



Eason Zhou, Wiky Liao, Yashraj Motwani, Divyansh Mittal

## Abstract

This application note describes the implementation of a 4-wire resistive touch screen using the MSPM0L1306 microcontroller, through GPIO and ADC interfaces for real time touch detection on a resistive touch screen. The implementation provides two examples, a touch detection example which only confirms the touch present and sends the coordinate data, and the click and drag detection example which distinguishes between a click event and a drag or long press event and sends the coordinate data accordingly. This document details the hardware setup, pin configuration strategy, firmware implementation, and provides a detailed set of steps for developers integrating resistive touch functionality into embedded systems using the MSPM0L1306 platform.

## Table of Contents

|                                        |           |
|----------------------------------------|-----------|
| <b>1 Introduction</b>                  | <b>2</b>  |
| <b>2 LaunchPad Setup</b>               | <b>3</b>  |
| <b>3 Hardware Connection</b>           | <b>4</b>  |
| <b>4 Design Considerations</b>         | <b>6</b>  |
| 4.1 IO Selection Consideration         | 6         |
| 4.2 Measurement Method                 | 7         |
| <b>5 Software Structure</b>            | <b>9</b>  |
| 5.1 Touch Detection Example            | 9         |
| 5.2 Resistive Click and Drag Detection | 9         |
| 5.3 Pin Mode Switching                 | 10        |
| <b>6 Steps for Execution</b>           | <b>10</b> |
| <b>7 Results</b>                       | <b>11</b> |
| 7.1 Touch Detection                    | 11        |
| 7.2 Click and Drag Detection           | 12        |
| <b>8 Summary</b>                       | <b>13</b> |
| <b>9 References</b>                    | <b>13</b> |
| <b>10 Revision History</b>             | <b>14</b> |

## List of Figures

|                                                |    |
|------------------------------------------------|----|
| Figure 2-1. Removal of Launchpad Jumpers       | 3  |
| Figure 3-1. System Block Diagram               | 4  |
| Figure 3-2. Hardware Setup                     | 5  |
| Figure 4-1. Touch Detection Setup              | 7  |
| Figure 4-2. X-axis ADC Measurement             | 8  |
| Figure 4-3. Y-axis ADC Measurement             | 8  |
| Figure 5-1. Software Project                   | 9  |
| Figure 7-1. Touch Detection: 12-bit Resolution | 11 |
| Figure 7-2. Output for a Click                 | 12 |
| Figure 7-3. Output for Long Press              | 12 |

## List of Tables

|                                     |   |
|-------------------------------------|---|
| Table 2-1. LaunchPad Pin Connection | 3 |
| Table 4-1. Absolute Maximum Ratings | 6 |

## Trademarks

Code Composer Studio™ is a trademark of Texas Instruments.  
All trademarks are the property of their respective owners.

## 1 Introduction

Resistive touch screens are widely adopted in industrial, medical, and consumer applications due to their robustness, cost-effectiveness, and ability to function with any input medium (including gloved fingers). Unlike capacitive touch screens, resistive implementations do not require specialized conductive materials and offer superior performance in harsh environments.

The MSPM0L1306 microcontroller provides the necessary analog and digital interfaces to implement a complete resistive touch screen controller. This application note presents practical firmware implementations for touch detection and analysis using GPIO pins to drive the resistive layers and the integrated ADC to acquire coordinate measurements. Two example projects are provided: a basic touch detection system, and an advanced system capable of distinguishing between click events and drag/long-touch gestures through a timing based measurement using the Timer G peripheral.

A resistive touch screen works by applying a voltage across a resistor network and measuring the change in resistance at a given point on the matrix where a screen is touched by an input stylus, pen, or finger. This device consists of two transparent resistive layers made by ITO (Indium Tin Oxide) and separated by a thin air gap. When a user applies pressure with a stylus, pen, or finger, the top layer contacts the bottom layer, creating a closed circuit at that specific location. This method is used by the microcontroller to get the location of the touch.

The related software can be accessed at [MSPM0L1306-RESISTIVE-TOUCH-APPSW](#).

