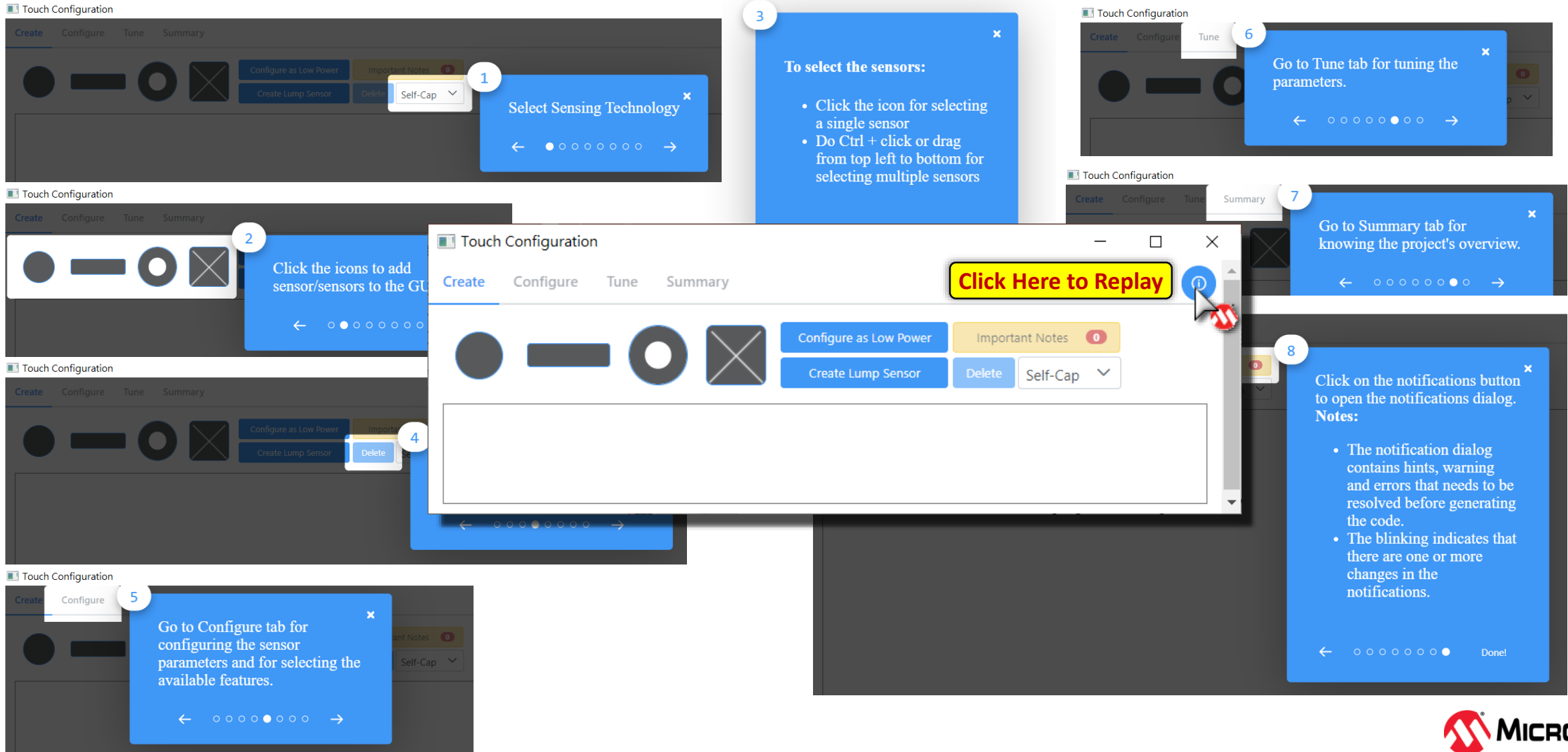
⚙️ Lab1 Touch Buttons

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

The screenshot displays the Microchip MCC v5.4.1 software interface. On the left, the 'Project Resources' pane shows a tree view with 'Libraries' expanded, including 'Harmony', 'Peripherals', 'RTC', 'SERCOM', 'Touch', and 'System'. The 'Touch' category is selected, showing 'PTC' and 'Touch Library'. On the right, the 'Project Graph' pane shows a block diagram of the system components. A yellow star icon is placed over the 'Plugins' dropdown menu, which is open, showing a list of configuration options: 'Clock Configuration', 'Event Configurator', 'NVIC Configuration', 'Pin Configuration', 'DMA Configuration', and 'Touch Configuration'. A yellow callout box with the text 'Click Touch Configuration' points to the 'Touch Configuration' option. The 'Project Graph' also shows various peripheral blocks like 'NVMCTRL', 'SERCOM5', 'RTC', 'PTC', and 'Touch Library', with connections between them.

Lab1 Touch Buttons

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



1 Select Sensing Technology

2 Click the icons to add sensor/sensors to the GUI

3 To select the sensors:

- Click the icon for selecting a single sensor
- Do Ctrl + click or drag from top left to bottom for selecting multiple sensors

4

5 Go to Configure tab for configuring the sensor parameters and for selecting the available features.

6 Go to Tune tab for tuning the parameters.

7 Go to Summary tab for knowing the project's overview.

8 Click on the notifications button to open the notifications dialog.

Notes:

- The notification dialog contains hints, warning and errors that needs to be resolved before generating the code.
- The blinking indicates that there are one or more changes in the notifications.

Click Here to Replay

⚙️ 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”.

The image illustrates the process of creating a button sensor in the Touch Configuration software. It consists of three sequential screenshots:

- Step 1:** The 'Create' tab is selected. The 'Buttons' icon is highlighted. The number '3' is entered in the 'Buttons' field. The 'Add' button is highlighted. A yellow callout box labeled 'Click' points to the 'Buttons' icon. Another yellow callout box labeled 'Self-Cap default' points to the 'Self-Cap' dropdown menu.
- Step 2:** An 'Information' dialog box is displayed. The message states: 'The clock for CPU and peripherals are set and Sensor pins are auto assigned on creation. Go to configure > Sensor Pins to change.' The 'Ok' button is highlighted.
- Step 3:** The 'Touch Configuration' window is shown with the 'Buttons' icon selected. Below the icons, three circular buttons labeled '0', '1', and '2' are visible.

✂ 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)

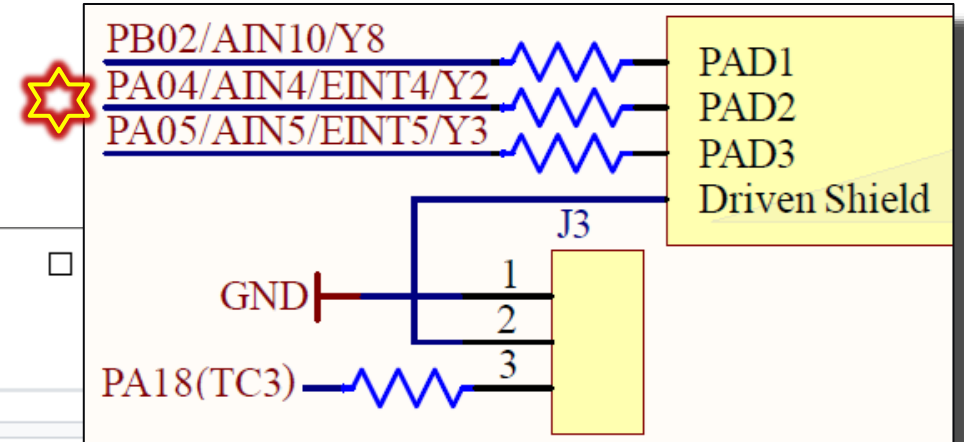
Touch Configuration

Create **Configure** Tune Summary

Sensor Pins

Signals will be swapped if it's used by other sensor/segment during selection

Sensors	Signals
Button 0	Y8 (PB02)
Button 1	Y2 (PA04)
Button 2	Y3 (PA05)



✖ 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**

Touch Configuration

Create **Configure** Tune Summary

Sensor Parameters

PTC Clock Frequency Info

Red indicates invalid PTC-Clock Frequency . Blue indicates valid PTC-Clock Frequency

Prescalar

PTC Clock

GCLK / CLKPER
4.0000 MHz

4
8
16
32

1.0000 MHz
0.5000 MHz
0.2500 MHz
0.1250 MHz

Sensors	Digital Filter Gain	Digital Filter Oversampling	Analog Gain	Series Resistor	PTC Clock (MHz)	Sensor Detect Threshold	Sensor Hysteresis	Sensor AKS
Button 0	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
Button 1	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 1
Button 2	1	16 samples	1	No resistor	1.0000	20	25 %	AKS Group 2

Sensor Pins

Sensor Parameters

Common Parameters

Driven Shield

Boost Mode

Frequency Hop / AFA

Functional Safety

Self Test (BIST)

Host Interface

AKS Group 1

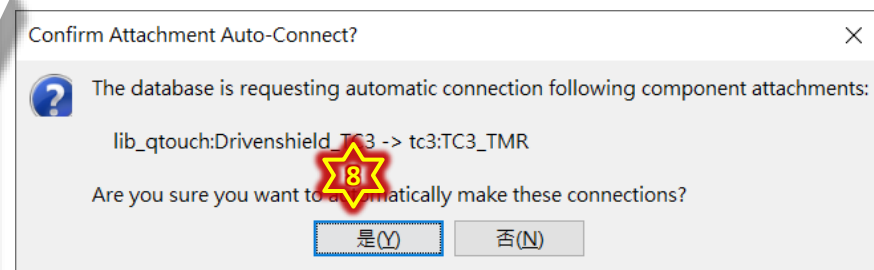
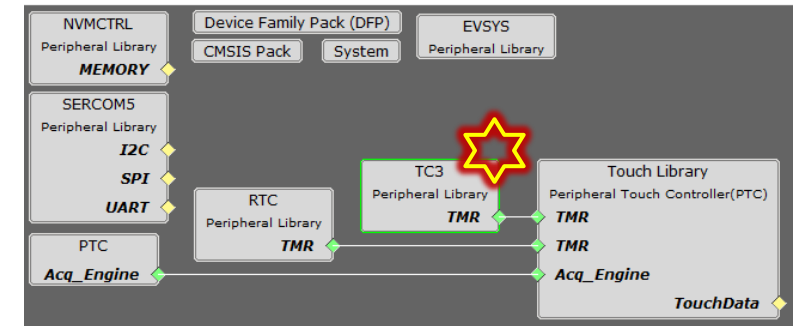
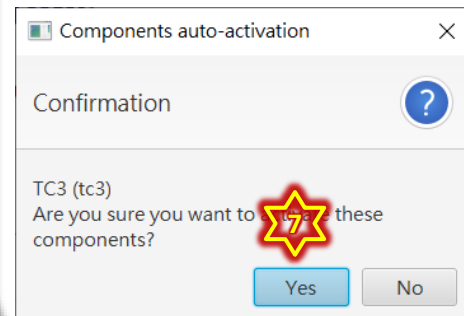
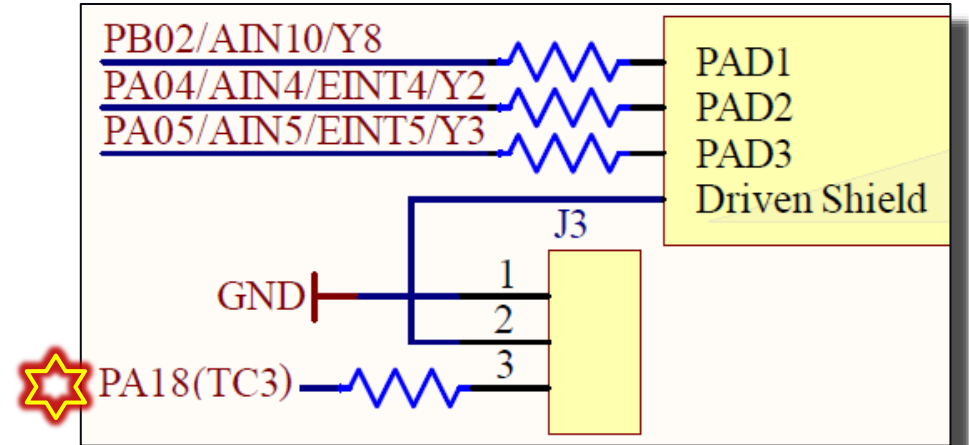
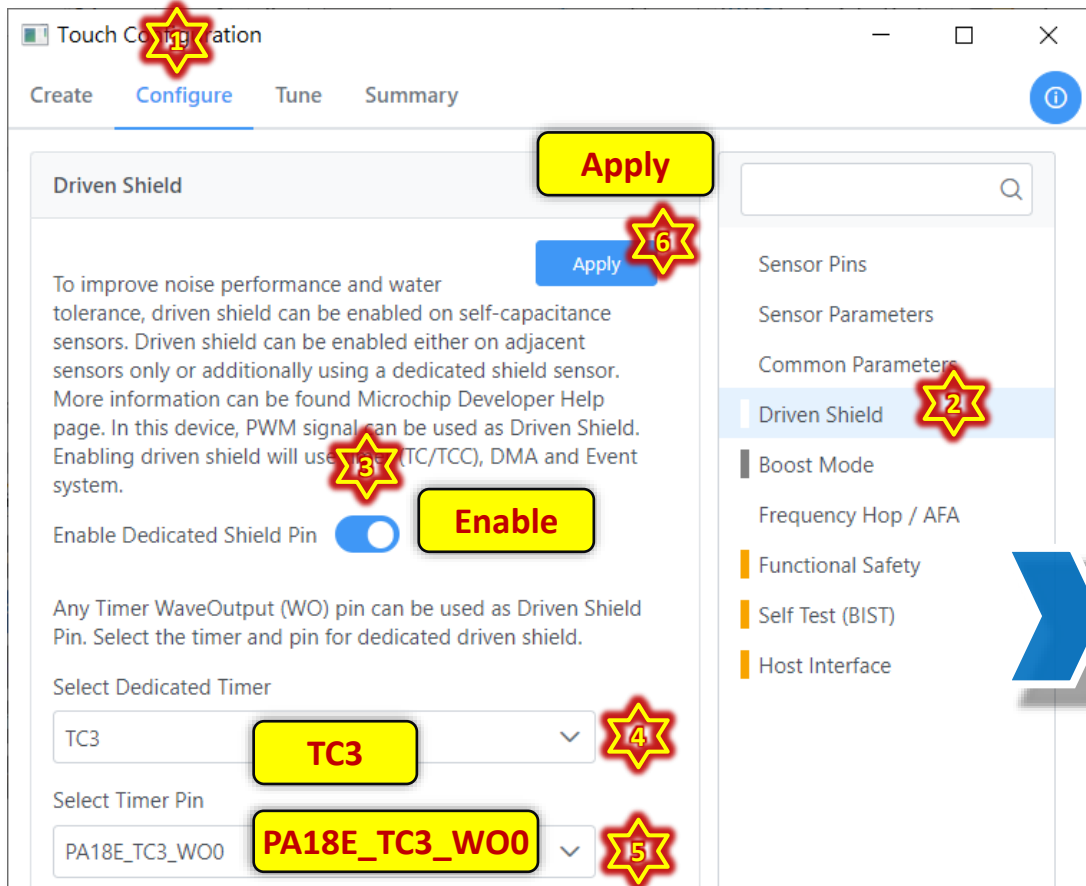
AKS Group 1

