Application Report
SLAA877–December 2018

Capacitive Touch Gesture Software and Tuning

Dennis Lehman *MSP430 System Applications*

ABSTRACT

Capacitive touch human-machine interfaces (HMIs) are becoming more common as a replacement for traditional mechanical button or keypad interfaces in products ranging from appliances to personal electronics . Capacitive touch interfaces enable products to be developed with sleeker designs, enhanced user experience, lower cost, and improved product reliability. Incorporating gesture software algorithms with a product's user interface can, for example, transform a simple capacitive touch wheel into a multi-function user interface with multiple tap zones and swipe or slide gestures for controlling various application functions or features. Choosing to incorporate gesture software algorithms with traditional capacitive touch sensors opens up new opportunities for product design flexibility and functionality.

Contents

List of Figures

Trademarks

CapTIvate, LaunchPad, Code Composer Studio are trademarks of Texas Instruments.
IAR Embedded Workbench is a registered trademark of IAR Systems.
Windows is a registered trademark of Microsoft Corporation.

1 Introduction

How a user interacts with the interface is driven by the software that creates a sensor's behavior. At a fundamental level, determining a finger touch or position is performed by the sensor's basic detection software or library driver. Tracking and processing how long a finger touches, how far the finger moves, and looking for one or more specific patterns requires a software algorithm that, in this context, is referred to as a gesture algorithm. The gesture software algorithms create new sensor interface behaviors described as tap, double tap, tap-hold, swipe, and slide.

This application note, along with the Gesture-Based Capacitive Touch Speaker Interface (TIDM-02004) reference design, provide a introduction intoduction into gesturing software, how it can be incorporated into new product designs, and demonstrate a real-world gesturing application that controls a Windows® Media Player using the CAPT-MSP-FR2633 CapTIvate™ Capacitive Touch development kit.

NOTE: References to the MSP430FR2633 MCU, MSP-CAPT-FR2633 development kit, CapTIvate Design Center, CapTIvate Technology Guide, and TIDM-02004 reference design are made throughout this document. Follow these links for more information related to these topics.

2 Gestures

2.1 Theory of Operation

To understand how gesture detection works, first look at how a gesture is detected. The MSP430FR2633 MCU with CapTIvate capacitive touch technology measures the change in capacitance on the CAPTIVATE-BSWP panel caused by a finger touch on a button, slider, or wheel sensor. The CapTIvate touch library includes fundamental algorithms that determine if a sensor is touched and the position of the finger on slider and wheel sensors.

The MSP430FR2633 has a dedicated 16-bit CapTIvate timer that generates a periodic capacitive touch measurement interrupt every 20 milliseconds, by default, and is user configurable in software. During each interrupt, the buttons, slider and wheel sensors are measured, followed by the gesture processing, which uses this periodic sampling rate as a time base measurement for finger touch and release events. Since the CapTIvate peripheral has it's own dedicated timer, no general-purpose 16-timers on the MCU are needed, leaving them available for the main application.

NOTE: Important Concept: Gesture timing is based on the sensor sampling rate.

2.2 Sensor Gesture Processing

There are two types of gesture attributes that define a gesture. The first attribute type is time. Time is expressed in sample counts and is measured by counting the number of sensor measurement samples between two events, such as a finger touch followed by release. For example, when a sensor is sampled every 20 milliseconds, a touch that lasts 10 sample periods represents a touch for 200 milliseconds. The time attribute applies to buttons, sliders, and wheels. A gesture can use one or more timing attributes to define parameters such as "sample count min" or "sample count max". A sample count min parameter = 5 means there is a minimum of 5 sample counts required.

The second attribute type refers to distance. Distance is expressed in points of resolution and measured between two reported finger positions on the slider and wheel sensor. In addition to the time attribute, a slider or wheel gesture can use one or more distance attributes to define parameters such as "swipe distance min" or "slide step size min". A swipe distance min parameter = 50 means the finger must move a minimum distance of 50 points. The number of resolution points chosen for a distance parameter depends on the resolution of the slider or wheel sensor. In the example software, the slider has a 1000-pt resolution and the wheel has a 100-pt resolution. So if a slider is configured with 1000-pts of resolution, a finger that moves 10% of the slider length moves 100-pts.

Why is it important to use parameters to set rules for gestures? Because a finger gesture duration and speed can vary from user to user and using parameters help improve gesture behavior repeatability and detection accuracy. The slider, wheel and button sensor types have their own gesture parameters and are configurable in software. Because each sensor type can support different gesture behaviors, processing is specific to a sensor. This is why each sensor has its own gesture state machine and gesture parameters. The sensor gesture software function is essentially a state machine that is executed after each sensor measurement as part of the sensor's callback function.

2.3 Gesture Software Overview

The gesture software is simply a collection of three sensor specific functions providing sensor specific gestures for the TIDM-02004 reference design demonstration. Examine the example demo source code in the TIDM-02004 reference design for specific sensor implementation. Sensor state machine diagrams are provided in the Gesture Software OverviewSection 2.3 section. Refer to the TIDM-02004 reference design for details on the gesture timing and motion diagrams.

2.3.1 Slider Gesture

The processSliderGesture() function, located in file media_player_slider.c, is called from the slider sensor handler and operates as a state machine performing the gesture processing after every slider sensor measurement. This function uses the timing and motion parameters from the SliderGestureParams data structure:

```
tSensorGestureParams SliderGestureParams = {
    .ui16TouchSampleCount_Min = 3,
    .ui16TouchSampleCount_Max = 16,
    .ui16SwipeSampleCount_Max = 10,
    .ui16DoubleTapDelayCount_Max = 10,
    .ui16FingerDistance_Min = 10,
    .ui16SlideStepSize_Min = 25,
    .ui16SwipeDistance_Min = 50,
};
```

Figure 1 shows the state machine states and actions.