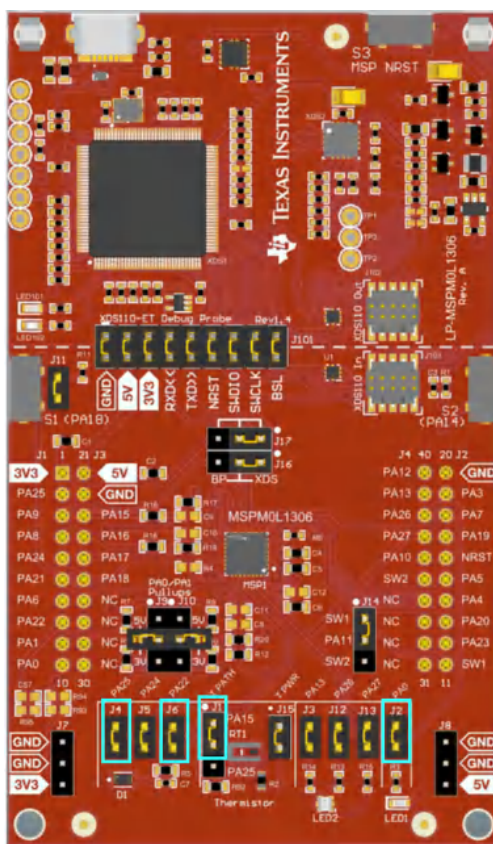
## 2 LaunchPad Setup

For the MSPM0L1306 LaunchPad, [Table 2-1](#) shows the pins and pin functions. This can also be seen in [Figure 3-1](#). For more details, refer to the SysConfig file in the software project.

**Table 2-1. LaunchPad Pin Connection**

| Used Pin | Pin Function        | Screen Control            |
|----------|---------------------|---------------------------|
| PA16     | GPIO; ADC channel 8 | Y-axis positive direction |
| PA24     | GPIO                |                           |
| PA0      | GPIO                |                           |
| PA17     | GPIO                | X-axis positive direction |
| PA25     | GPIO                | Y-axis negative direction |
| PA21     | GPIO                |                           |
| PA22     | GPIO                |                           |
| PA15     | GPIO; ADC channel 9 | X-axis negative direction |

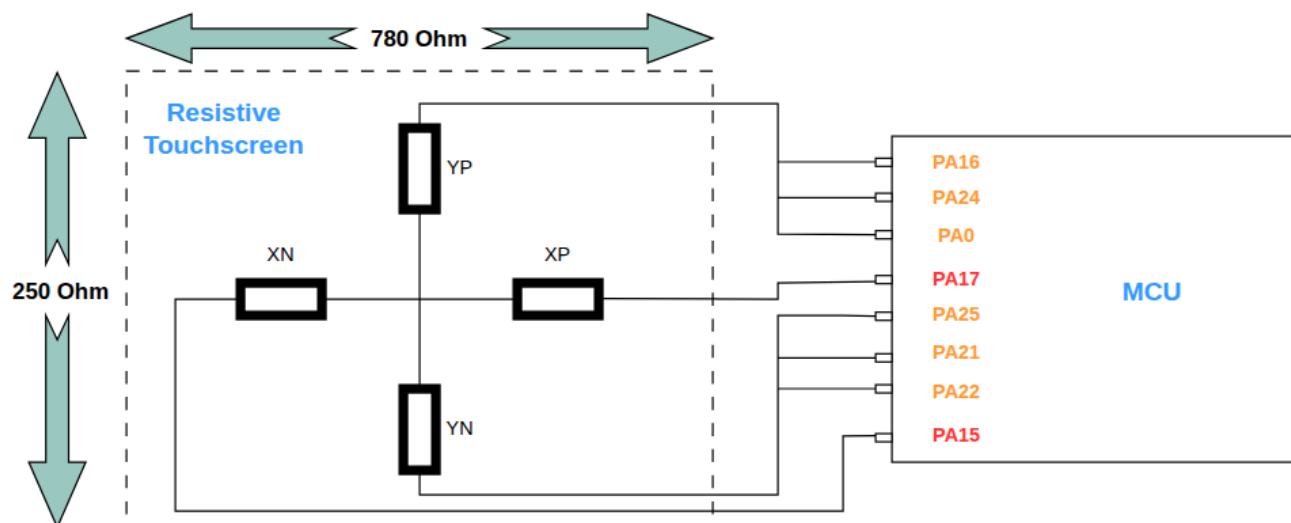
The pins used in this demo are also connected to other peripherals on the LaunchPad. These jumpers must be removed before testing as shown in [Figure 2-1](#). Removing these jumpers releases these pins from their default peripheral assignments, allowing them to function as GPIO and ADC pins. This is essential for the touchscreen implementation, because the firmware requires direct GPIO control and ADC sampling capability on these pins. The Jumpers, J4 is removed for PA25, J6 for PA22, J1 for PA15 and J2 for PA0.