AKS Group 2

✖ Lab1 Touch Buttons

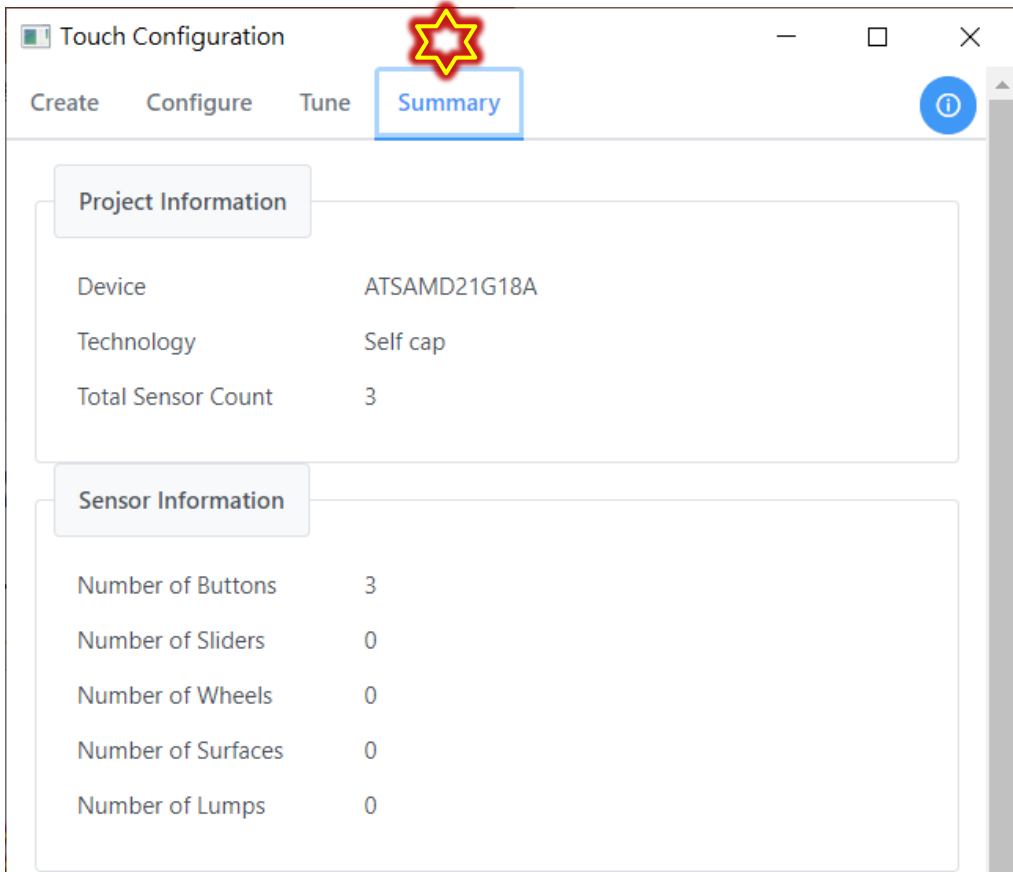
- In **Configure** Tab, Click on **Driven Shield** and configure Driven Shield as below.

- Enable Dedicated Driven Shield : 
- Select Dedicated Timer : TC3
- Select Timer Pin : PA18E_TC3_WO0



Lab1 Touch Buttons

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



Touch Configuration

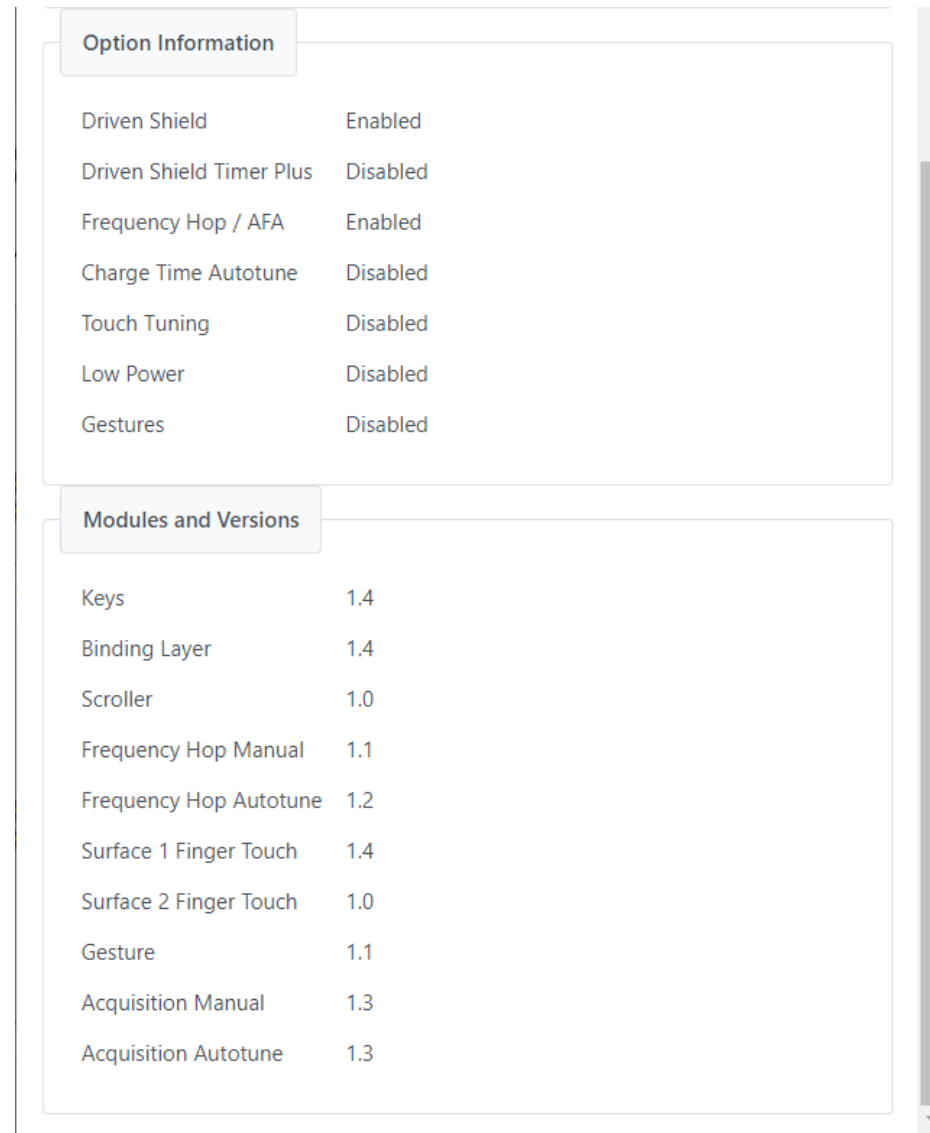
Create Configure Tune **Summary**

Project Information

Device	ATSAMD21G18A
Technology	Self cap
Total Sensor Count	3

Sensor Information

Number of Buttons	3
Number of Sliders	0
Number of Wheels	0
Number of Surfaces	0
Number of Lumps	0



Option Information

Driven Shield	Enabled
Driven Shield Timer Plus	Disabled
Frequency Hop / AFA	Enabled
Charge Time Autotune	Disabled
Touch Tuning	Disabled
Low Power	Disabled
Gestures	Disabled

Modules and Versions

Keys	1.4
Binding Layer	1.4
Scroller	1.0
Frequency Hop Manual	1.1
Frequency Hop Autotune	1.2
Surface 1 Finger Touch	1.4
Surface 2 Finger Touch	1.0
Gesture	1.1
Acquisition Manual	1.3
Acquisition Autotune	1.3

Lab1 Touch Buttons

- Check the Touch Sensor Pins 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			
9	PA04		PTC_Y2	Analog	High Impedance	n/a	☐	☐	NORMAL			
10	PA05		PTC_Y3	Analog	High Impedance	n/a	☐	☐	NORMAL			
47	PB02		PTC_Y8	Analog	High Impedance	n/a	☐	☐	NORMAL			

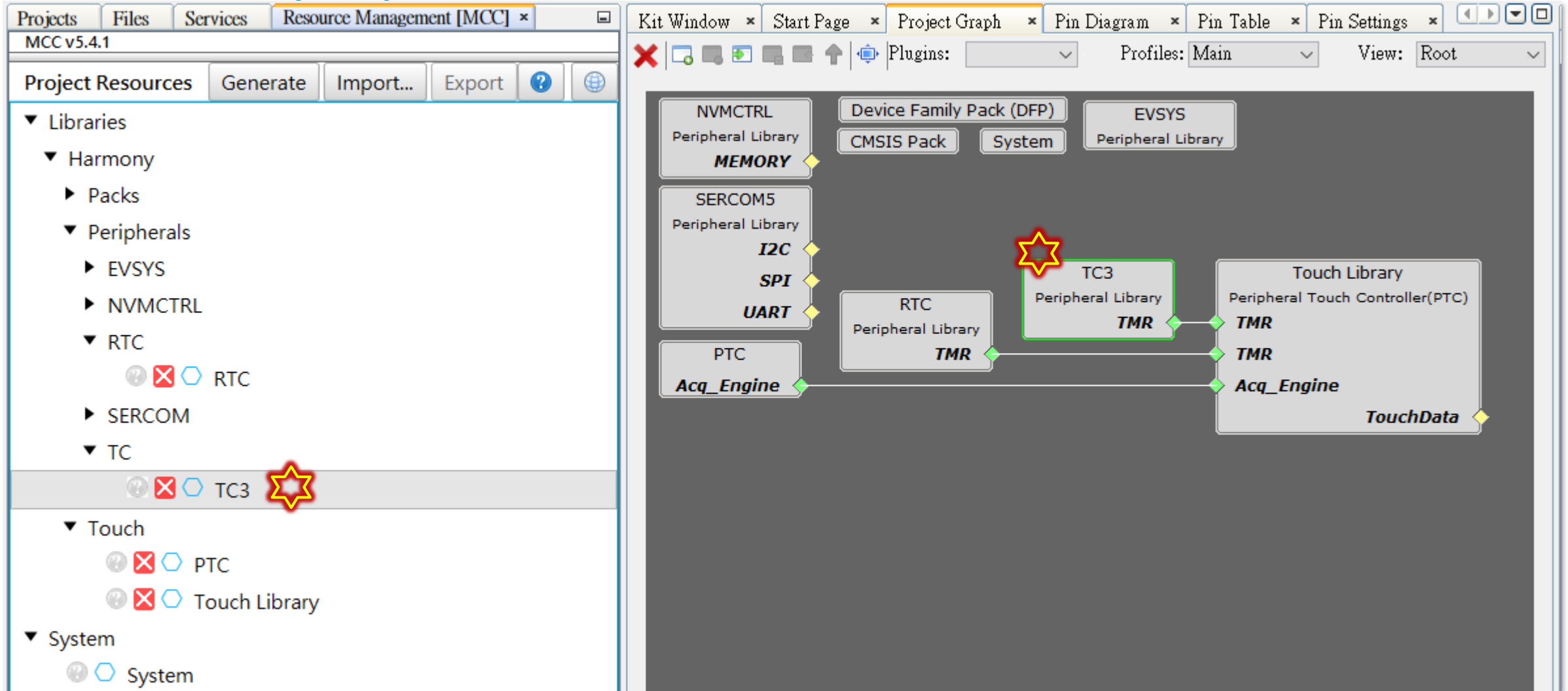
Lab1 Touch Buttons

- 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 ▼

⚙️ Lab1 Touch Buttons

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



Lab1 Touch Buttons

API functions in Touch.c

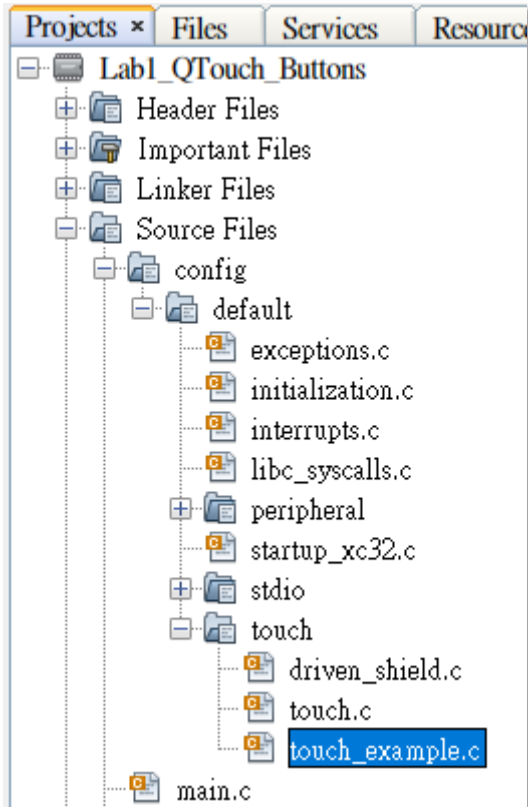
<code>uint8_t measurement_done_touch</code>	Touch acquisition done flag
<code>touch_process ()</code>	Main processing function of touch library. This function initiates the acquisition, calls post processing after the acquisition complete and sets the flag for next measurement based on the sensor status.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

API functions in driven_shield.c

<code>drivenshield_configure ()</code>	Initialization Driven Shield for Touch in touch_init()
--	--

⚙️ Lab1 Touch Buttons

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



```
#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
    }
}
```

```
/*=====
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");
    ...
}
```

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;
            }

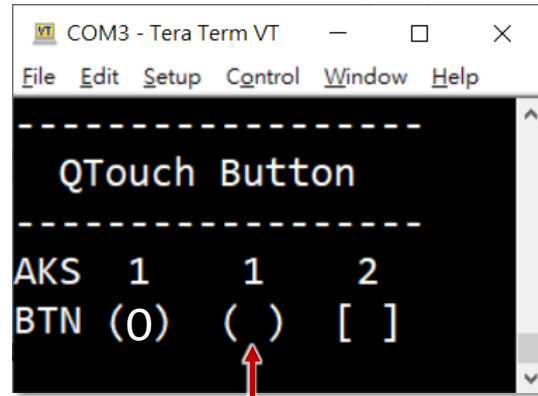
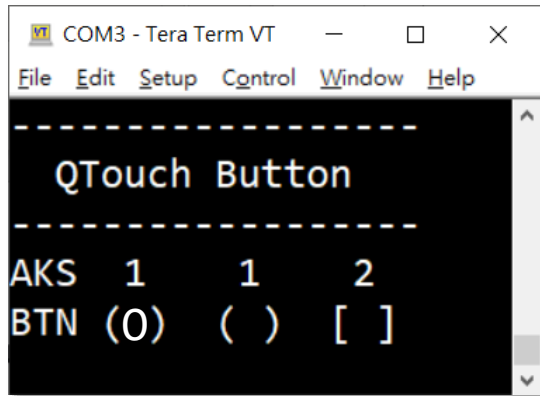
            switch( get_sensor_state(2) )
            {
                case QTM_KEY_STATE_DETECT:  KeyStatus[2]=true;  LED3_Set();  break;
                case QTM_KEY_STATE_NO_DET:  KeyStatus[2]=false; LED3_Clear(); break;
            }