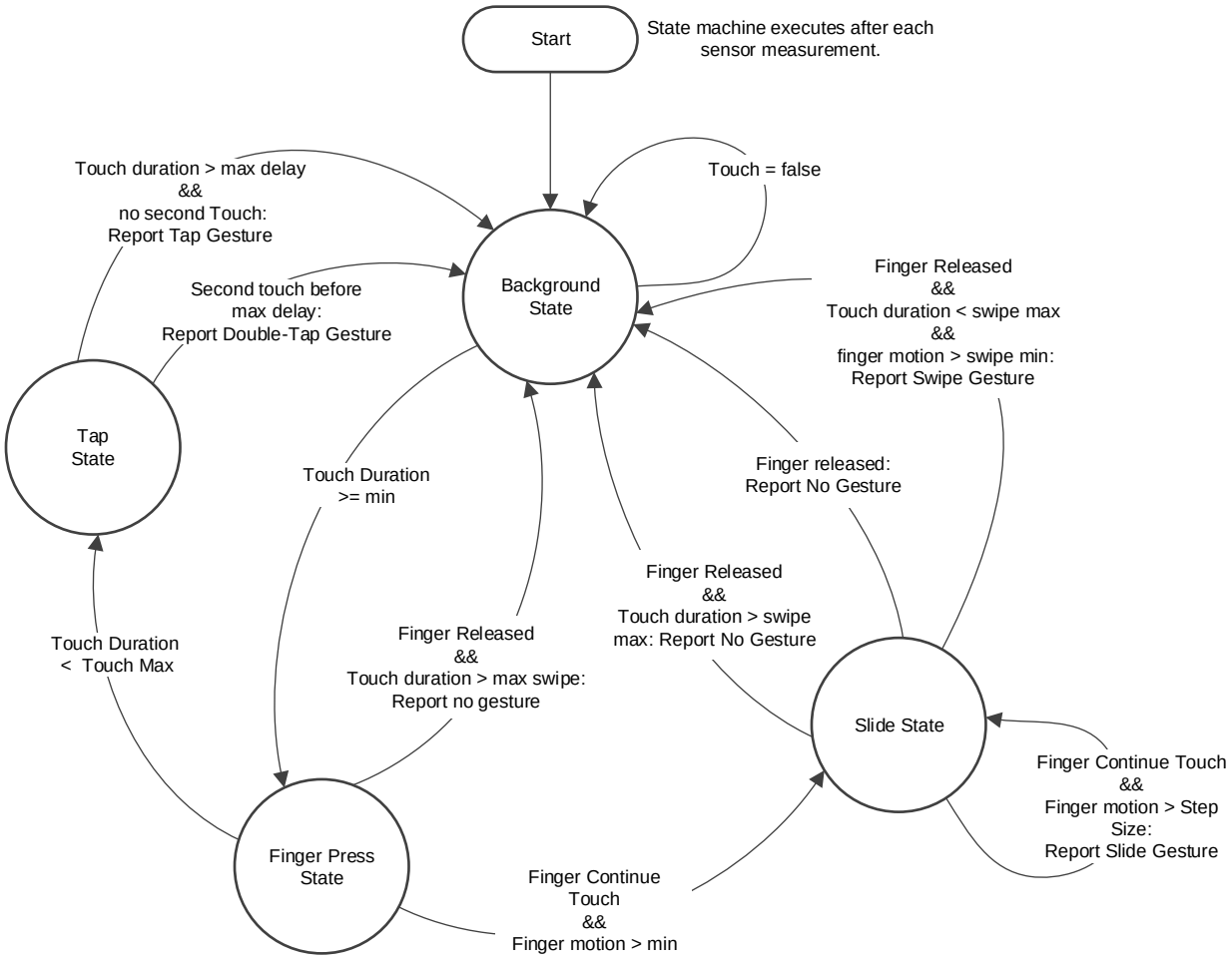

Figure 1. Slider Sensor State Diagram

2.3.2 Wheel Gesture

The processWheelGesture() function, located in file media_player_wheel.c, is called from the wheel sensor handler and operates as a state machine performing the gesture processing after every wheel sensor measurement. This function uses the timing and motion parameters from the SliderGestureParams data structure:

```
tSensorGestureParams WheelGestureParams =
{
    .ui16TouchSampleCount_Min = 3,
    .ui16TouchSampleCount_Max = 16,
    .ui16SwipeSampleCount_Max = 10,
    .ui16FingerDistance_Min = 3,
    .ui16SwipeDistance_Min = 10,
    .ui16SlideStepSize_Min = 5,
};
```

Figure 2 shows the state machine states and actions.

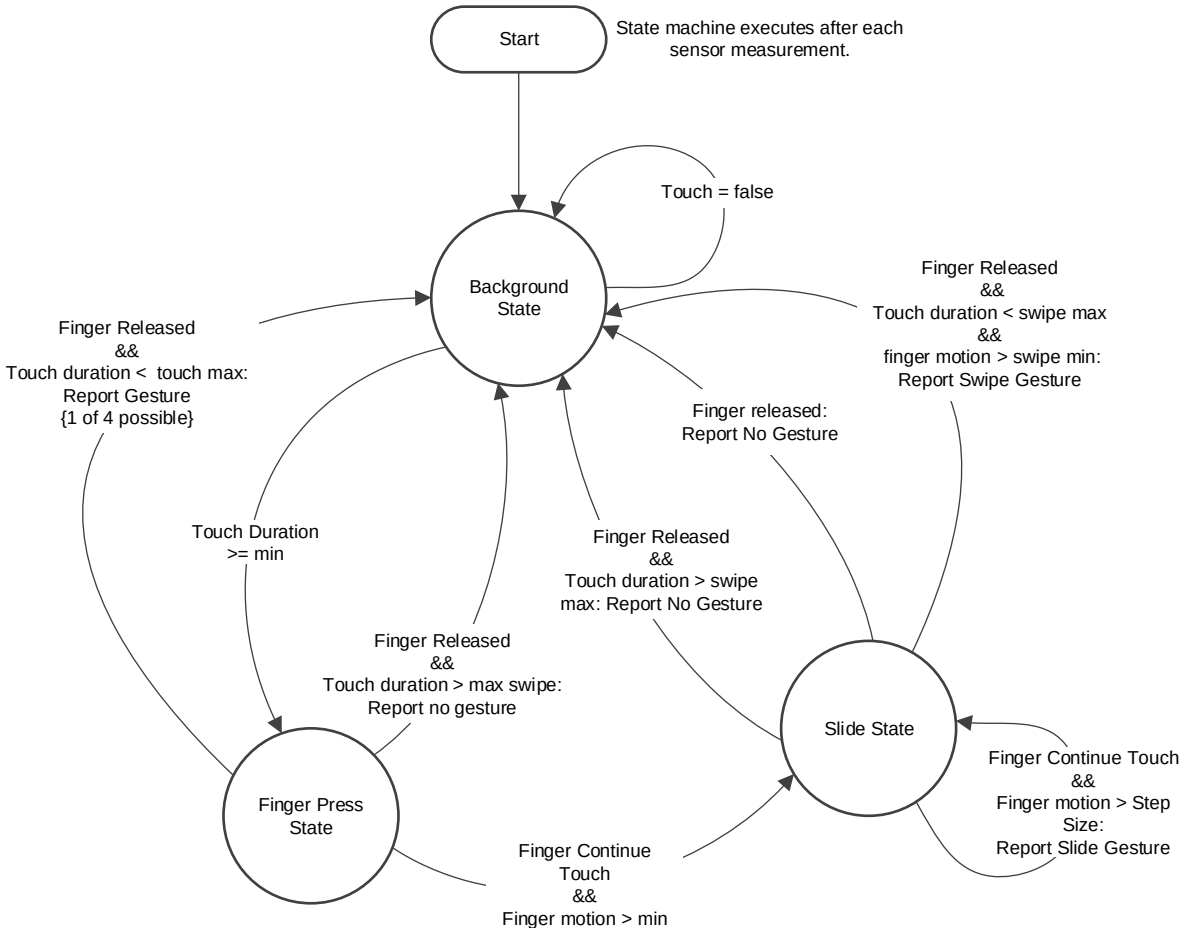

Figure 2. Wheel Sensor State Diagram

2.3.3 Button Gesture

The processButtonGroupGesture() function, located in media_player_buttons.c, is called from the button sensor handler and operates as a state machine performing the gesture processing after every button group sensor measurement. This function uses the timing and motion parameters from the SliderGestureParams data structure:

```
tSensorGestureParams ButtonGroupGestureParams = {
     .ui16TouchSampleCount_Min = 3,
     .ui16TouchHoldSampleCount_Min = 8,
};
```

Figure 3 shows the state machine states and actions.