**Figure 2-1. Removal of Launchpad Jumpers**

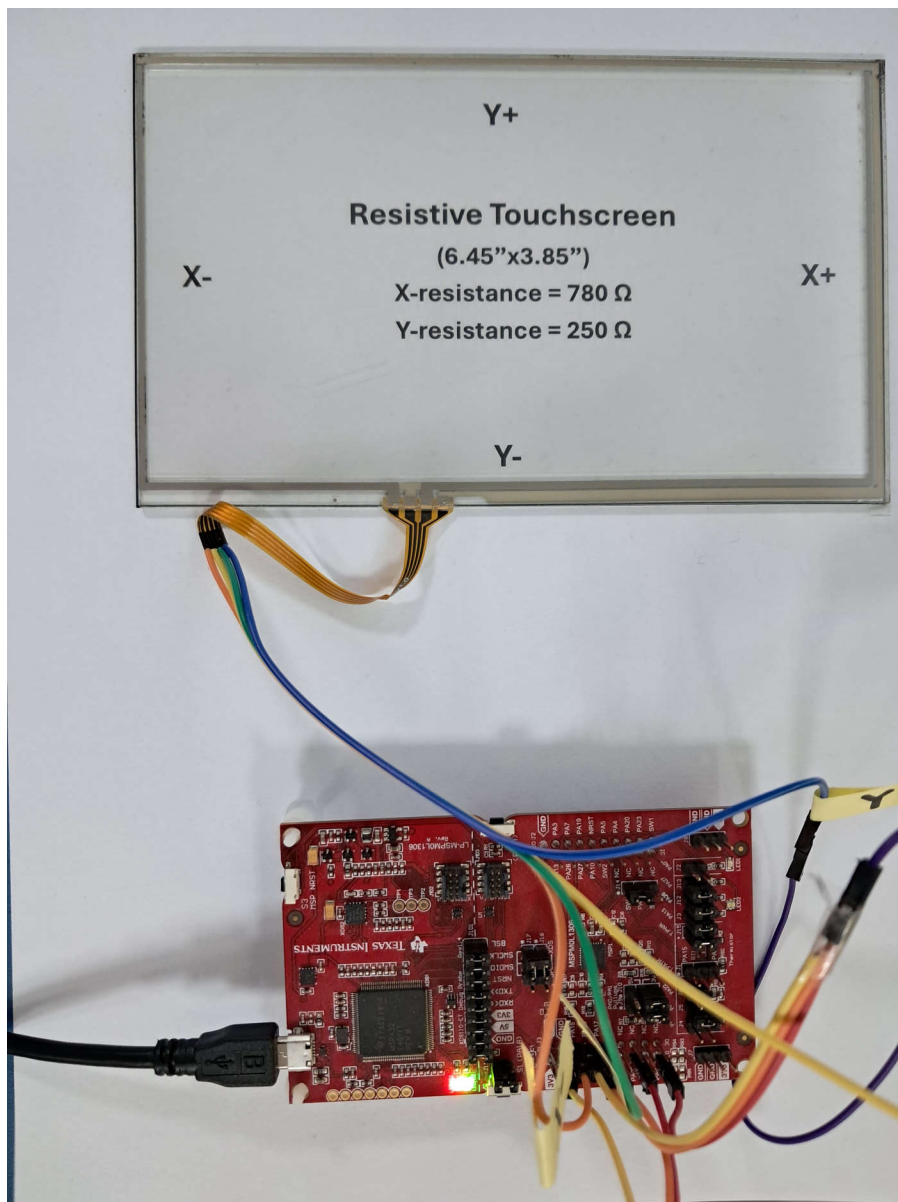
### 3 Hardware Connection

Figure 3-1 shows the system block diagram which follows the [Launchpad Pin Connection](#) table.



**Figure 3-1. System Block Diagram**

The hardware setup is shown in [Figure 3-2](#) where the details in the touchscreen can be seen. The Resistive Touchscreen used is of  $780\Omega$  and  $250\Omega$  resistance across the X and Y axis respectively. The pin configurations information for the resistive touchscreen being used and the calculations that need to be made based on the resistance, is shown in the [Design Considerations](#) section.



**Figure 3-2. Hardware Setup**

## 4 Design Considerations

### 4.1 IO Selection Consideration

In [Figure 3-1](#), 3 IOs are used for Y-axis direction drive and one IO is used for X-axis drive. There are two considerations to remember:

1. MSPM0 IO current output and input limitation
2. MSPM0 IO internal resistance

The resistive touch screen has X-axis: 780Ω and Y-axis: 250Ω. Use 3.3V as the system power supply.

$$I = V/R \quad (1)$$

That means users need 4.2mA to drive the resistance in X-axis, and need 13.2mA to drive the resistance in Y-axis.