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

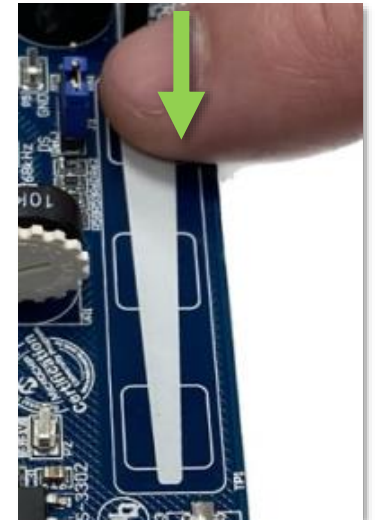
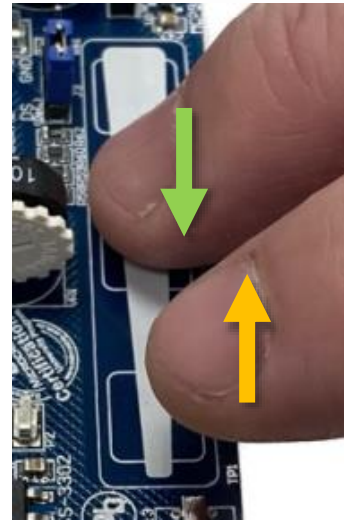
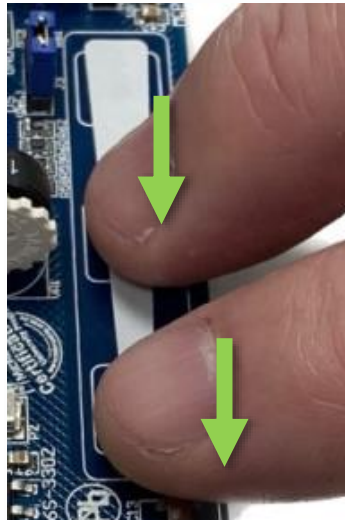
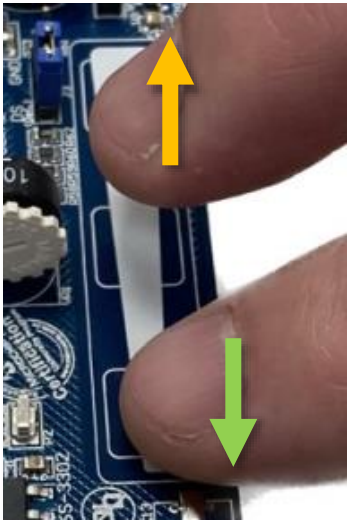
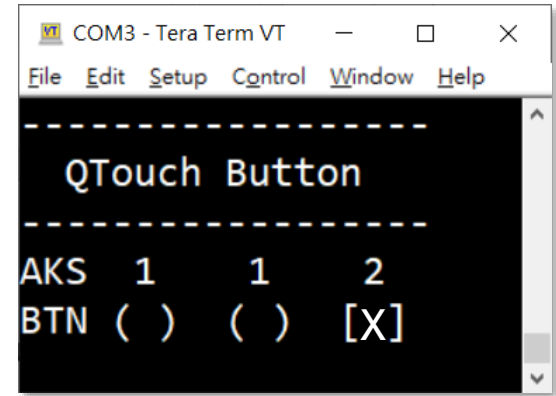
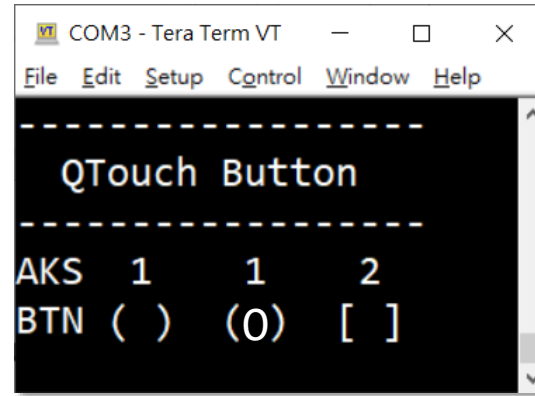
        }
        ...
    }
}
```

⚙️ Lab1 Touch Buttons

- Result of Touch Key implementation.



(Adjacent **R**ed **S**uppression)



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);
        }

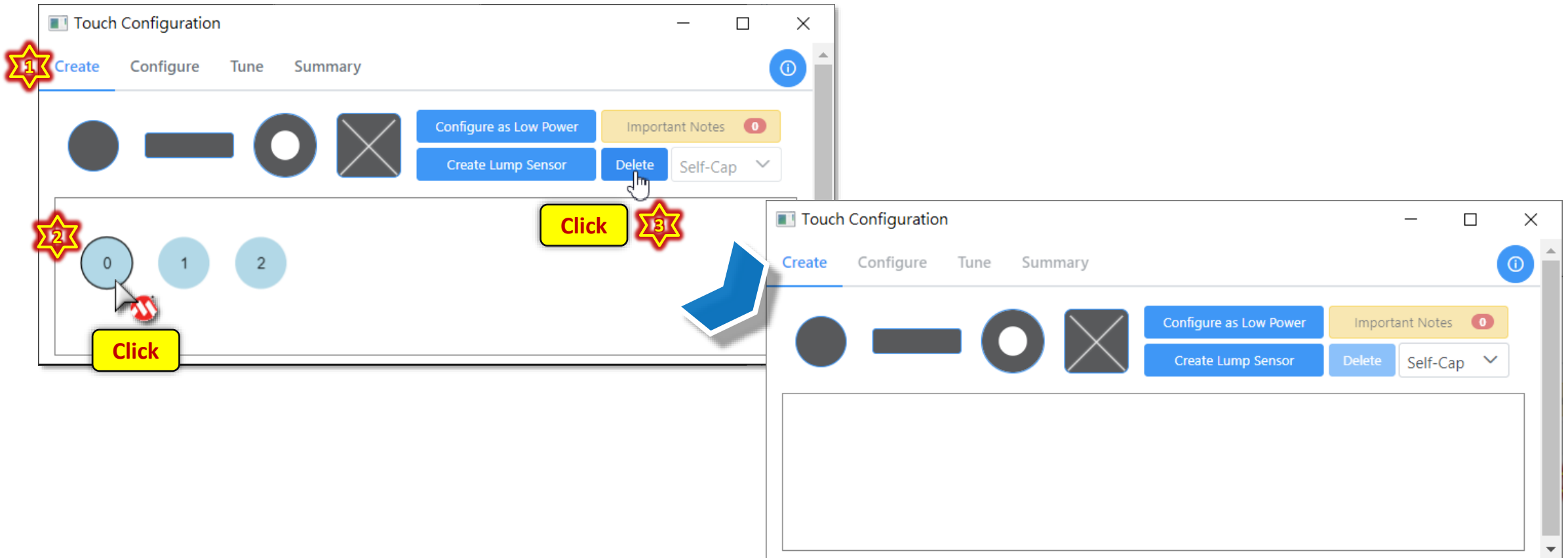
        if( measurement_done_touch == 1 )
        {
            measurement_done_touch = 0;
            switch( get_sensor_state(0) )
            {
                case QTM_KEY_STATE_DETECT:  KeyStatus[0]=true;  LED1_Set();  break;
                case QTM_KEY_STATE_NO_DET:   KeyStatus[0]=false; LED1_Clear();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

Lab2 Touch Slider

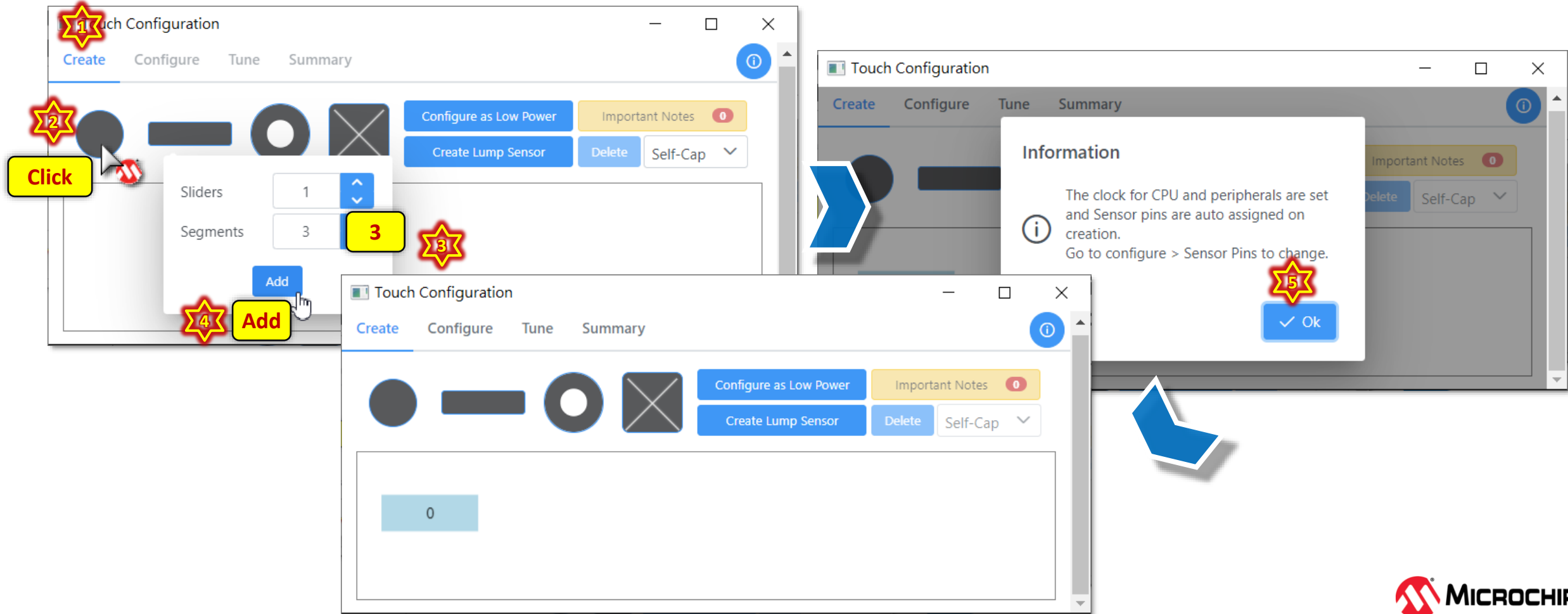
Lab2 Touch Slider

- 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”.



The image illustrates the steps to create a touch slider in the Microchip Studio IDE:

1. Open the **Touch Configuration** window.
2. Click on the **Slider** icon.
3. Change the number of **Segments** to **3**.
4. Click the **Add** button.
5. An **Information** dialog box appears, stating: "The clock for CPU and peripherals are set and Sensor pins are auto assigned on creation. Go to configure > Sensor Pins to change." Click **Ok**.

✂ 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)

Touch Configuration

Create **Configure** Tune Summary

Sensor Pins

Signals will be swapped if it's used by other sensor/segment during selection

Sensors	Signals
Slider 0	 Y8 (PB02) Y8 (PB02)
	 Y2 (PA04) Y2 (PA04)
	 Y3 (PA05) Y3 (PA05)

Sensor Pins 

Sensor Parameters

Common Parameters

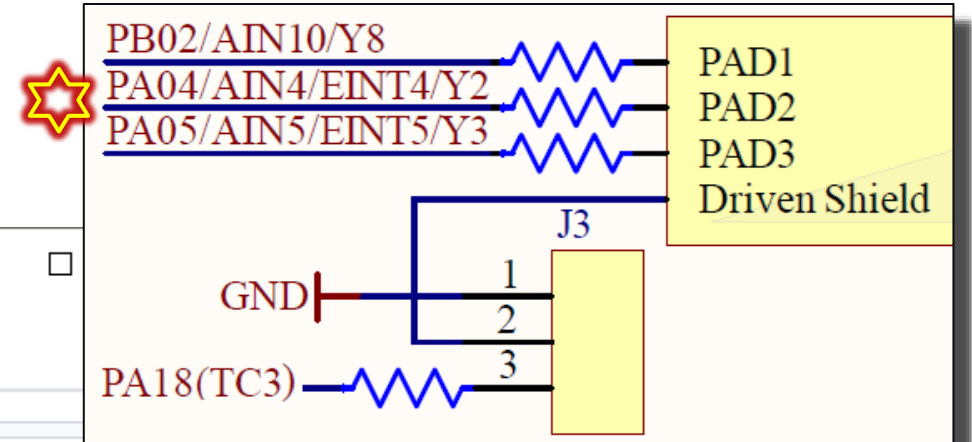
Driven Shield

Boost Mode

Frequency Hop / AFA

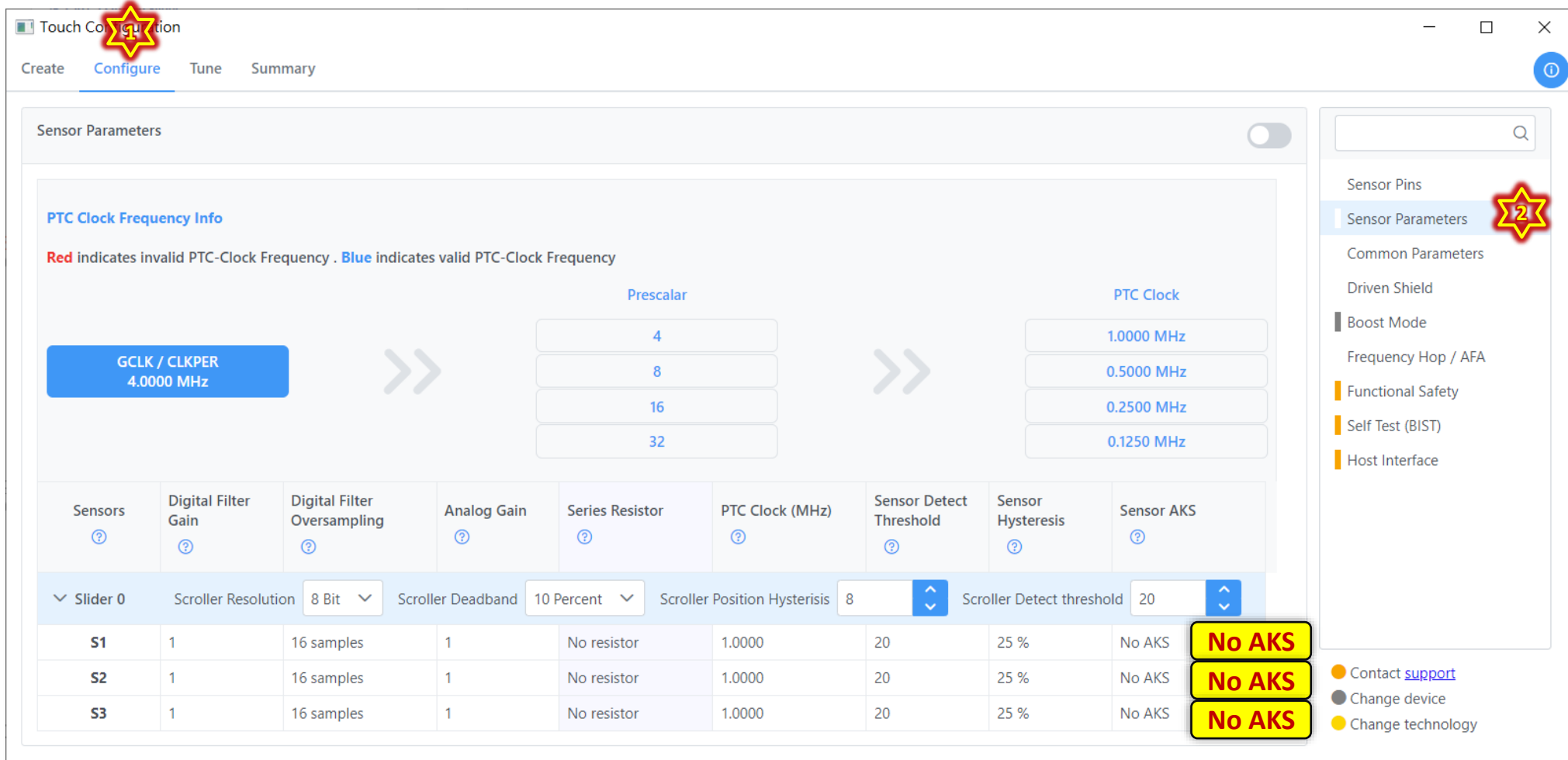
Functional Safety

Self Test (BIST)



Lab2 Touch Slider

- In **Configure** Tab, Click on **Sensor Parameters** and check the configuration of **Sensor AKS**(Adjacent **K**ey **S**uppression) group as below.
- S1 : no AKS, S2 : no AKS, S3 : no AKS



Touch Configuration

Create **Configure** Tune Summary

Sensor Parameters

PTC Clock Frequency Info

Red indicates invalid PTC-Clock Frequency . Blue indicates valid PTC-Clock Frequency

Prescalar

PTC Clock

GCLK / CLKPER
4.0000 MHz

4
8
16
32

1.0000 MHz
0.5000 MHz
0.2500 MHz
0.1250 MHz

Sensors	Digital Filter Gain	Digital Filter Oversampling	Analog Gain	Series Resistor	PTC Clock (MHz)	Sensor Detect Threshold	Sensor Hysteresis	Sensor AKS
S1	1	16 samples	1	No resistor	1.0000	20	25 %	No AKS
S2	1	16 samples	1	No resistor	1.0000	20	25 %	No AKS
S3	1	16 samples	1	No resistor	1.0000	20	25 %	No AKS

Slider 0 Scroller Resolution 8 Bit Scroller Deadband 10 Percent Scroller Position Hysteresis 8 Scroller Detect threshold 20

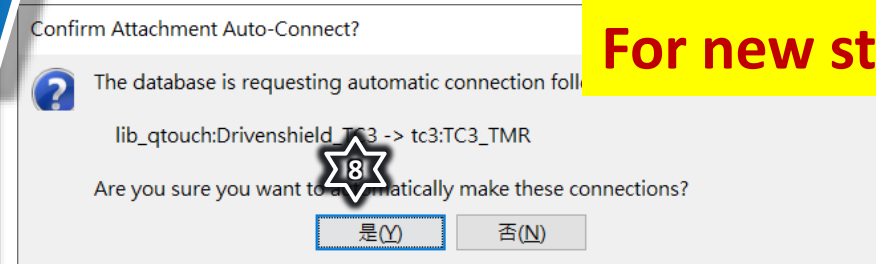
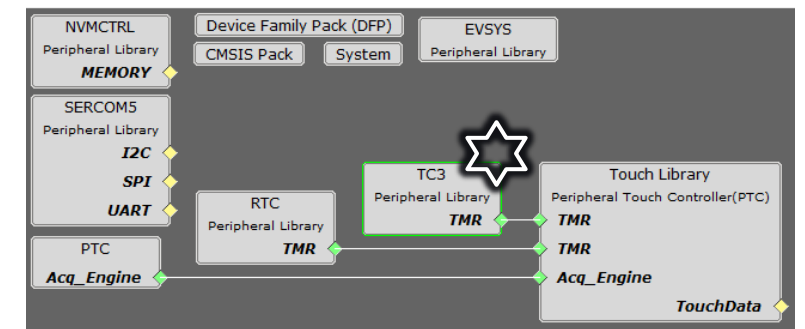
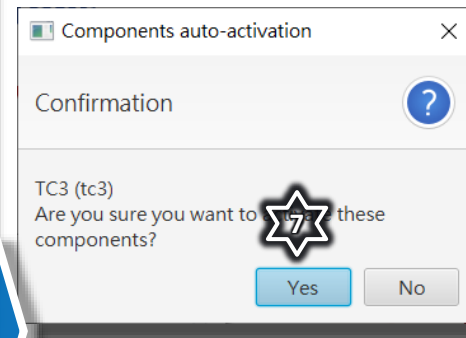
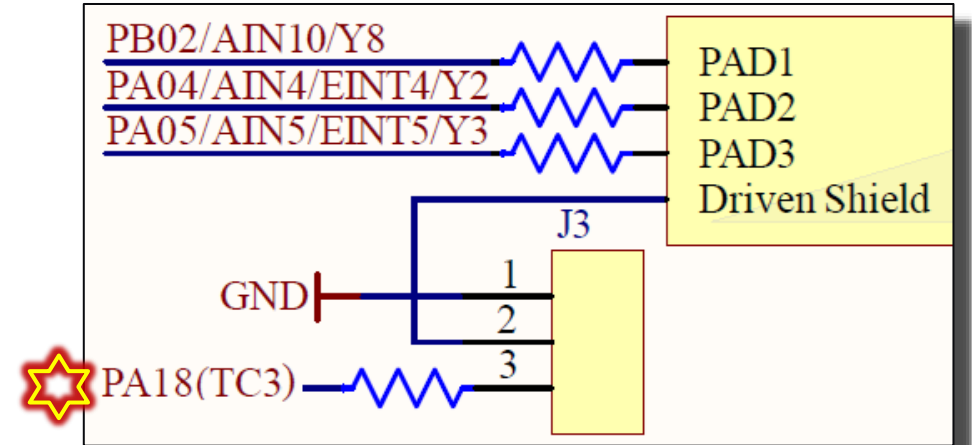
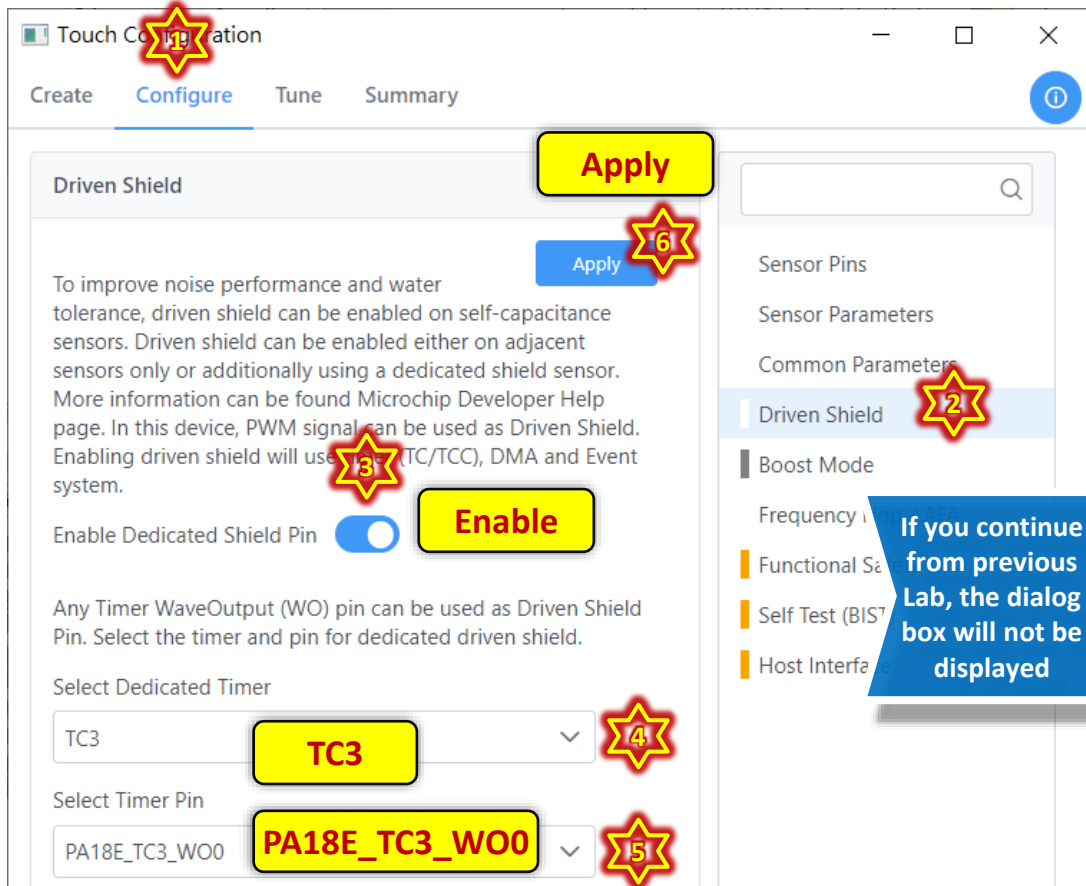
No AKS
No AKS
No AKS

Contact [support](#)
Change device
Change technology

✖ Lab2 Touch Slider

- In **Configure** Tab, Click on **Driven Shield** and configure Driven Shield as below.

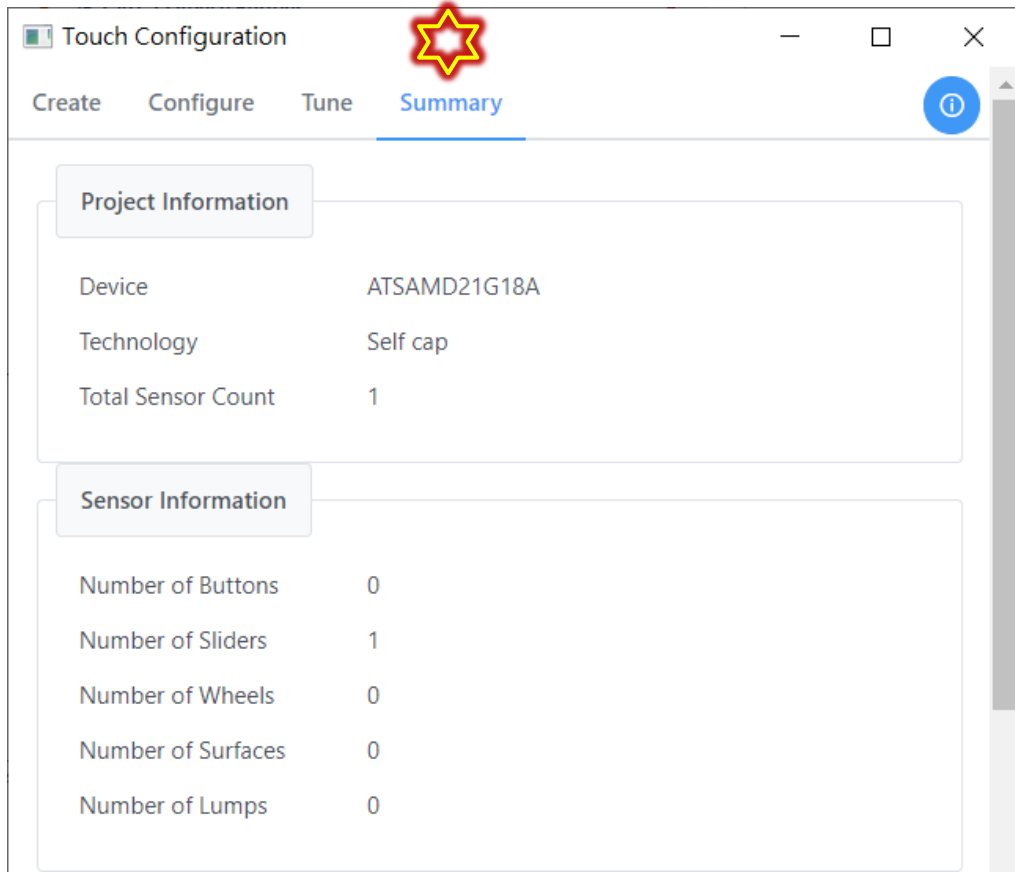
- Enable Dedicated Driven Shield : 
- Select Dedicated Timer : TC3
- Select Timer Pin : PA18E_TC3_WO0



For new start project only

Lab2 Touch Slider

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



Touch Configuration

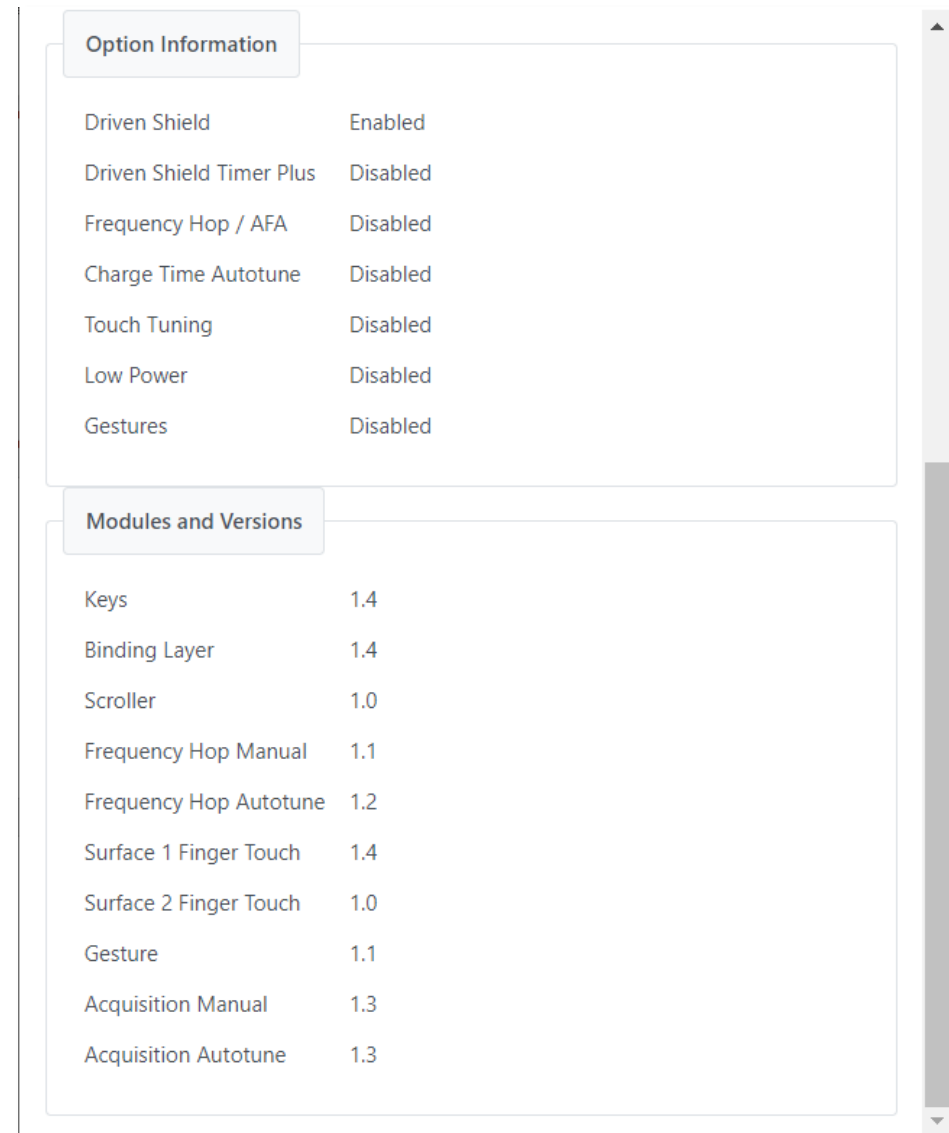
Create Configure Tune **Summary**