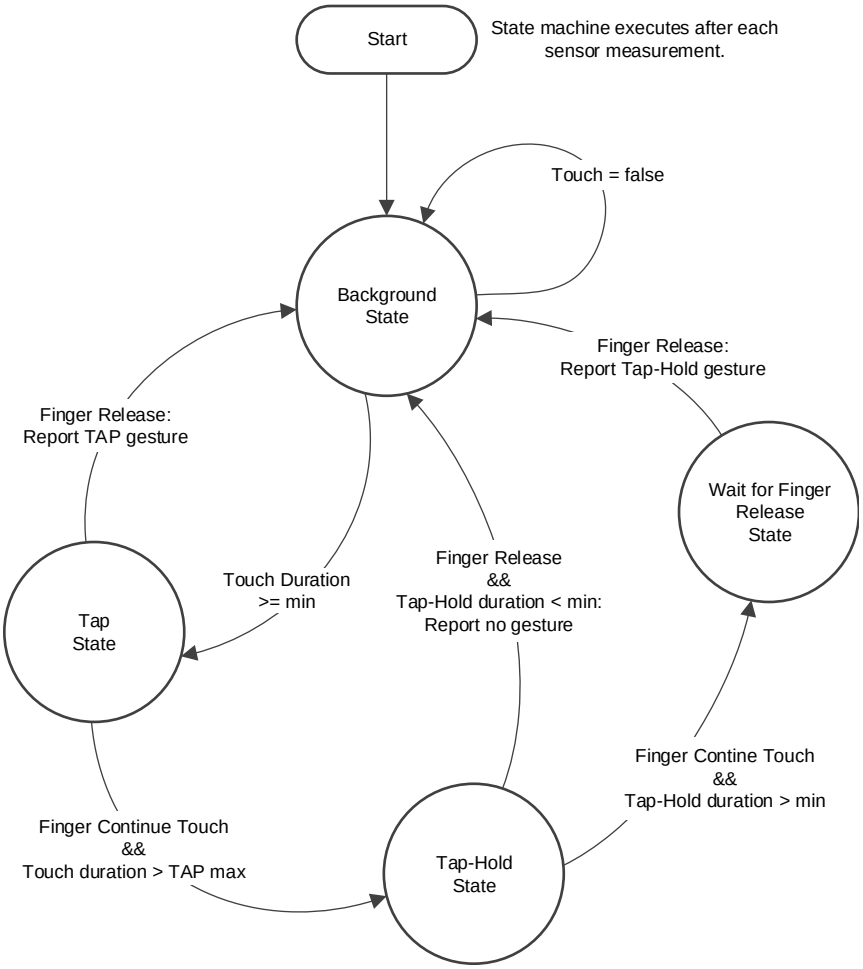

Figure 3. Button Sensor State Diagram

2.3.4 Sensor Handlers

The example code includes three sensor handlers, one for each supported type of sensor. The handlers are placed in file media_player.c. The sensor handlers emulate the F7, F8, F9 CTRL-P, CTRL-B and CTRL-F keys on a keyboard that control the various Windows media player functions. These keys are mapped to specific gestures from each of the three sensors. When a valid gesture is detected, a lookup table provides the corresponding key value and is reported to the USB HID device. In a normal software design, the handler could incorporate the gesture algorithm within the same function, but in this demo, the handler and gesture functions are separate on purpose for code clarity.

The sensor gesture state diagrams in Section 2.3 provide an overview of the gesture functionality. Refer to the TIDM-02004 example demo source code for specific details regarding the implementation.

2.3.5 How to Add Gestures to the CapTIvate Framework

The TIDM-02004 example demo software is only one example of how gestures can be implemented. You are encouraged to modify these examples as needed for your application.

When considering incorporating gesture software into your project, remember that each function is written to support specific gestures on the button, slider, and wheel sensors. Some gestures may be common across all three sensors, while others are specific to the slider and wheel sensor. You need to modify the gesture software and sensor parameters to achieve the specific gesture behavior for your application.

For example, the sensor timing parameters are based on a default 20-ms sensor sample rate. A different sensor sampling rate requires modifying all of the sensors timing parameters from their default settings. For example, if the sampling rate increases from 20 ms to 10 ms, all timing parameters must double to maintain the same timing and responsiveness.

Likewise, should the slider or wheel resolution change from their defaults, the motion limits and step sizes need to be changed accordingly. The slider motion parameters are based on 1000-pt resolution and wheel motion parameters are based on 100-pt resolution.

During the capacitive touch sensor design process, the CapTIvate Design Center (CDC) is used to create the initial sensor configuration and baseline source code project. Incorporating the gesture algorithm into the generated source code project is simple, requiring only a callback function (a sensor handler) and a call to register the sensor handler in the MCU startup code.

The callback or sensor handler is called after a sensor has been measured and the fundamental touch processing has been completed. This is typically where any additional processing, such as communication, toggling an LED, turning on or off a motor control or, in this example, the sensor's gesture algorithms are performed. This does not happen automatically. The application must first register a callback for each sensor. The following example shows the wheel sensor callback registration from the example code. This occurs in the Demo_Init() function in the file media_player.c.

```
MAP_CAPT_registerCallback(&wheelSensor, &wheelSensorHandler);
```

For more information about the callbacks, refer to the Software Library chapter of the CapTIvate Technology Guide.

In summary, all that is needed to incorporate gesturing into a CapTIvate project is create a sensor specific gesture algorithm as the sensor's handler and register it as the sensor's callback function.

3 Tuning

To achieve a good user experience with gesture-based capacitive touch interfaces, a gesture's responsiveness should be adjusted to meet the application's user interface requirements. This is an iterative process requiring the adjustment of the gesture timing and distance parameters. You are encouraged to experiment with the timing and motion parameters to understand the relationship between the settings and the sensor's responsiveness.

3.1 CapTIvate Design Center

To demonstrate the gesture tuning process, the CapTIvate Design Center (CDC) and TIDM-02004-BSWP-Demo.ser project is used. The MSP430FR2633 must be programmed with the TIDM-02004-Gesture-Demo firmware. The CapTIvate Design Center is available for download if not already installed. The CapTIvate Design Center was used to create the sensor configuration and application framework for the CAPT-BSWP panel. The resulting CapTIvate Design Center project, TIDM-02004-BSWP-Demo.ser, is located in the TIDM-02004 software download directory "DesignCenterFiles".

Connect the MSP430FR2633 MCU and CAPT-PGMR programmer with a USB cable to a PC. Launch the CapTIvate Design Center and open the TIDM-02004-BSWP-Demo CDC project. It should appear as shown in Figure 4. Double click on the user data log icon to open the oscilloscope plot view. Click the box labeled "Connected" to start communications with the CapTIvate Design Center.

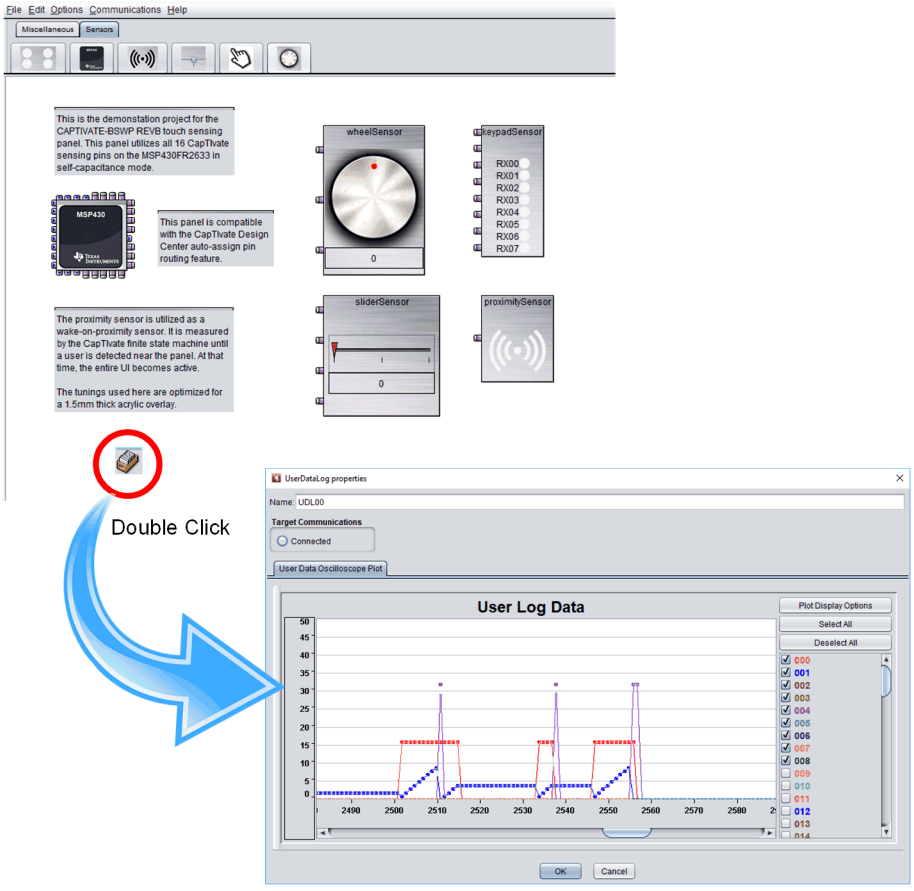

Figure 4. CapTIvate Design Center Project

3.1.1 Enabling Gesture Sensor Output

To simplify the gesture tuning process, the CapTIvate Design Center can be used to display the gesture status and output using the User Data Oscilloscope Plot view. This is possible only when the demonstration software function sendGestureStatus_GUI() in the file media_player.c is enabled.