The MSPM0 L1306 current limitation is 6mA as shown in [Table 4-1](#), and can be found in the [MSPM0 Datasheet](#). If users go beyond the specification, then permanent damage on MSPM0 can occur. That is why TI recommends to use 3 GPIOs to drive the Y-axis and one GPIO to drive the X-axis for this resistive touch screen. However, the number of pins required can change with the Touchscreen being used. Please follow the method provided above to find out the ideal number of pins required for implementation.

#### Note

When a user needs to add or remove a Pin based on the above resistance of the panel, they would also need to ensure that the Jumpers are removed in the MSPM0 such that the pin can function as required.

**Table 4-1. Absolute Maximum Ratings**

| PARAMETER        |                               | TEST CONDITIONS                            | MIN  | MAX                                | UNIT |
|------------------|-------------------------------|--------------------------------------------|------|------------------------------------|------|
| VDD              | Supply voltage                | At VDD pin                                 | −0.3 | 4.1                                | V    |
| V <sub>I</sub>   | Input voltage                 | Applied to any 5V tolerant open-drain pins | −0.3 | 5.5                                | V    |
| V <sub>I</sub>   | Input voltage                 | Applied to any common tolerance pins       | −0.3 | V <sub>DD</sub> + 0.3<br>(4.1 MAX) | V    |
| I <sub>VDD</sub> | Current into VDD pin (source) | −40°C ≤ T <sub>j</sub> ≤ 130°C             |      | 80                                 | mA   |
|                  | Current into VDD pin (source) | −40°C ≤ T <sub>j</sub> ≤ 85°C              |      | 100                                | mA   |
| I <sub>VSS</sub> | Current out of VSS pin (sink) | −40°C ≤ T <sub>j</sub> ≤ 130°C             |      | 80                                 | mA   |
|                  | Current out of VSS pin (sink) | −40°C ≤ T <sub>j</sub> ≤ 85°C              |      | 100                                | mA   |
| I <sub>IO</sub>  | Current of SDIO pin           | Current sunk or sourced by SDIO pin        |      | 6                                  | mA   |
|                  | Current of HSIO pin           | Current sunk or sourced by HSIO pin        |      | 6                                  | mA   |
|                  | Current of ODIO pin           | Current sunk by ODIO pin                   |      | 20                                 | mA   |
| I <sub>D</sub>   | Supported diode current       | Diode current at any device pin            |      | ±2 <sup>(1)</sup>                  | mA   |

- (1) Stresses beyond those listed under *Absolute Maximum Rating* can cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Condition*. Exposure to absolute-maximum-rated conditions for extended periods can affect device reliability.

MSPM0 internal IO resistance causes an offset on the resistive screen detection. For some high-resolution applications, this is not acceptable. If users want to reduce the influence, then there are two methods to further improve performance:

1. Add more IOs to drive the screen to spread the load current to more IOs pins.
2. Add ADC to both positive and negative axis IOs. Directly measure the voltage assigned to the IO internal resistors when driving the screen.

## 4.2 Measurement Method

### 4.2.1 Touch Detection Method

When the resistive layers are not connected, the ADC is floating. When there is no touch, the result is strongly affected by the noise coupled from IO switch. That means ADC constantly outputs a fixed coordinate. When touch happens, a resistance is included between two resistive layers, which is very high when the touch is very weak. This also generates disturbance on the ADC output.

The MSPM0 setup is shown in [Section 2](#). Set the Y-axis to positive IO and Y-axis to negative IO for all outputs high when doing the detection. In this demo, ADC is used to do the measurement. For low-power requirement, users can use GPIO instead and paired with the wake-up function.