Project Information

Device	ATSAMD21G18A
Technology	Self cap
Total Sensor Count	1

Sensor Information

Number of Buttons	0
Number of Sliders	1
Number of Wheels	0
Number of Surfaces	0
Number of Lumps	0



Option Information

Driven Shield	Enabled
Driven Shield Timer Plus	Disabled
Frequency Hop / AFA	Disabled
Charge Time Autotune	Disabled
Touch Tuning	Disabled
Low Power	Disabled
Gestures	Disabled

Modules and Versions

Keys	1.4
Binding Layer	1.4
Scroller	1.0
Frequency Hop Manual	1.1
Frequency Hop Autotune	1.2
Surface 1 Finger Touch	1.4
Surface 2 Finger Touch	1.0
Gesture	1.1
Acquisition Manual	1.3
Acquisition Autotune	1.3

Lab2 Touch Slider

- Check the Touch Sensor Pins 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
9	PA04		PTC_Y2	Analog	High Impedance	n/a	☐	☐	NORMAL
10	PA05		PTC_Y3	Analog	High Impedance	n/a	☐	☐	NORMAL
47	PB02		PTC_Y8	Analog	High Impedance	n/a	☐	☐	NORMAL

Lab2 Touch Slider

- 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									
For new start project only									
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.

The screenshot displays the Microchip Harmony IDE interface. On the left, the **Resource Management [MCC]** window shows a tree view of project resources. Under the **Libraries** section, the **TC3** library is highlighted with a red star icon. The **Project Graph** window on the right shows a diagram of the project components. The **TC3** library is highlighted with a red star icon. The **Project Graph** window also shows the **Touch Library** and **PTC** components. A yellow box at the bottom right of the **Project Graph** window contains the text: **For new start project only**.

Resource Management [MCC] v5.4.1

- Project Resources: Generate, Import..., Export
- Libraries
 - Harmony
 - Packs
 - Peripherals
 - EVSYS
 - NVMCTRL
 - RTC
 - RTC
 - SERCOM
 - TC
 - TC3
 - Touch
 - PTC
 - Touch Library
 - System
 - System

Project Graph

- Kit Window, Start Page, Project Graph, Pin Diagram, Pin Table, Pin Settings
- Plugins: Profiles: Main View: Root
- Components: NVMCTRL (Peripheral Library, MEMORY), Device Family Pack (DFP), CMSIS Pack, System, EVSYS (Peripheral Library), SERCOM5 (Peripheral Library, I2C, SPI, UART), RTC (Peripheral Library, TMR), PTC (Acq_Engine), TC3 (Peripheral Library, TMR), Touch Library (Peripheral Touch Controller(PTC), TMR, Acq_Engine, TouchData)

For new start project only

Lab2 Touch Slider

API functions in Touch.c

uint8_t measurement_done_touch
touch_process ()

touch_init ()

touch_timer_config ()

touch_sensors_config ()

get_sensor_node_signal ()

update_sensor_node_signal ()

get_sensor_node_reference ()

update_sensor_node_reference ()

get_sensor_cc_val ()

update_sensor_cc_val ()

get_sensor_state ()

update_sensor_state ()

calibrate_node ()

get_scroller_state

get_scroller_position

Touch acquisition done flag

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.

Initialization PTC, timer, and datastreamer modules of touch library.

Initialization necessary RTC timer for Touch in touch_init()

Initialization of touch key sensors in touch_init()

Get sensor node acquisition signals.

Update sensor node acquisition signals.

Get sensor node channel reference.

Update sensor node channel reference.

Get sensor node compensation capacitor value.

Update sensor node compensation capacitor value.

Get sensor node state

Update sensor node state

Sensor node calibration

Get scroller sensor state

Get scroller position

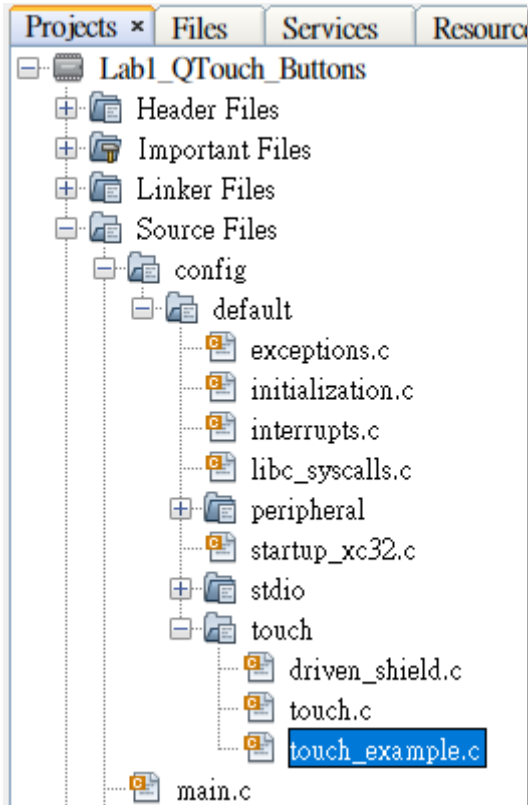
API functions in driven_shield.c

drivenshield_configure ()

Initialization Driven Shield for Touch in touch_init()

Lab2 Touch Slider

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