By default, this function is not enabled. To enable it, modify the following line of code in the file media_player.h.

```
#define SEND_GESTURE_DATA (true)
```

This function uses the CapTIvate library function CAPT_writeGeneralPurposeData() to send user-defined data to the GUI where it is displayed in the User Data Oscilloscope Plot view. The sensor's gesture data is transmitted to the GUI using the available UART on the MSP430FR2633 MCU board. By default, the UART is disabled. To enable the UART modify the following line of code in the file CAPT_UserConfig.h.

```
#define CAPT_INTERFACE  (__CAPT_UART_INTERFACE__)
```

Each sensor callback by default will call the sendGestureStatus_GUI() when it is enabled. This creates a problem because all three sensors will send their data to the GUI and the information that is displayed gets confusing. To prevent the other two sensors from sending gesture data to the GUI, temporarily disable themby changing the #define in file gesture_definitions.h from true to false. The following code example shows how to enable just the slider sensor.

```
#define USE_BUTTONGROUP (false)
#define USE_SLIDER (true)
#define USE_WHEEL (false)
```

Because the demonstration software operates with wake on proximity mode enabled by default, the MCU is in a low-power mode when the CAPT-BSWP panel is not touched and no data is streamed to the CapTIvate Design Center. Touching any of the sensors wakes the MCU and data begins to stream. If this becomes inconvenient while tuning the sensor gestures, temporarily disable the wake on proximity feature and allow the data to stream continuously by modifying the following code example in file CAPT_UserConfig.h:.

```
#define CAPT_WAKEONPROX_ENABLE (false)
```

After all the required modifications, build and reprogram the MSP430FR2633 target MCU.

3.1.2 CDC Plot Channel Assignments

When the gesture information is streamed to the oscilloscope plot view from the MSP430FR2633 MCU, there are 9 channels, each containing a 16-bit value. To view only the ones of interest, click a data channel in the oscilloscope view to select or deselect it. The channels and assigned data values are:

Ch 0 = SensorTouch (this indicates when sensor is touched)

Ch 1 = SampleCount (current value of sample counter)

Ch 2 = SingleTap (this indicates a tap gesture)

Ch 3 = DoubleTap (this indicates a double-tap gesture)

Ch 4 = SlideLeft (this indicates a slide left motion gesture)

Ch 5 = SlideRight (this indicates a slide right motion gesture)

Ch 6 = SwipeLeft (this indicates a swipe left motion gesture)

Ch 7 = SwipeRight (this indicates a swipe right motion gesture)

Ch 8 = TapHold (this indicates a tap-and-hold gesture)

3.2 Tuning Process

The default sensor sample rate, sensor timing and motion parameters in the example gesture software have been selected to provide good sensor response with the CAPT-BSWP board. Each sensor's gesture time and motion parameters are configurable and can be tuned for different gesture responsiveness. You are encouraged to experiment with the sensor gesture parameters and sample rate settings.

To change the sensor sampling rate, modify the g_uiApp data structure element found at the very end of the file CAPT_UserConfig.c as shown here. Keep in mind this file will be overwritten if updated from the CapTIvate Design Center.

```
.ui16ActiveModeScanPeriod = 10,
```

Build and reprogram the MSP430FR2633 target MCU.

To see how the gesture tuning process works,perform all the steps above to enable streaming the slider gesture data to the CapTIvate Design Center. While data is being displayed, perform a tap on the slider sensor. The sensor's sample count, touch status and gesture output should appear in the oscilloscope view. Make sure the appropriate channels for the Tap gesture are enabled. Experiment with fast and slow finger taps. If your tap gesture meets the min and max timing parameters your plot should appear similar to the Valid Tap example shown in Figure 5.

If your finger tap is slow, the gesture may not be detected because the finger was touching too long and the ui16TouchSampleCount_Max parameter was exceeded. In this case your plot will look similar to the Invalid Tap example shown inFigure 7 . Try the tap again but this time the finger should touch only briefly.

If you finger tap is too quick, the gesture may not be detected because the finger was not touching long enough and did not meet the ui16TouchSampleCount_Min parameter. The plot should appear similar to the Invalid Tap example shown in Figure 6 . If you desire a quicker response, modify the ui16TouchSampleCount default value from 3 to 2, or possibly 1. With the sample rate at 20msec, going from 3 to 1 is effectively decreasing the time from 60msec to 20msec. Build and reprogram the MSP430FR2633 target MCU. Test again to see if the gesture detects the tap reliably.

Experiment with the swipe and slide gesture to see how the time and distance parameters affect the sensor's responsiveness. Adjust accordingly to achieve your desired response.

3.2.1 Tap

The tap gesture is controlled by two timing parameters: ui16TouchSampleCount_Min and ui16TouchSampleCount_Max. The data channels used for Tap tuning are (000) finger touch, (001) sample count and (002) Tap gesture detect.

Figure 5 shows a valid tap gesture. When the finger touches, the sample count resets to zero then is incremented after each sensor measurement. The finger touches for 10 samples, meeting both the minimum and maximum timing parameters. A valid tap gesture is reported.

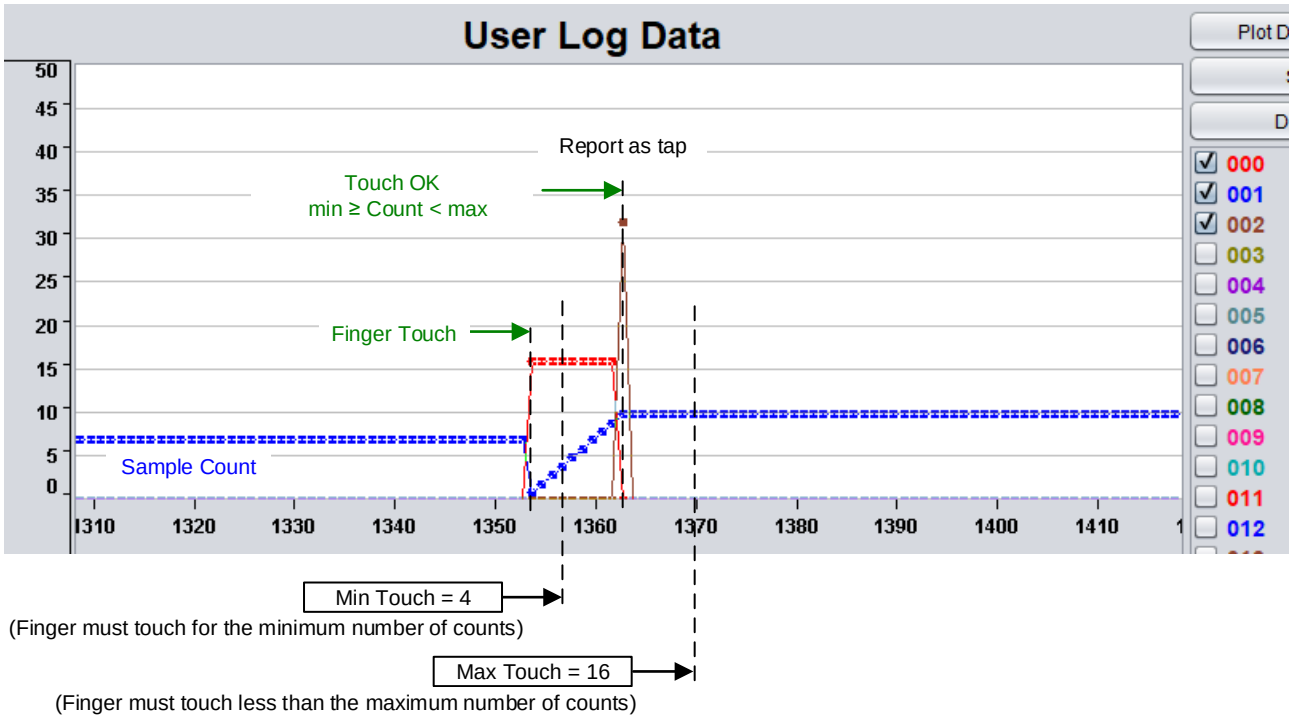

Figure 5. Valid Tap

Figure 6 shows an invalid tap gesture caused by a brief touch. In this capture the finger touches for only two samples. Because this does not meet the minimum timing parameter, no gesture is reported.