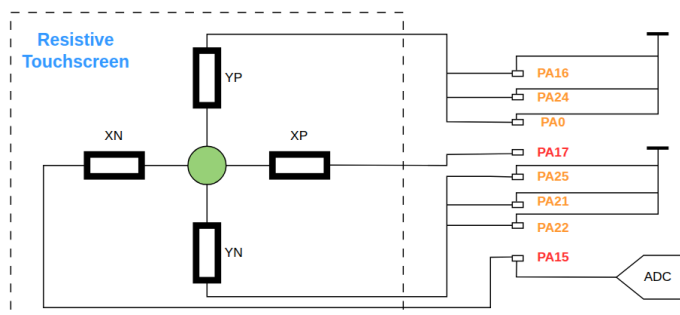


Figure 4-1. Touch Detection Setup

Use ADC to do the detection to filter the low-pressure condition for a touch action. The MSPM0 ADC is configured in 12-bit mode, which means the max output counter is 4095 for the ADC value being read. The `TOUCH_DETECTION_THD` macro defined in the `resistive_detection.h` file is set to 4000 to detect the touch detection on the resistive panel. Users can adjust this parameter value to check when to release the detection signal based on trial and error method depending on the noise of the ADC being read.

### 4.2.2 X-Axis and Y-Axis Measurement

When a user applies pressure, the top layer contacts the bottom, creating a closed circuit at that location. By applying a voltage across one axis (for example Y-axis) and measuring the voltage drop across the perpendicular axis (X-axis), the microcontroller can determine the precise X and Y coordinates of the touch point.

The touch detection method is shown in [Section 4.2.1](#). When there is a touch, the detection method establishes a complete circuit between the resistive layers. By applying a known voltage (3.3V), across one axis (Y-axis) and measuring the resulting voltage drop on the perpendicular axis (X-axis), the microcontroller calculates the precise touch location.

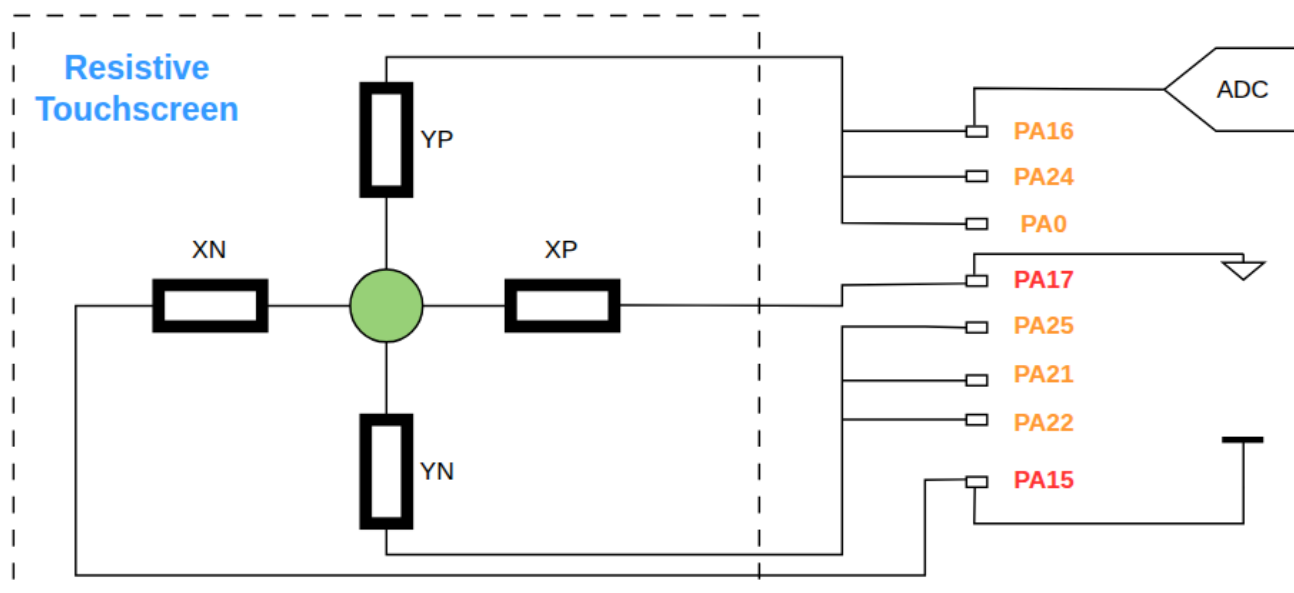


Figure 4-2. X-axis ADC Measurement

For the X-axis detection, set PA17 to output high and PA15 to output low. Set other pins to be analog functions (Hiz mode). Then, enable the ADC channel at PA16 to do the measurement.