```
#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
    }
}
```

```
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;
    }
}
```

```
/*=====
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
    }
}
```

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("0");
                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();

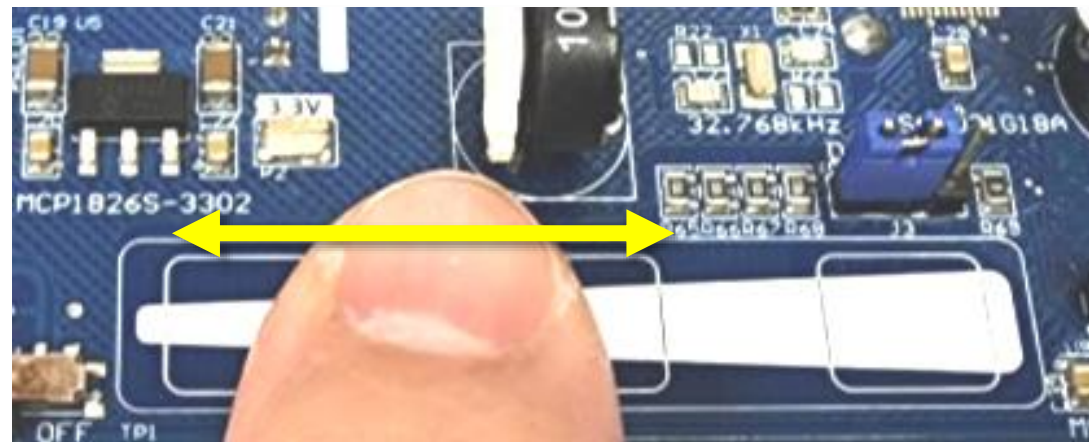
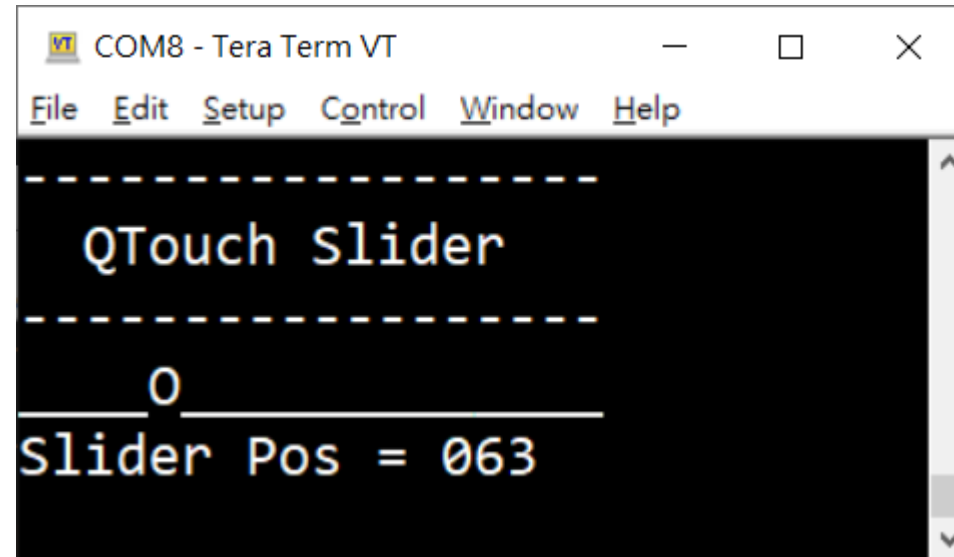
        if(BT1_Get()==0)
        {
            while(BT1_Get()==0);
            calibrate_node(0);
            calibrate_node(1);
            calibrate_node(2);
        }

        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("0");
                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;
                    ...
                }
            }
        }
    }
}
```

⚙️ Lab2 Touch Slider

- Result of Touch Slider(Scroller) implementation.



Lab3 Data Visualizer for Touch

✖ Lab3 Data Visualizer for Touch

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

MPLAB X IDE v6.00 - Lab3_QTouch_DataVisualizer : default

File Edit MHC View Navigate Source Refactor Production Debug Team Tools Window Help

Workspace: Clear Load... Save Save As... Views: Time Plot Dashboard More

Connections: Lab3_QTouch_DataVisualizer COM3 on COM3

Variable Streamers: (x) FrameCounter UInt8 (x) Signal0 UInt16 (x) Reference0 UInt16 (x) Delta0 Int16 Compensation0

Lab3_QTouch_DataVisualizer Options

Auto-config enabled: ☒

Auto-config

Config ID: 03EB00000000000000AA5501

Search path: C:\Exercises_SAM2002ADV\Lab3_QTouch_DataVisualizer\firmware\src

Recursive search: ☒

Maximize dashboard: ☒

Statistics

Time Plot x Dashboard x

BasicData Graph AdvanceData AllData

QTouch Modular Library User Guide: www.microchip.com
Glossary: www.microchip.com

Channel ID	Sensor Type	State	Delta	Threshold
0	Slider 0[0]	0	1	20
1	Slider 0[1]	1	141	20
2	Slider 0[2]	1	79	20

Sensor	State	SW Delta	SW Threshold
Slider 0	1	217	20

Channel ID	Sensor Type	Signal	Reference	Delta	Compensation pF
0	Slider 0[0]	513	512	1	19.75725
1	Slider 0[1]	650	509	141	20.76975
2	Slider 0[2]	591	512	79	23.06475

Variable	Value
FrameCounter on Lab3_QTouch_DataVisualizer	74
QTouchLibError on Lab3_QTouch_DataVisualizer	0

Counter for datastreamer packets. Missing count indicate packet drop
Indicates library error state. Zero: no error. Refer "Error Code" section in User Guide

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

Marker: 5075.7s
Data values: 513, 0, 0, 0, 20, 20, 20, 0, 20

Vertical 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.

Signal0 on Lab3_QTouch_DataVisualizer Signal1 on Lab3_QTouch_DataVisualizer Signal2 on Lab3_QTouch_DataVisualizer Reference0 on Lab3_QTouch_DataVisualizer Reference1 on Lab3_QTouch_DataVisualizer
Reference2 on Lab3_QTouch_DataVisualizer Delta0 on Lab3_QTouch_DataVisualizer Delta1 on Lab3_QTouch_DataVisualizer Delta2 on Lab3_QTouch_DataVisualizer Threshold0 on Lab3_QTouch_DataVisualizer
Threshold1 on Lab3_QTouch_DataVisualizer Threshold2 on Lab3_QTouch_DataVisualizer SWDelta0 on Lab3_QTouch_DataVisualizer SWThreshold0 on Lab3_QTouch_DataVisualizer

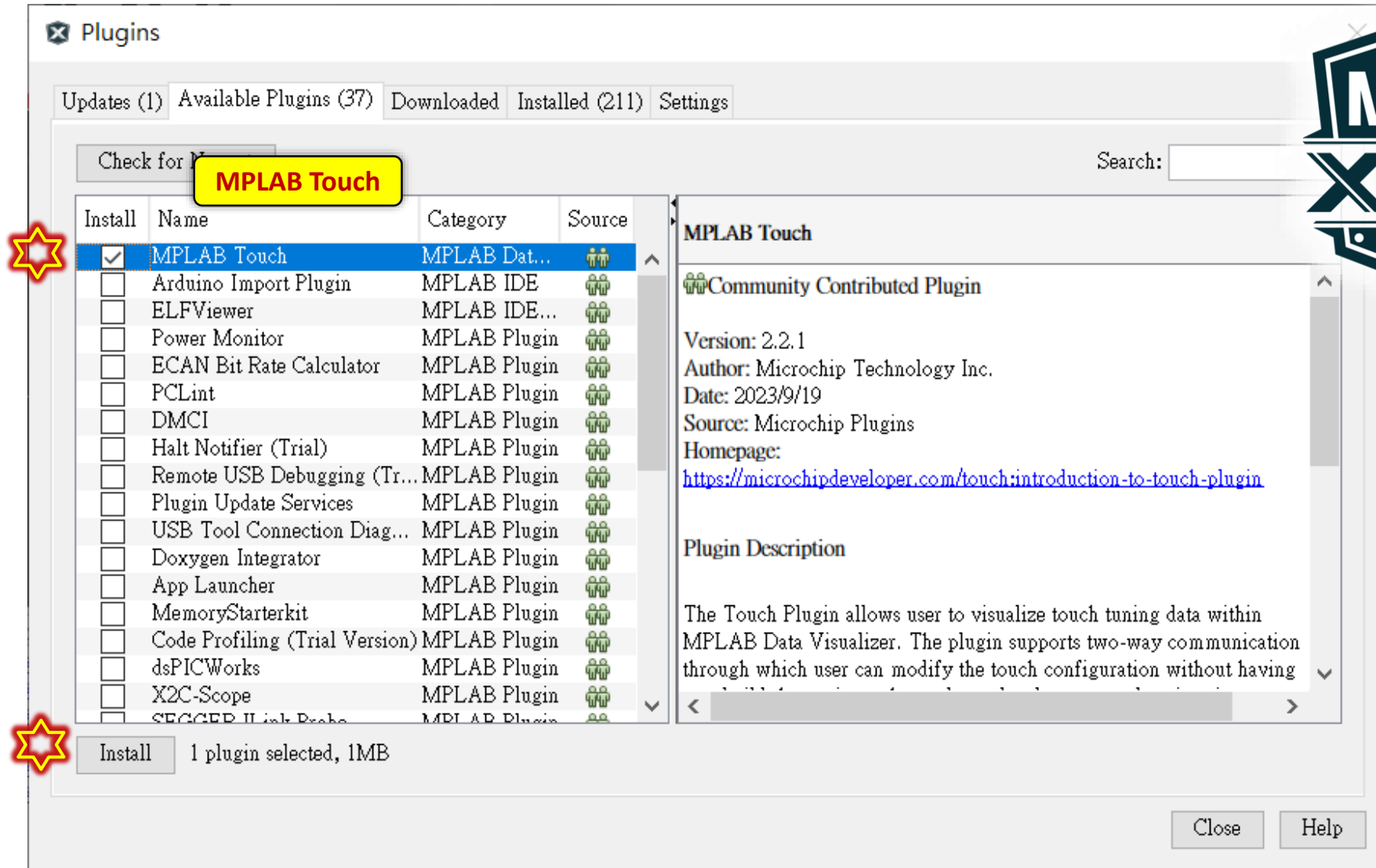
News Console Output

55:11



⚙️ 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)**

The screenshot shows the 'Touch Configuration' window with the 'Tune' tab selected. The window has a title bar with standard OS controls and a blue information icon. Below the title bar are tabs for 'Create', 'Configure', 'Tune' (active), and 'Summary'. The main content area contains text explaining 'Touch Visualization' and its requirements. It includes a toggle switch for 'Enable Touch Tuning Data connection' (labeled with a yellow star '2') and three radio button options for visualization (labeled with a yellow star '3'). The first option is 'Touch Tuning (bidirectional, requires UART Tx and Rx)'. The second option, 'Touch Visualization (unidirectional, requires UART Tx)', is selected and highlighted with a yellow box. The third option is 'Touch Tuning - 2D Surface (bidirectional, requires UART Tx and Rx)'. At the bottom, there is a section for 'Charge Time Autotune' with a toggle switch.