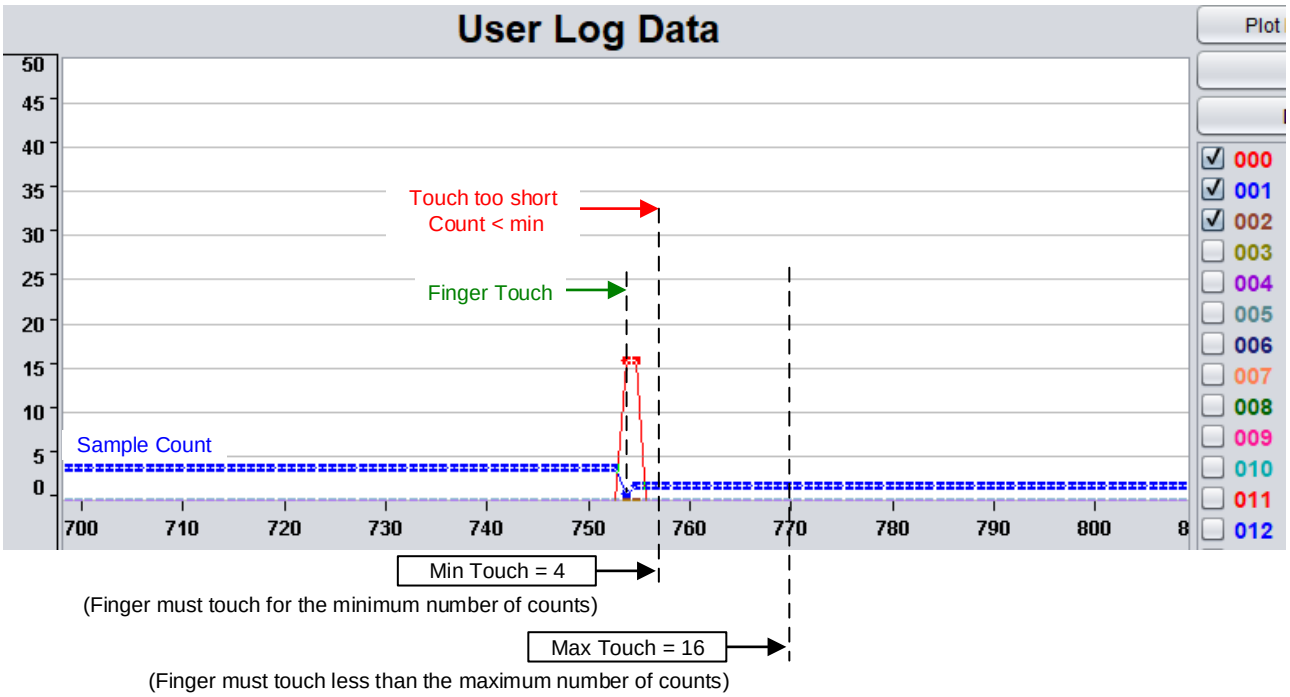

Figure 6. Invalid Tap: Touch Too Short

Figure 7 shows an invalid tap gesture caused by the finger touching too long. Because the maximum timing parameter is exceeded, no gesture is reported.

NOTE: This behavior is different if double tap is supported (see Section 3.2.2).

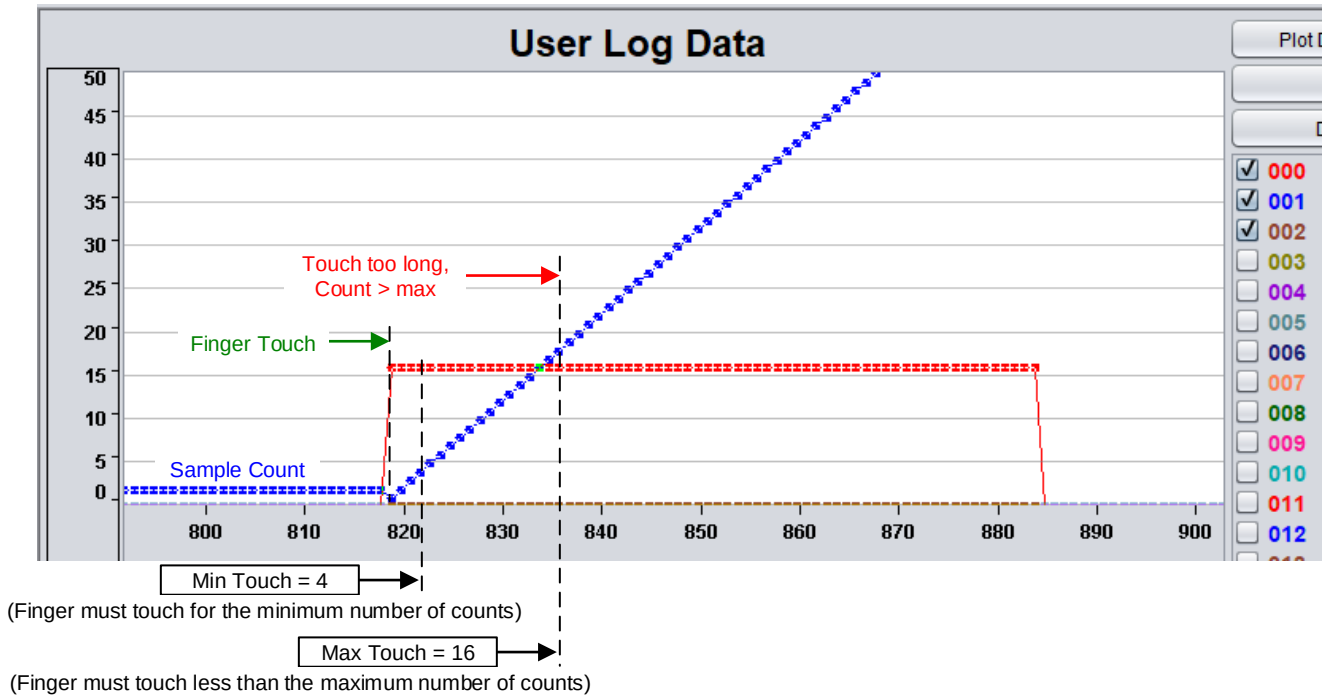

Figure 7. Invalid Tap: Touch Too Long

3.2.2 Double Tap

The double-tap gesture is controlled by two timing parameters: ui16TouchSampleCount_Min and ui16DoubleTapDelayCount_Max. The data channels used for tap tuning are (000) finger touch, (001) sample count, (002) tap gesture detect, and (003) double-tap gesture detect.

Figure 8 show a valid double tap gesture. The first finger touch and release meets the minimum timing parameter, followed by a second finger touch. The maximum delay allowed between finger release and second touch was not exceeded, so a valid gesture is reported.