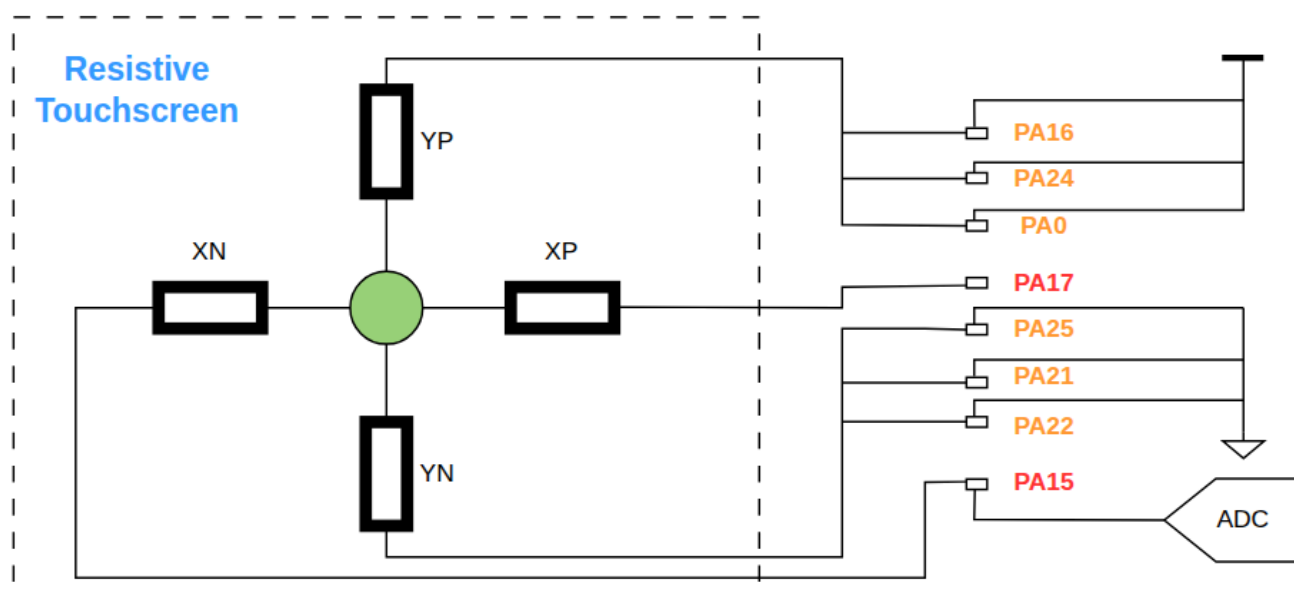


Figure 4-3. Y-axis ADC Measurement

For the Y-axis detection, set PA16, PA24 and PA0 to output high and, PA25, PA22 and PA21 to output low. Set other pins to be analog functions (Hiz mode). Then, enable the ADC channel at PA15 to do the measurement.

After getting the voltage, the ADC does an oversampling to reduce the noise and directly use the ADC result as the coordinate and output to UART console.

## 5 Software Structure

The software project developed in CCS is shown in [Figure 5-1](#). The software project mainly consists of three parts:

1. The resistive\_detection part includes the IO and ADC control to get the Y-axis and X-axis.
2. The UART part mainly includes the write function based on UART protocol for M0 and uploads the measured results to the PC through the back channel UART.
3. The main part includes the highest system function code.

After MCU powers on, the MSPM0 starts the resistive touch detection. If there is touch detected, then the resistive screen detection measures the voltage at Y-axis and X-axis and uploads the coordinate to the UART console.

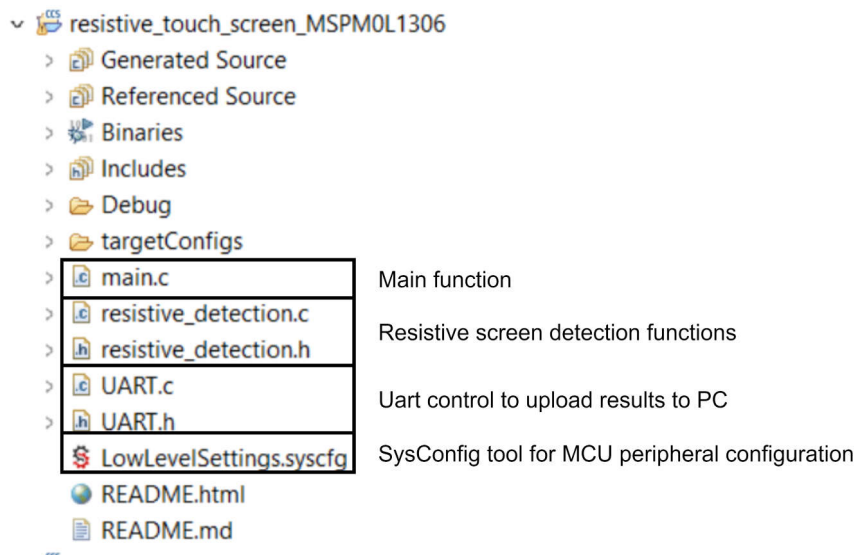


Figure 5-1. Software Project

### 5.1 Touch Detection Example

For the Touch detection example, any touch detected, prints the coordinate values at a sampling rate of 200ms (*INACTIVE\_SAMPLE\_DELAY*). In this example, there is no separation between a click and a long press or drag.