Touch Configuration

Create Configure **Tune** Summary

Touch Visualization enables to visualize relevant touch data on screen in real time. The communication is established using the UART capabilities of the touch MCU requiring **only TX** connected from Touch MCU to UART-USB bridge (such as [Microchip Touch Bridge](#) or [MCP2221A breakout module](#)). Use MPLAB Data Visualizer with [Auto-Configure](#) in Variable Streamers tab or [Atmel Data visualizer](#). MPLAB Data Visualizer exists within MPLAB X IDE as well as a standalone application.

This should be enabled and used for project development and validation and be disabled for production code to remove overhead, power consumption and flash/RAM footprint..

Enable Touch Tuning Data connection ☒ **Enable Touch Tuning Data connection**

☐ Touch Tuning (bidirectional, requires UART Tx and Rx)

3 ☒ **Touch Visualization (unidirectional, requires UART Tx)** **Touch Visualization (unidirectional, requires UART Tx)**

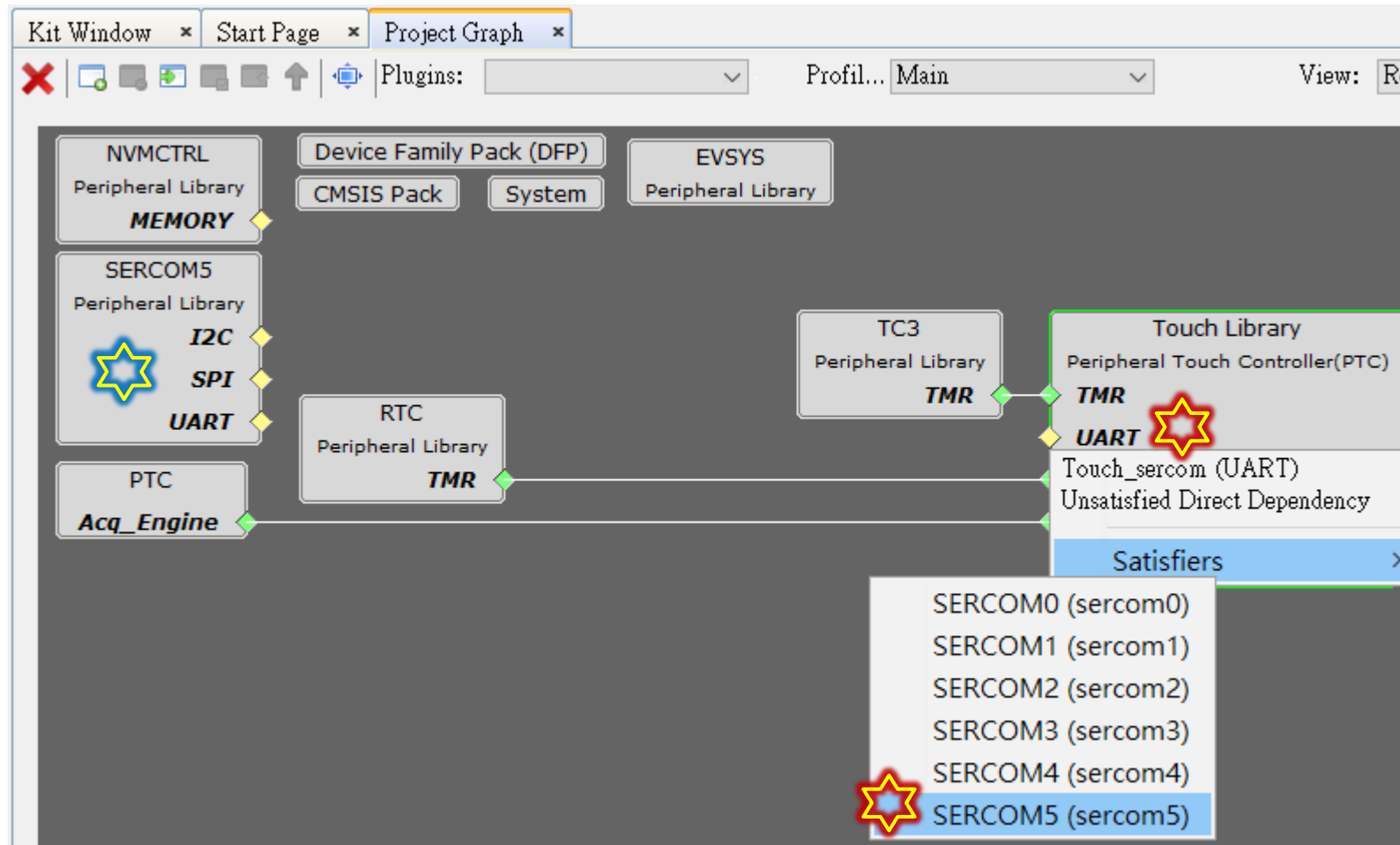
☐ Touch Tuning - 2D Surface (bidirectional, requires UART Tx and Rx)

Charge Time Autotune feature aids the tuning of sensor charge time. Manual tuning process is explained in [Microchip Developer Help page](#)
!!! Users must enable this feature ONLY during development and configure the recommended settings in touch.h file and disable this option in production application.

Enable Charge Time Autotune ☐

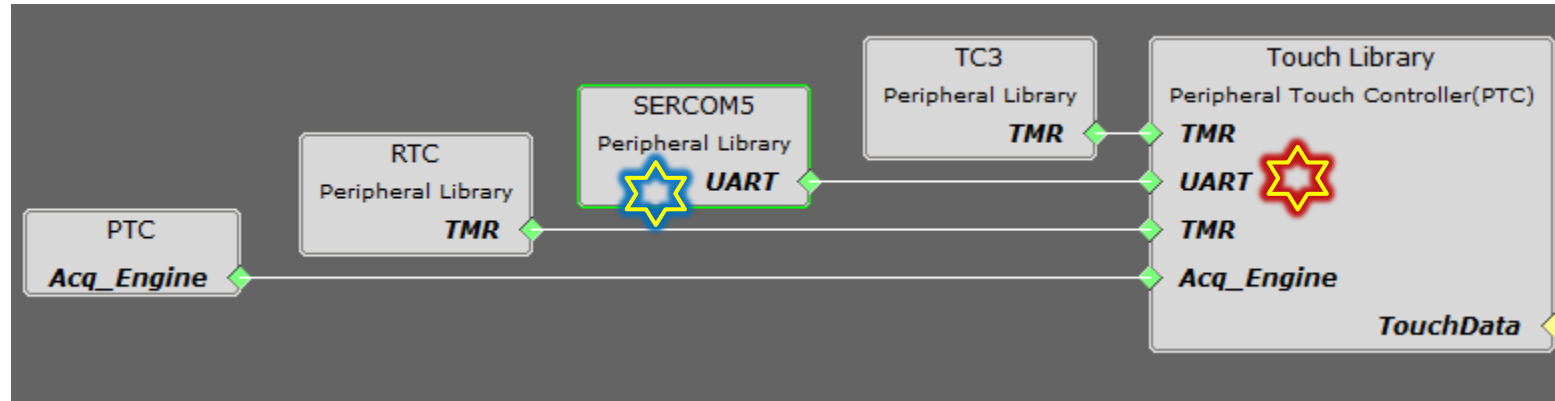
✖ Lab3 Data Visualizer for Touch

- 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**.



⚙️ 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

SERCOM5

Select SERCOM Operation Mode: USART with internal Clock

Operating Mode: Blocking mode **Use Blocking Mode**

Receive Enable: ☒

Transmit Enable: ☒

Frame Format: USART frame

Baud Rate in Hz: 115,200

Parity Mode: No Parity

Character Size: 8 Bits

Stop Bit Mode: One Stop Bit

Receive Pinout: SERCOM PAD[3] is used for data reception

Transmit Pinout: PAD[2] = TxD; PAD[3] = XCK

Enable Run in Standby: ☐

⚙️ 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 )
{
    ...

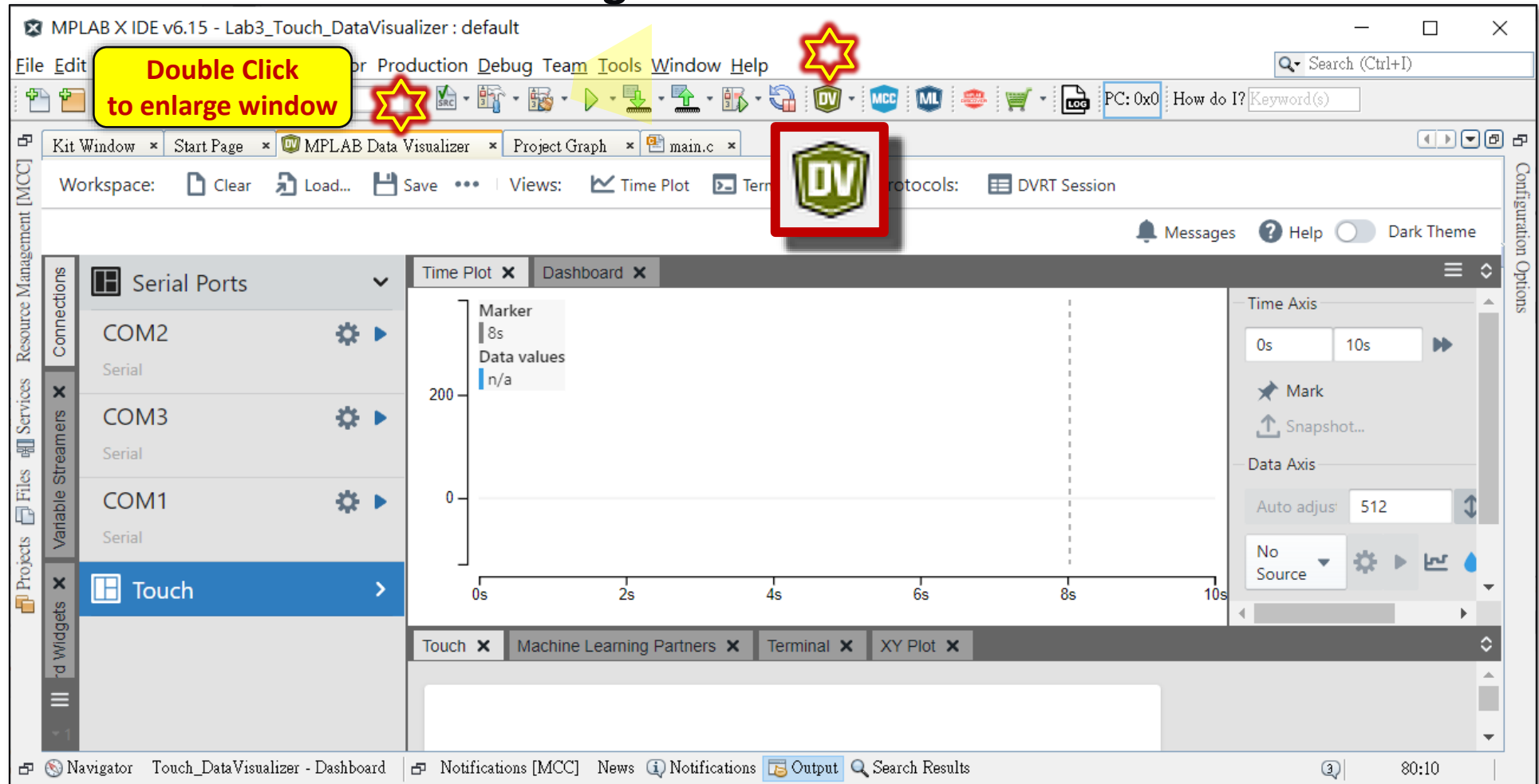
    while ( true )
    {
        touch_process();

        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("0");
                //myprintf("\033[5;1H");
                //myprintf("Slider Pos = %03d", ScrollPos);