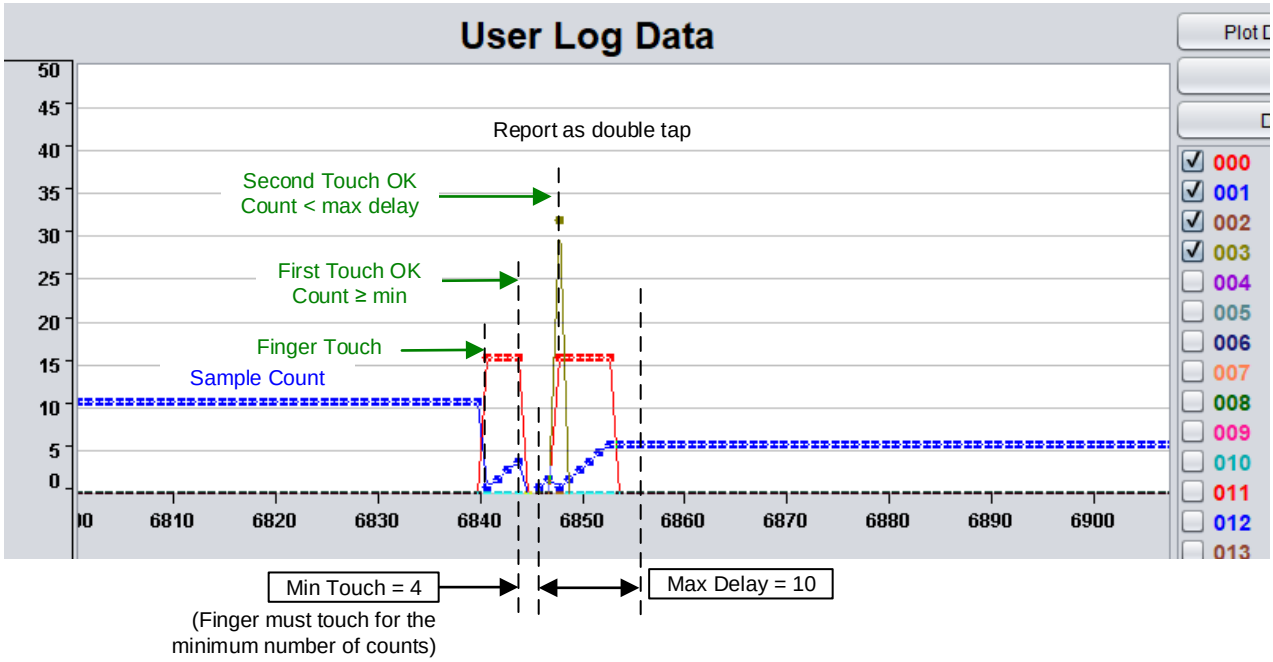

Figure 8. Valid Double Tap

Figure 9 shows an invalid double tap gesture that appears as only a single tap. The first finger touch and release meets the minimum timing parameter; however, the second touch never occurs. Because this is the pattern for a single tap event, this gesture is reported as a tap gesture. The tap report is delayed until the maximum timing parameter is exceeded.

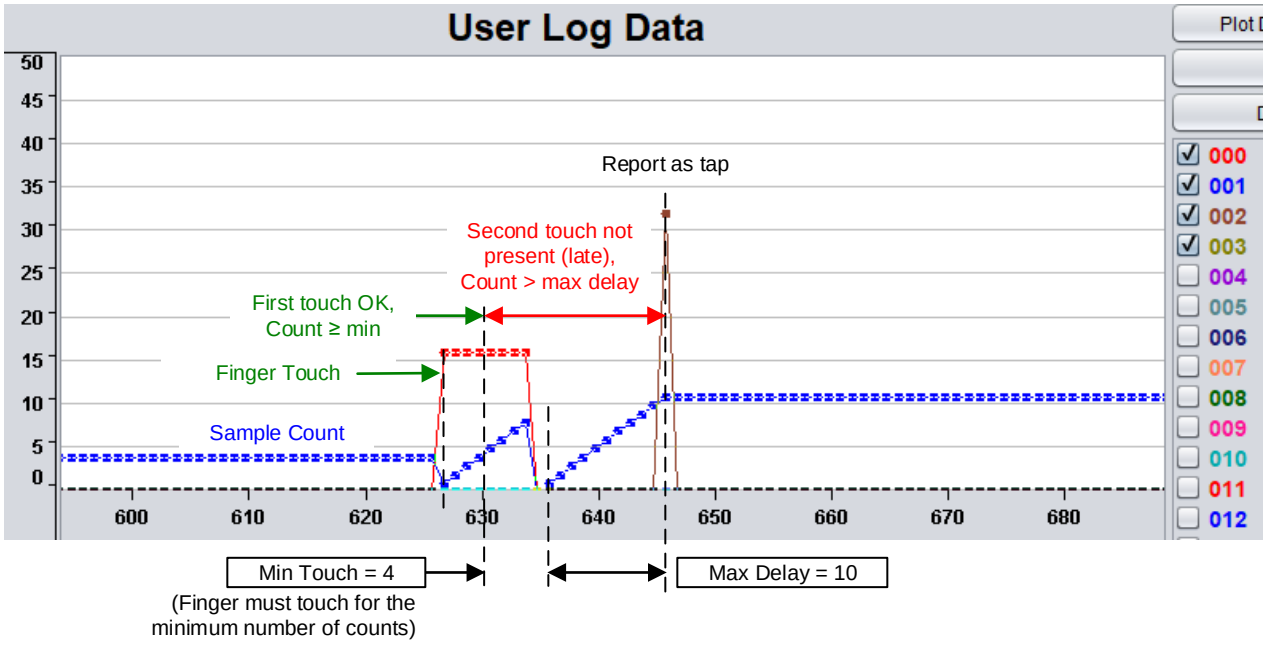

Figure 9. Invalid Double Tap: Missing Second Touch

Figure 10 shows an invalid double tap gesture that appears as two single taps close together. The first finger touch and release meets the minimum timing parameter; however, the second touch occurs after the maximum timing parameter, so the first touch is reported as a single tap gesture. The second finger touch also meet the minimum timing parameter, but because no additional finger touches occur, the maximum timing parameter is violated again and the touch is reported as a single tap gesture.