Once a touch is detected, the ADC values are read to get the X and Y coordinates and are sent through UART to the PC console.

#### Note

The user should update the Min and Max values of the ADC depending on the configurations of the Resistive Touchscreen panel being used.

### 5.2 Resistive Click and Drag Detection

For this example, sampling rate in ms for cases when there is no touch detected on the Resistive Touchscreen is 200ms (*INACTIVE\_SAMPLE\_DELAY*). There is a threshold set of 80ms (*LONG\_TOUCH\_THRESHOLD\_MS*) which is used to distinguish between a click/short press and a drag/long press. If the time taken of the touch is less than the threshold, the MSPM0 classifies this as a click. But if the touch exceeds the threshold, MSPM0 classifies this as a long press.

For Long press, there is continuous sampling at 4ms (*TOUCH\_RELEASE\_SAMPLING\_RATE*) which is used to detect when the touch is released. Each sampling produces a set of X and Y coordinates which are sent using the UART to the PC console.

The classification is done on the basis of time duration of the touch. This is calculated using the *TimerG* peripheral which is enabled using the Sysconfig tool. This helps in getting the start and stop time for the touch.

**Note**

The max touch duration on the current setup is of 100s. If the touch exceeds that duration, then the timer resets and will give the wrong reading of the time and also could hamper the classification of click and drag.

**Note**

The user should change the Min and Max values of the ADC of the resistive touchscreen panels being used. If not, this would hamper the coordinates being sent and the software may not work as intended.

### 5.3 Pin Mode Switching

This section explains the pin setup for *PA15* and *PA16* which were used as ADC pins and GPIO pins. This is done by performing pin mode switching the two pins. The microcontroller cannot simultaneously read and drive on the same physical pins. Therefore, a dual-phase measurement approach is required.

When reading the X coordinate, the Y-axis pins are driven as the GPIO outputs to create the voltage potential and the X-axis pin (*PA15*) is configured as the ADC input to measure the resulting voltage through Channel 8 of the ADC. This is implemented in the *readTouchX()* API in the *resistive\_detection.c* file. It is a similar implementation for reading the Y coordinate using the *PA16* pin. This necessitates switching of the pins *PA15* and *PA16* between GPIO and ADC modes.

The *touchIOReset()* function converts all touch pins back to peripheral analog mode using *DL\_GPIO\_initPeripheralAnalogFunction()*. The measurement functions then call *DL\_GPIO\_initDigitalOutput()* to temporarily switch specific pins to GPIO mode. After measurement completes, pins return to analog mode for the next cycle. A settle delay is applied after pin configuration changes to maintain voltage stabilization before the ADC measurement is done.

### 6 Steps for Execution

This section discusses the steps to follow to run the example for resistive touch detection on the MSPM0L1306 Launchpad.

1. Calculate the IOs (Resistance and Pins) needed by following the instructions in [Section 4.1](#).
2. Implement the Hardware Setup between MSPM0L1306 LaunchPad and the resistive touch screen following the instructions in [Section 3](#).
3. Install Code Composer Studio™ (CCS) on the computer.
4. Import the example given in the link in the [Introduction Section](#): load the *resistive\_touch\_detection* or *resistive\_click\_drag\_detection* example.
5. The user should add the X and Y axis, Min and Max values for the touchscreen being used. This maintains that no noise is sent as a coordinate.
6. User should also adjust the *TOUCH\_DETECTION\_THD* and detection frequency according to the sensitivity and noise of the panel.
7. Build the example.
8. Create a Target configuration for MSPM0 in CCS, *MSPM0\_XDS110.ccxml*, select the XDS110 Debug Probe and the MSPM0L1306 Device type.
9. Right-click on *MSPM0\_XDS110.ccxml* in Target Configurations window
10. Select *Launch Selected Configuration*
11. Select Run, Connect Target
12. Select Run, Load, Load Program
13. Browse to the binary for the MSPM0 project, and click OK.
14. This writes the flash with the binary.
15. After touching the Resistive Touchscreen, the ADC values for the X and Y axis are found and the result is sent through the UART peripheral to the Command line as shown in the [Results Section](#). The coordinates are only updated when there is touch detected on the panel.