                switch( ScrollPos*4/255 )
```

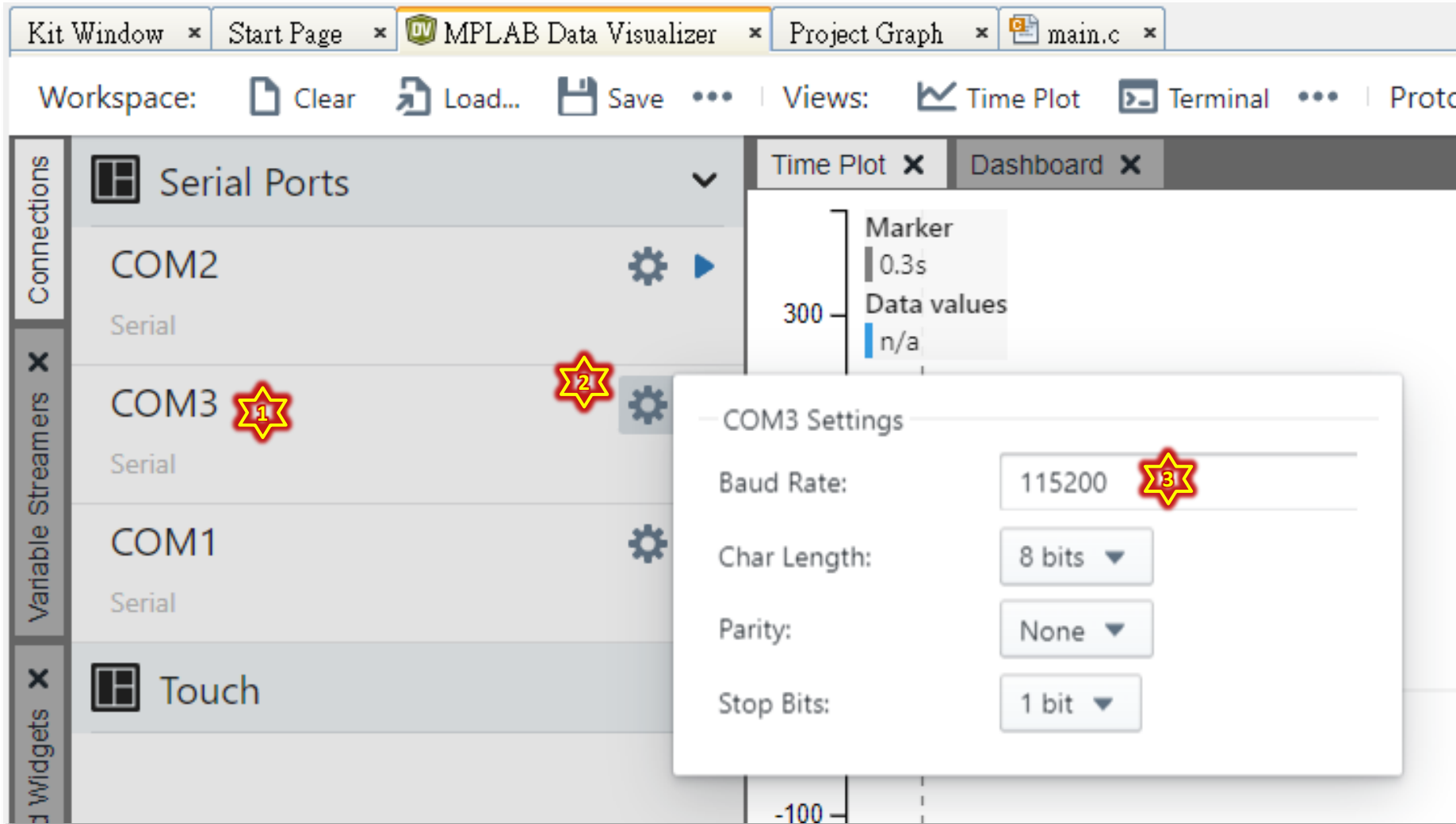
⚙️ Lab3 Data Visualizer for Touch

- Open **MPLAB® Data Visualizer** by simple click on icon  , you could **double 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**.



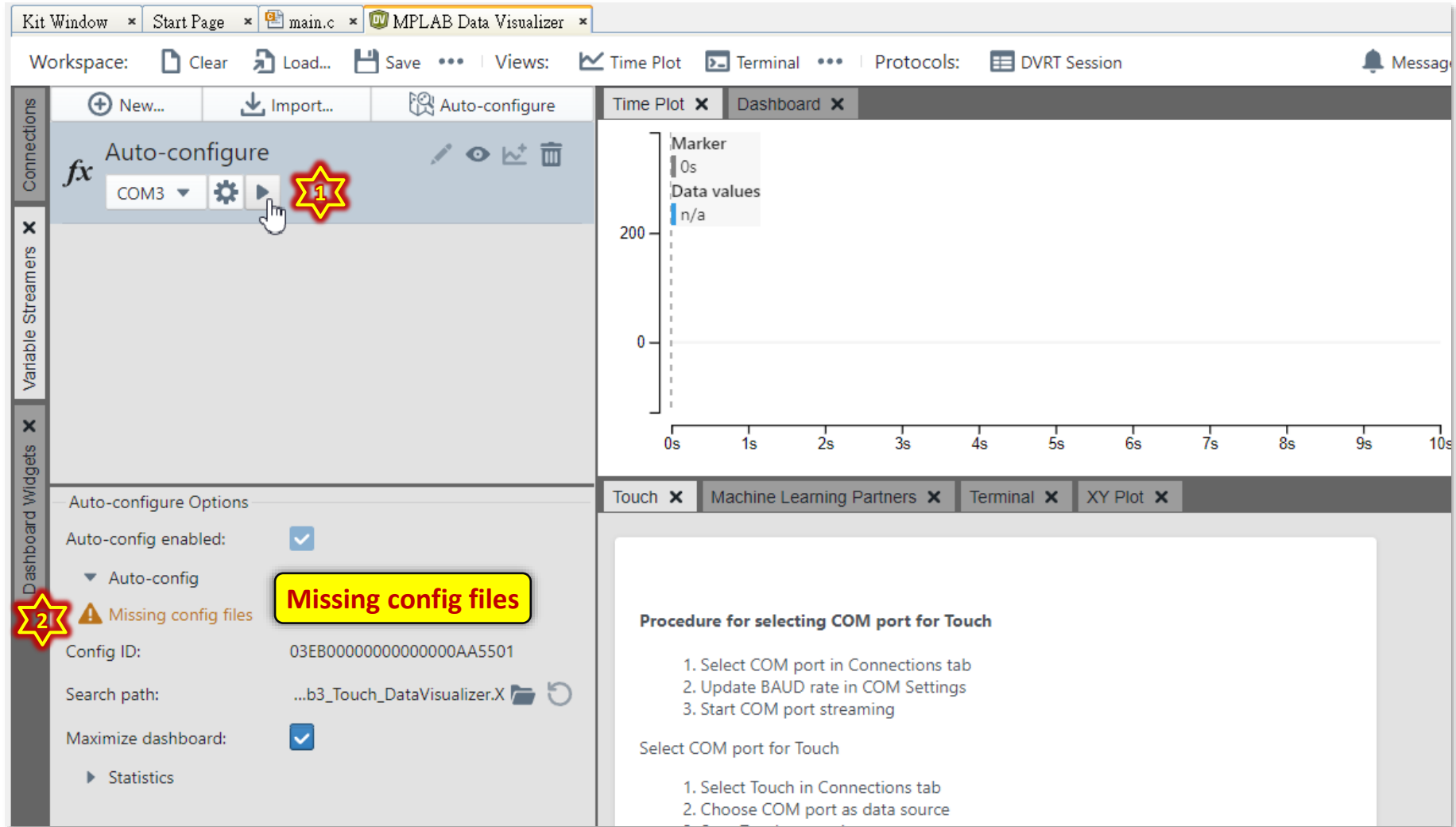
⚙️ Lab3 Data Visualizer for Touch

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

The screenshot displays the MPLAB Data Visualizer software interface. The 'Variable Streamers' tab is selected in the left sidebar, indicated by a yellow star with the number 1. The 'Auto-configure' button is highlighted with a yellow star with the number 2. A dropdown menu is open, showing the 'No Source' option selected, with a yellow star and the number 3 pointing to the 'COM3' option at the bottom of the list. The 'Auto-configure Options' section shows 'Auto-config enabled' checked. The 'Config ID' is 'Not decoded'. The 'Search path' is '...b3_Touch_DataVisualizer.X'. The 'Recursive search' and 'Maximize dashboard' options are also checked. The 'Time Plot' tab is active in the main window, showing a graph with 'Marker' at '0s' and 'Data values' at 'n/a'. The 'Touch' tab is selected in the bottom right panel, which contains a 'Procedure for selecting COM port for Touch' section with three steps: 1. Select COM port in Connections tab, 2. Update BAUD rate in COM Settings, and 3. Start COM port streaming. Below this, it says 'Select COM port for Touch' followed by two steps: 1. Select Touch in Connections tab and 2. Choose COM port as data source.

⚙️ Lab3 Data Visualizer for Touch

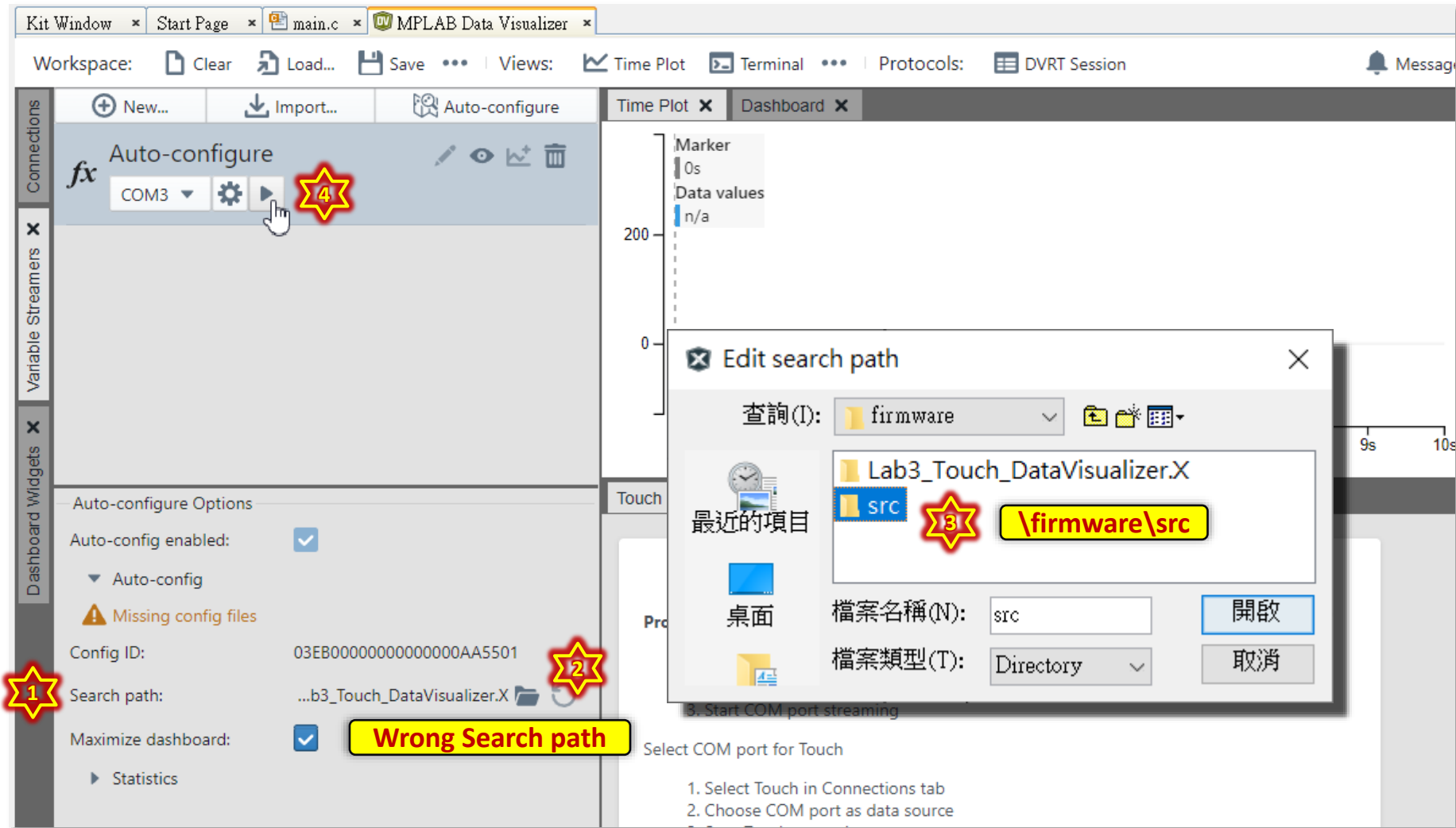
- Click on  to start communicate then you will see a **Miss configure files** message.



The screenshot displays the MPLAB Data Visualizer software interface. The top toolbar includes buttons for 'New...', 'Import...', and 'Auto-configure'. The 'Auto-configure' tab is active, showing a 'COM3' dropdown menu and a play button icon. A red star with the number '1' is placed over the play button. Below the 'Auto-configure' tab, the 'Auto-configure Options' section is visible, showing 'Auto-config enabled' checked, 'Config ID' as '03EB000000000000AA5501', and 'Search path' as '...b3_Touch_DataVisualizer.X'. A red star with the number '2' is placed over the 'Missing config files' warning icon. A yellow box with the text 'Missing config files' is overlaid on the 'Missing config files' warning. The right side of the interface shows a 'Time Plot' graph with a y-axis labeled 'Marker' and 'Data values' and an x-axis labeled '0s' to '10s'. The 'Data values' are 'n/a'. Below the graph, the 'Touch' tab is selected, displaying a 'Procedure for selecting COM port for Touch' with three steps: 1. Select COM port in Connections tab, 2. Update BAUD rate in COM Settings, 3. Start COM port streaming. Below the procedure, there is a section titled 'Select COM port for Touch' with two steps: 1. Select Touch in Connections tab, 2. Choose COM port as data source.