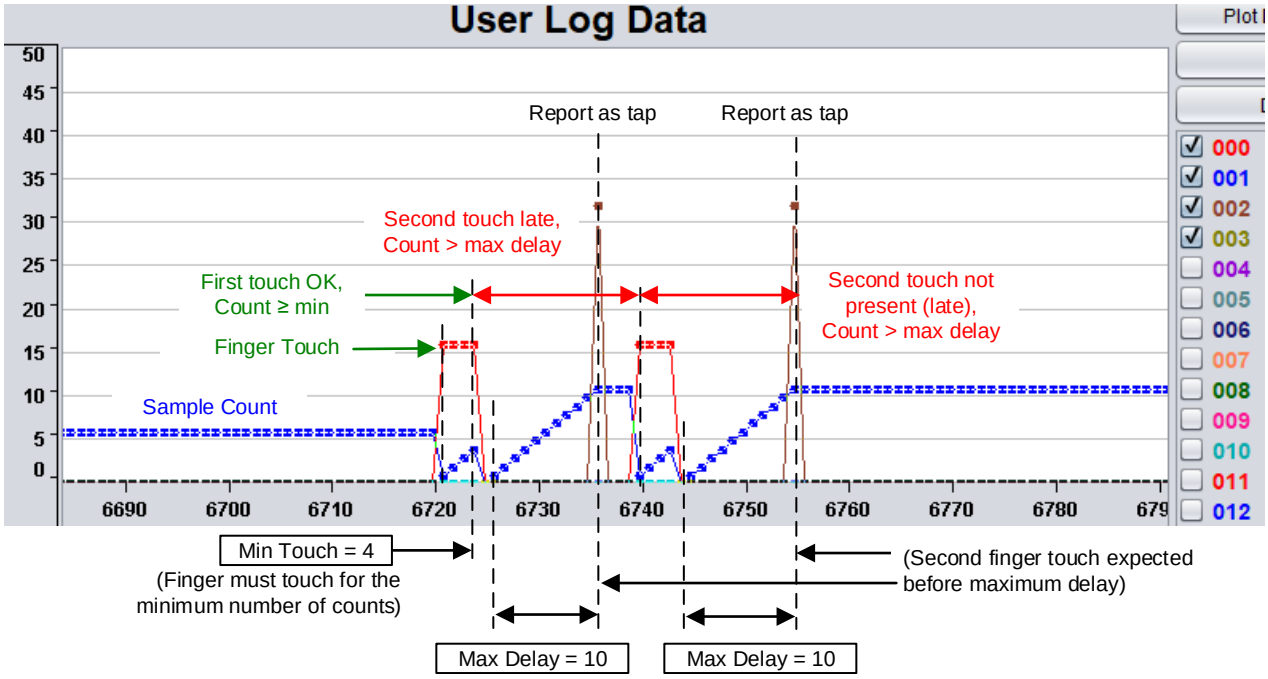

Figure 10. Invalid Double Tap: Time Between Taps Too Long

3.2.3 Swipe

The swipe gesture is controlled by one timing and two distance parameters: ui16SwipeSampleCount_Max, ui16FingerDistance_Min, and ui16SwipeDistance_Min. The data channels used for swipe tuning are (000) finger touch, (001) sample count, (006) swipe left gesture detect, and (007) swipe right gesture detect.

Figure 11 shows a valid finger swipe right gesture. In this capture, the finger touches and moves along the slider, meeting the minimum finger distance parameter. As the finger continues its motion, it moves far enough to meet the minimum swipe distance parameter, then the finger releases before the maximum timing parameter is exceeded. A swipe left or swipe right gesture is reported, depending on the direction of the motion. This gesture can be used with slider and wheel sensors.

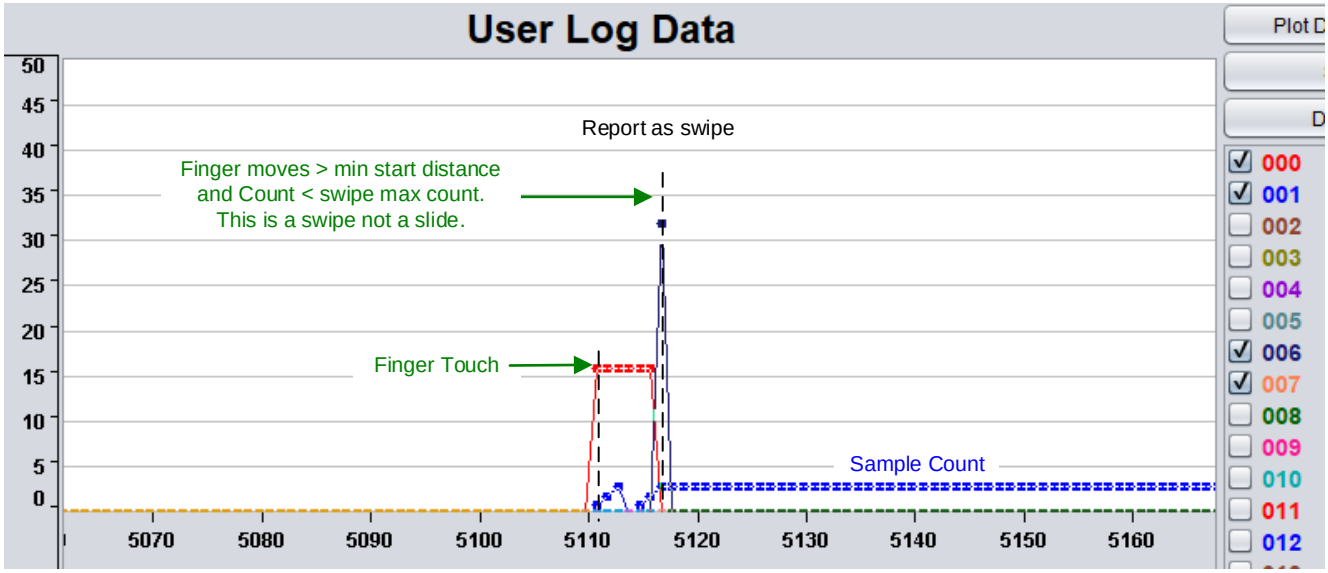

Figure 11. Valid Swipe

3.2.4 Slide

The slide gesture is controlled by two distance parameters: ui16FingerDistance_Min and ui16SlideStepSize_Min. The data channels used for slide tuning are (000) finger touch, (001) sample count, (004) slide left gesture detect, and (005) slide right gesture detect.

Figure 12 shows a valid finger slide left gesture. In this capture the finger touches and moves along the slider, meeting the minimum finger distance parameter. As the finger continues to move, each time the minimum step size parameter is met, a slide left or slide right gesture is reported, depending on the direction of the motion. This gesture can be used with slider and wheel sensors.

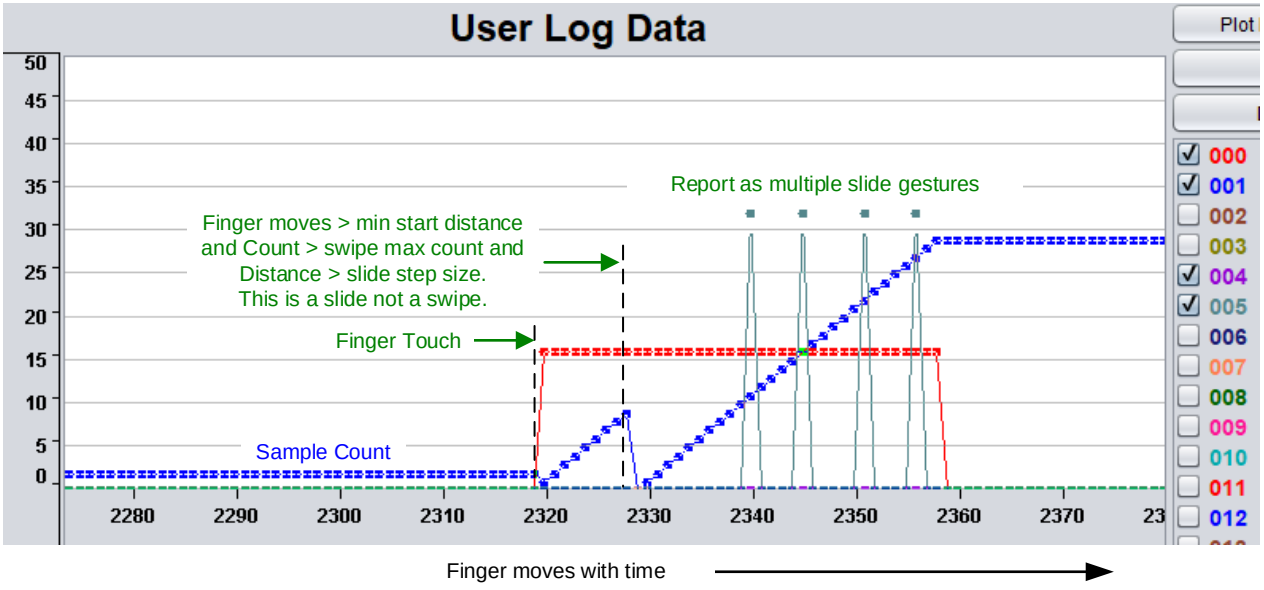

Figure 12. Valid Slide

Figure 13 shows a comparison of a swipe gesture and a slider gesture.

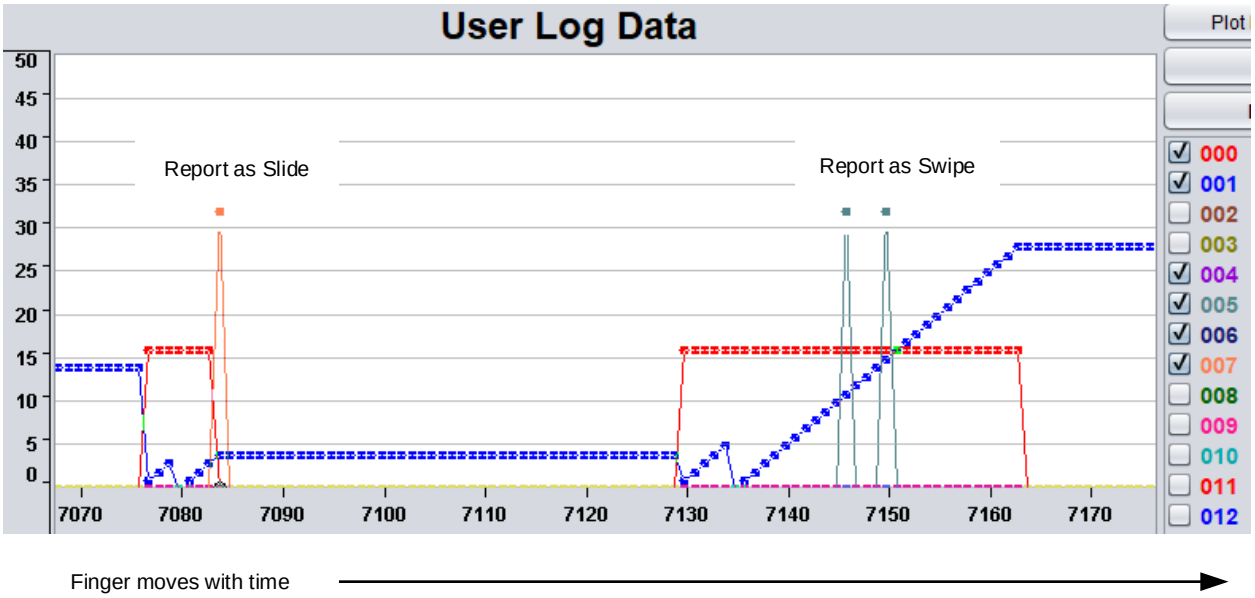

Figure 13. Comparison of Slide and Swipe

4 Example Demonstration Software Installation

The software that is described in this document is included in TIDM-02004_CapTIvate-Gesture_Demo_SW.zip, which available from the TIDM-02004 tool page. The zip file contains the source code for the MSP430FR2633 MCU board and the MSP430F5529 LaunchPad™ development kit. The TIDM-02004 reference design demo software package was created in the TI Code Composer Studio™ IDE version 8.2.0.00007 and the TI MSP430 compiler v18.1.4.LTS and is referenced throughout this document. The project code can be built using either Code Composer Studio IDE or IAR Embedded Workbench® IDE.

4.1 Project Directories

Two example projects are included (see Figure 14). This document focuses on the MSP430FR2633 gesturing software.

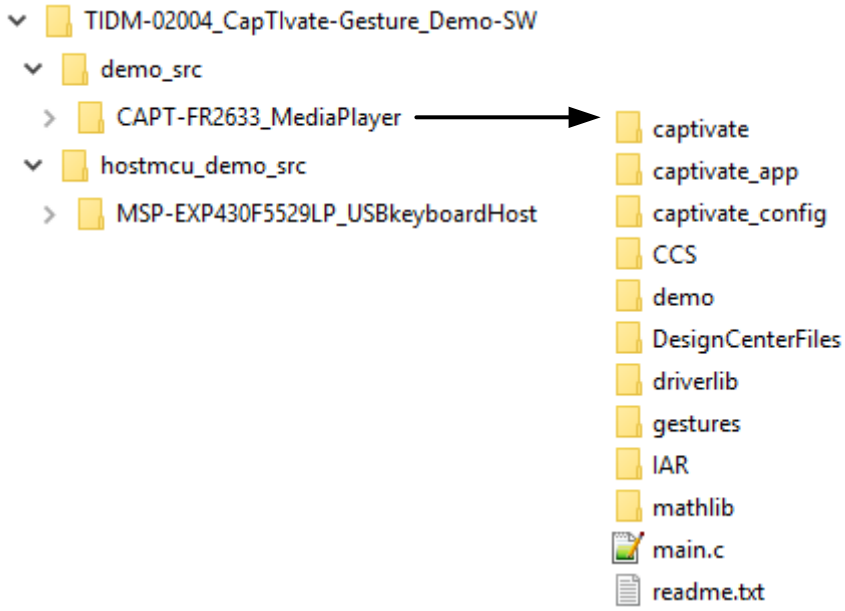

Figure 14. TIDM-02004 Software Directory Structure

4.2 Import the Project Into CCS

From the top menu in CCS select Project → Import CCS projects. When the dialog box appears, navigate to and click on the directory TIDM-02004_CapTIvate_Gesture_Demo, then click OK. Under discovered projects, select CAPT-FR2633-MediaPlayer.

The gesture demonstration code is built on top of the CAPT-BSWP demo project that comes with the CapTIvate Design Center installation. Figure 15 shows the files related to this demo in green.

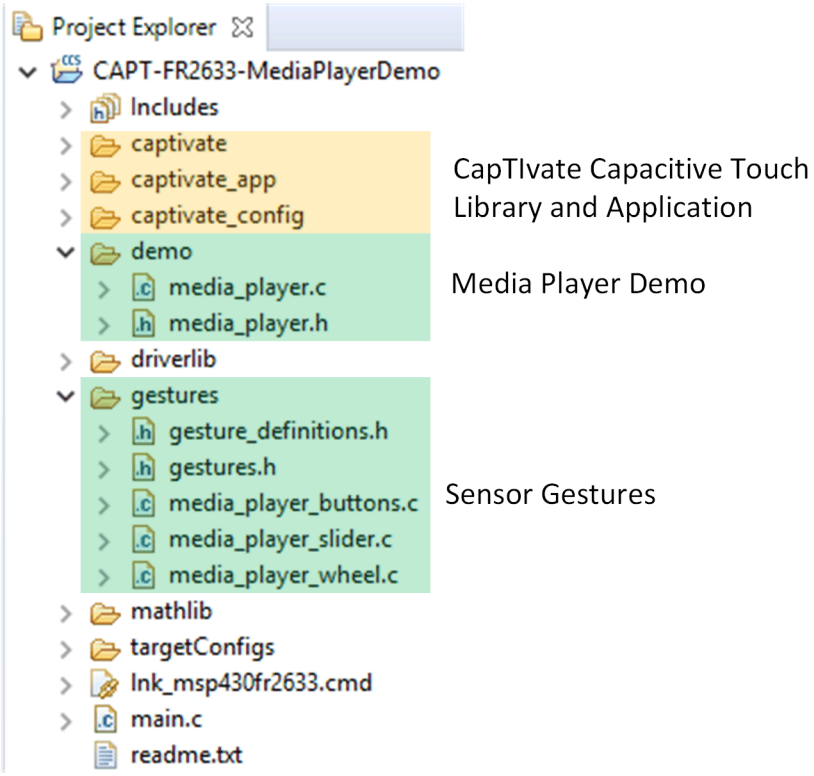

Figure 15. Project View in Code Composer Studio IDE

The CapTIvate application framework and library code are generated by the CapTIvate Design Center (CDC) based on the CAPT-BSWP hardware design. The application code performs the periodic sensor measurements and any communication with the CDC. For more information about the CapTIvate software, refer to the Software Library chapter of the CapTIvate Technology Guide and to the Exploring the CapTIvate Touch Library workshop chapter.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265