⚙️ Lab3 Data Visualizer for Touch

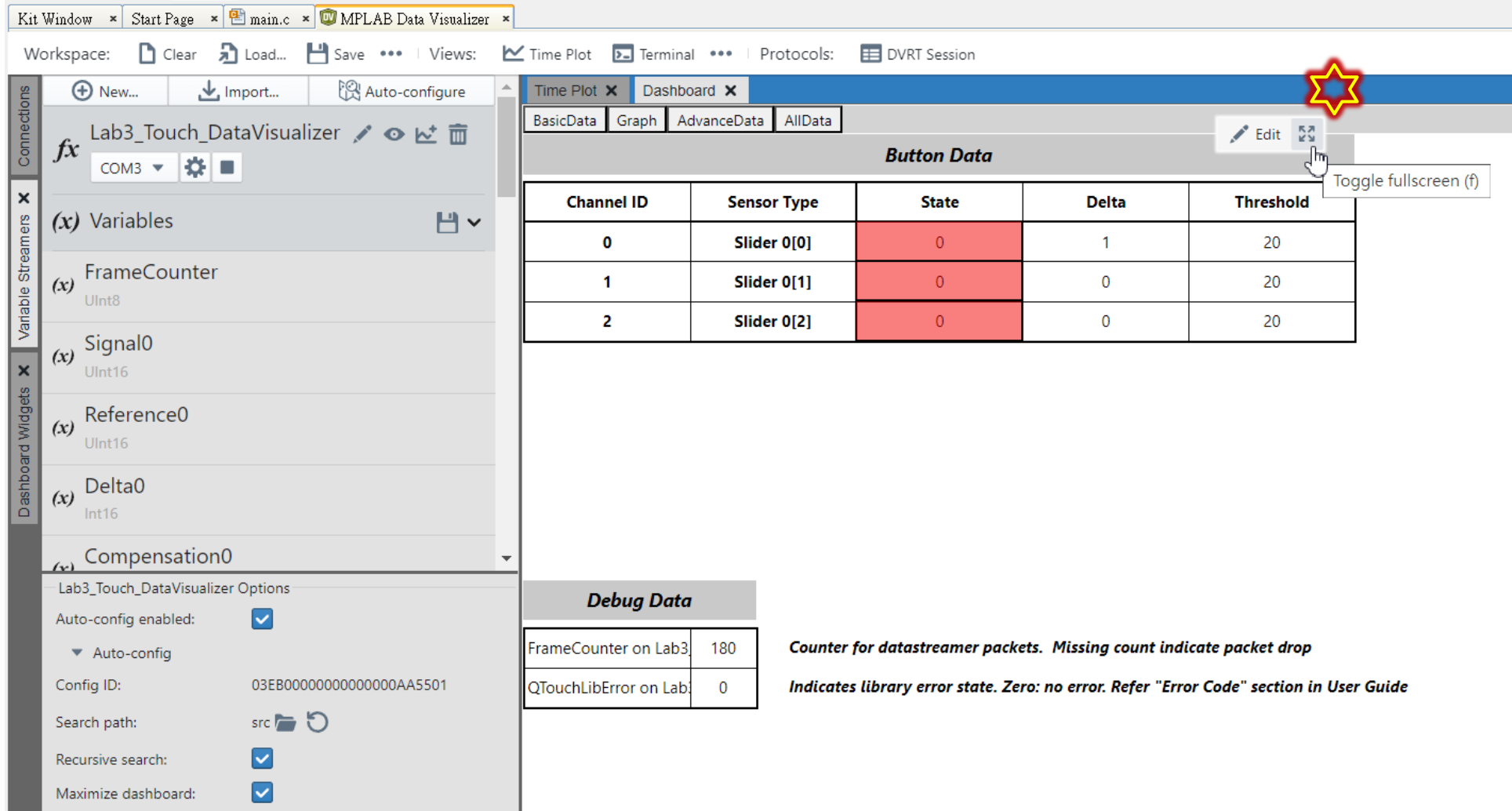
- The default Auto-configuration Options **Search Path** is wrong, please click on Icon  to change folder to **\firmware\src** then click  again.



The screenshot shows the MPLAB Data Visualizer interface. The 'Auto-configure' panel is active, displaying 'COM3' as the selected port. A red star with the number '4' is placed over the play button icon in the 'Auto-configure' section. Below this, the 'Auto-configure Options' section shows 'Auto-config enabled' checked. The 'Search path' is listed as '...b3_Touch_DataVisualizer.X', with a red star and the number '2' next to it. A yellow box with the text 'Wrong Search path' is overlaid on this section. A red star with the number '1' is placed over the folder icon next to the search path. An 'Edit search path' dialog box is open, showing a list of recent items. The path '\firmware\src' is highlighted with a red star and the number '3'. The 'Open' button is visible in the dialog box.

⚙️ 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.



Kit Window x Start Page x main.c x MPLAB Data Visualizer x

Workspace: Clear Load... Save ... Views: Time Plot Terminal ... Protocols: DVRT Session

Connections

fx Lab3_Touch_DataVisualizer

COM3

Variable Streamers

- (x) Variables
- (x) FrameCounter (UInt8)
- (x) Signal0 (UInt16)
- (x) Reference0 (UInt16)
- (x) Delta0 (Int16)
- (x) Compensation0

Dashboard Widgets

Lab3_Touch_DataVisualizer Options

- Auto-config enabled: ☒
- Auto-config
- Config ID: 03EB000000000000AA5501
- Search path: src
- Recursive search: ☒
- Maximize dashboard: ☒

Time Plot x Dashboard x

BasicData Graph AdvanceData AllData

Edit

Toggle fullscreen (f)

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

Debug Data

FrameCounter on Lab3	180
QTouchLibError on Lab3	0

Counter for datastreamer packets. Missing count indicate packet drop

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

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

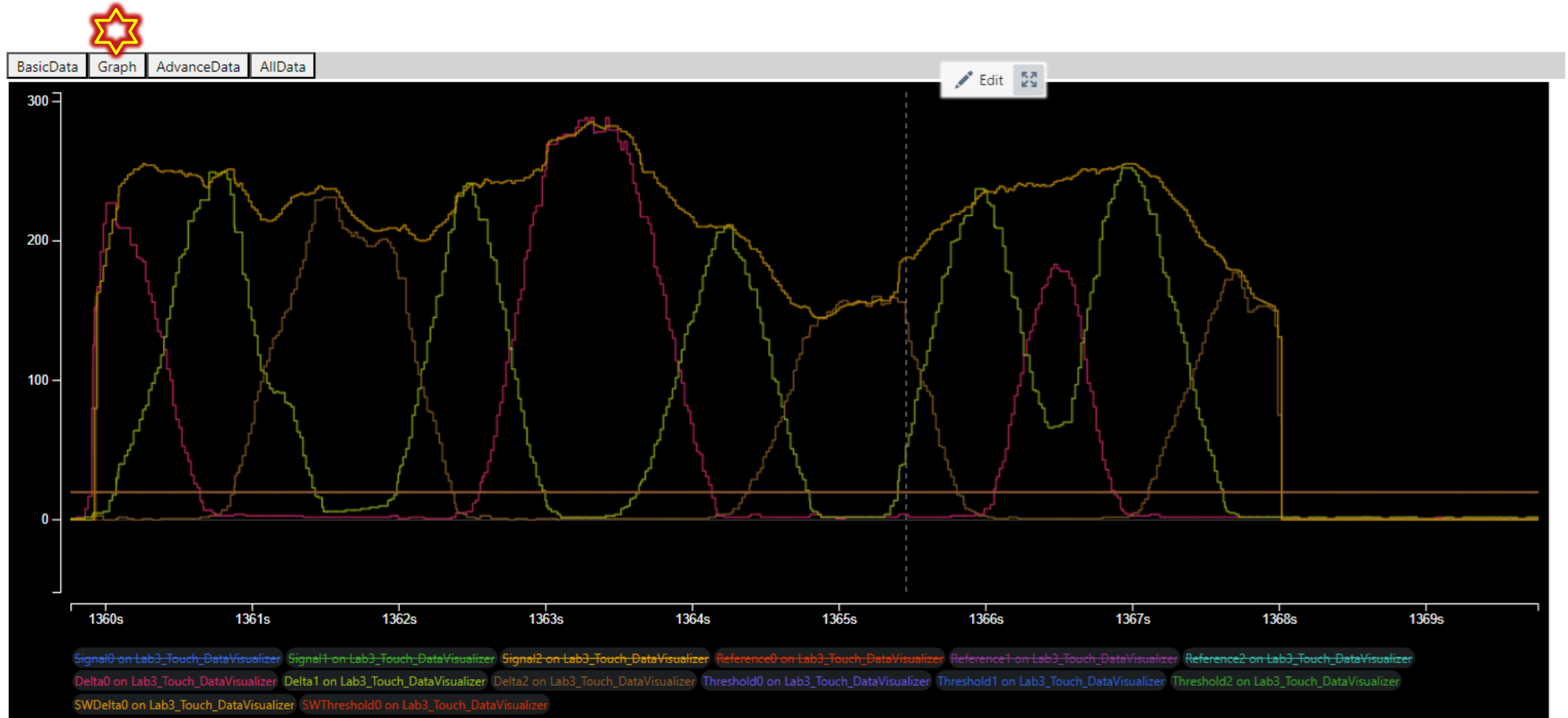
FrameCounter on Lab3	128
QTouchLibError on Lab	0

Counter for datastreamer packets. Missing count indicate packet drop

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

⚙️ Lab3 Data Visualizer for Touch

- **Graph** window of Touch Dashboard.



* Vertical 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.

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

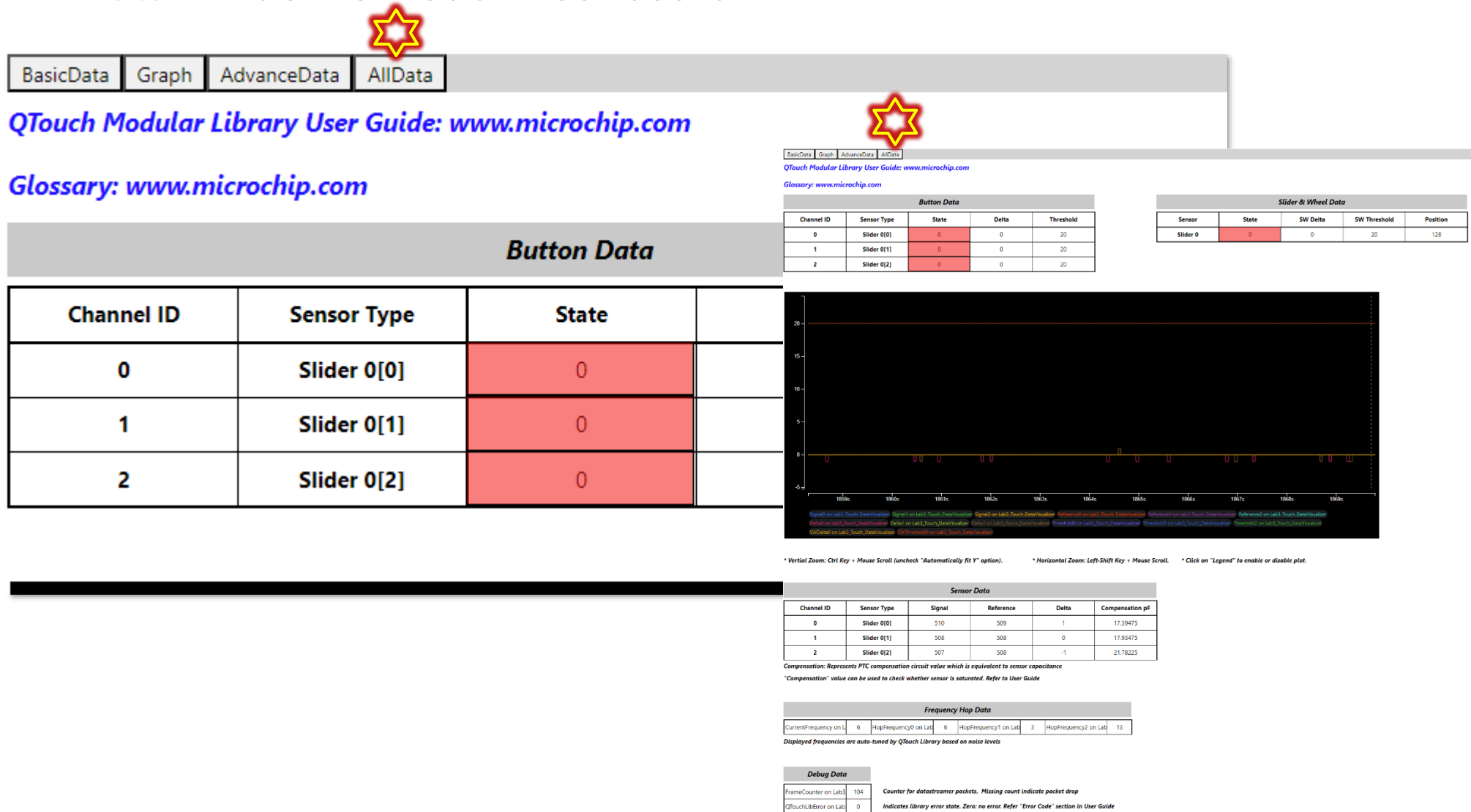
Frequency Hop Data

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

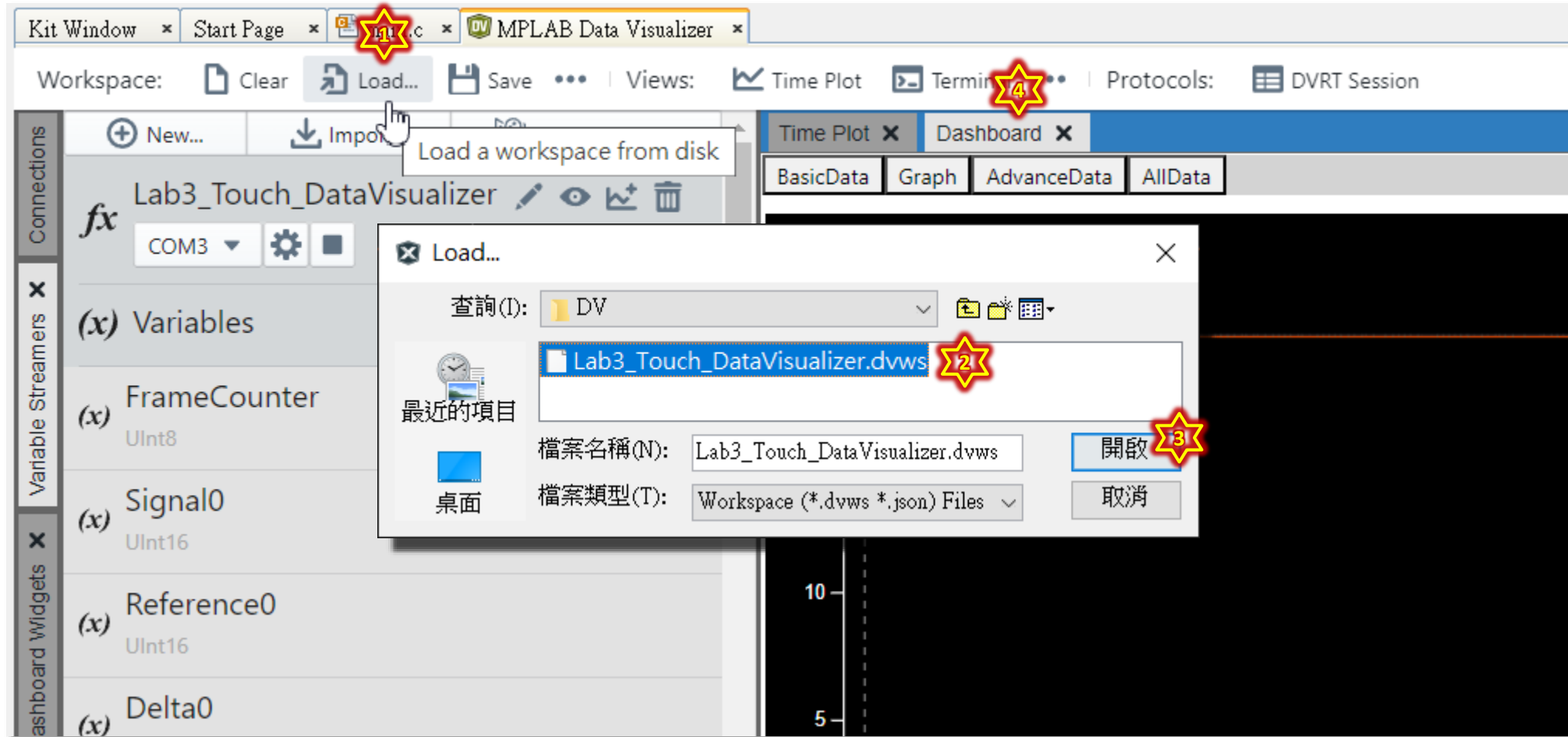
⚙️ Lab3 Data Visualizer for Touch

- All Data window of Touch Dashboard.



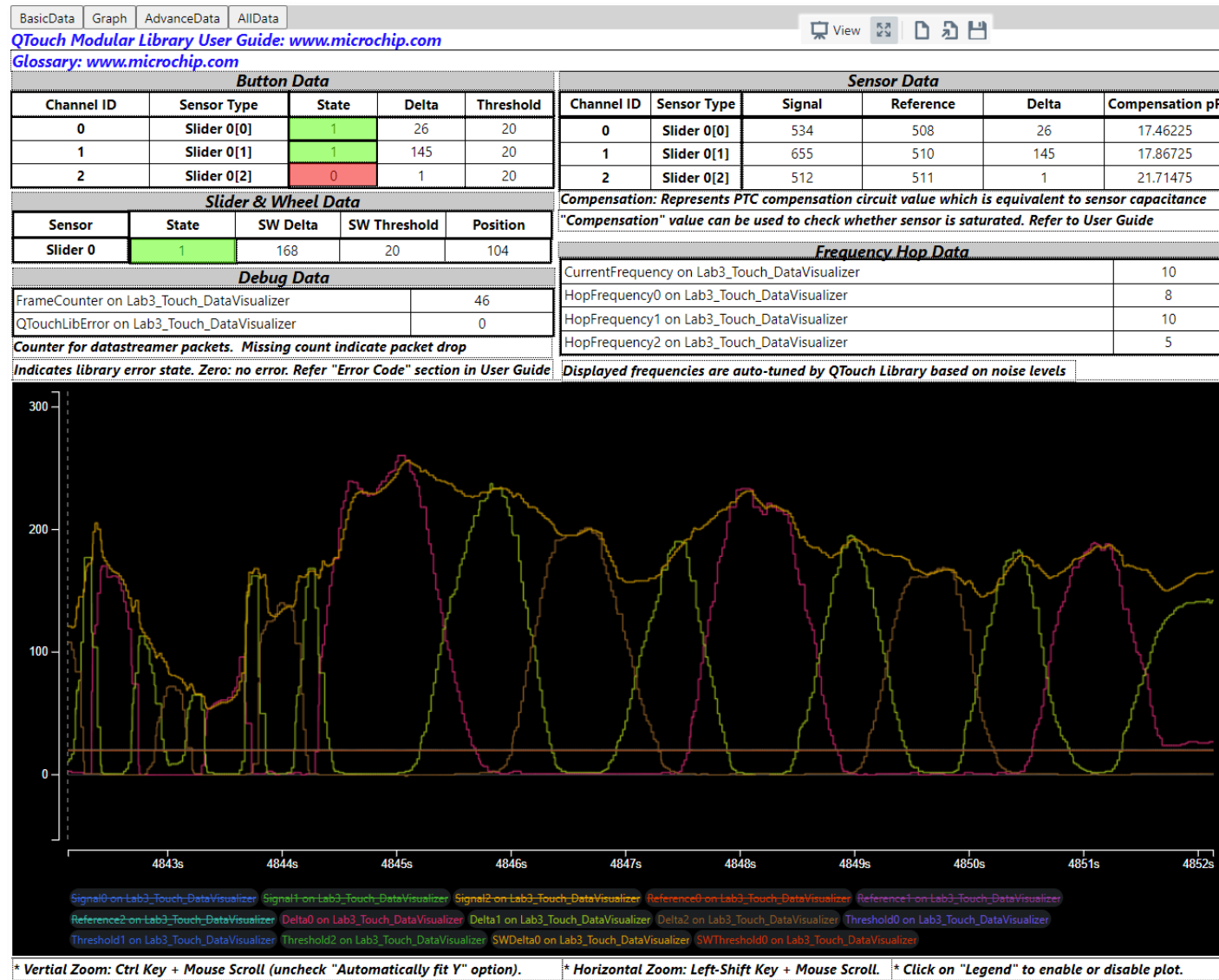
⚙️ Lab3 Data Visualizer for Touch

- Load a customized Dashboard **Lab3_Touch_DataVisualizer.dvws** from **\DV** folder 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, Klear, 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, SQL, 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