## 7 Results

### 7.1 Touch Detection

Figure 7-1 shows the outputs for the ADC that is configured to a 12-bit resolution.

```

DEBUG: Touch detection started
TOUCH DETECTED! X=2183 Y=1199
TOUCH DETECTED! X=1721 Y=1235
TOUCH DETECTED! X=1713 Y=1243
TOUCH DETECTED! X=1711 Y=1238
TOUCH DETECTED! X=3246 Y=2658
TOUCH DETECTED! X=1555 Y=2702
TOUCH DETECTED! X=2065 Y=2707
TOUCH DETECTED! X=2965 Y=571.
TOUCH DETECTED! X=3424 Y=498.
TOUCH DETECTED! X=2300 Y=477.
TOUCH DETECTED! X=1689 Y=470.
TOUCH DETECTED! X=1453 Y=477.
TOUCH DETECTED! X=3518 Y=487.
TOUCH DETECTED! X=1316 Y=462.
TOUCH DETECTED! X=1019 Y=447.
TOUCH DETECTED! X=980. Y=452.
TOUCH DETECTED! X=932. Y=465.
    
```

Figure 7-1. Touch Detection: 12-bit Resolution

## 7.2 Click and Drag Detection

The two figures, [Figure 7-2](#) and [Figure 7-3](#) show the output on the different types of press - click and long press (or drag), that can be done by a user on the resistive touch screen.

```
TOUCH DETECTED! X=2160.0 Y=1902.0 TIME=54.000ms - CLICK
TOUCH DETECTED! X=3053.0 Y=1656.0 TIME=46.000ms - CLICK
TOUCH DETECTED! X=3143.0 Y=1661.0 TIME=69.000ms - CLICK
TOUCH DETECTED! X=3893.0 Y=1577.0 TIME=38.000ms - CLICK
TOUCH DETECTED! X=3088.0 Y=1448.0 TIME=69.000ms - CLICK
TOUCH DETECTED! X=3265.0 Y=1478.0 TIME=54.000ms - CLICK
TOUCH DETECTED! X=3616.0 Y=1414.0 TIME=54.000ms - CLICK
```

**Figure 7-2. Output for a Click**

```
TOUCH DETECTED! X=2335.0 Y=1066.0 (2364.0,1092.0) (2723.0,1061.0) (3992.0,531.00) TIME=171.00ms - LONG TOUCH
TOUCH DETECTED! X=2374.0 Y=1173.0 (2358.0,1209.0) (2418.0,1198.0) (2412.0,1193.0) (2417.0,1199.0) (2387.0,1207.0) (2336.0,1221.0) (2331.0,1228.0) (2332.0,1229.0) (2352.0,1221.0) (2436.0,1185.0) (3032.0,1158.0) TIME=396.00ms - LONG TOUCH
TOUCH DETECTED! X=4056.0 Y=583.00 (2864.0,1392.0) (2881.0,1435.0) (2945.0,1485.0) (2929.0,1533.0) (3042.0,1630.0) (2557.0,1675.0) (2688.0,1732.0) (2719.0,1760.0) (2776.0,1805.0) (3055.0,1783.0) TIME=365.00ms - LONG TOUCH
TOUCH DETECTED! X=2980.0 Y=1405.0 (2552.0,1451.0) (2301.0,1466.0) (2933.0,1476.0) (2786.0,1538.0) (2595.0,1622.0) (2790.0,1659.0) (2518.0,1688.0) (2515.0,1723.0) (2526.0,1763.0) (2603.0,1798.0) (2761.0,1870.0) (2719.0,1922.0) (2790.0,1945.0) (3135.0,1958.0) TIME=489.00ms - LONG TOUCH
TOUCH DETECTED! X=2201.0 Y=830.00 (2086.0,961.00) (2144.0,1000.0) (2346.0,1051.0) (2444.0,1100.0) (2540.0,1183.0) (2863.0,1264.0) (2777.0,1345.0) (3221.0,1527.0) (3080.0,1481.0) (3841.0,1632.0) (3942.0,1576.0) TIME=388.00ms - LONG TOUCH
TOUCH DETECTED! X=2154.0 Y=1024.0 (1932.0,1062.0) (2235.0,1031.0) TIME=139.00ms - LONG TOUCH
TOUCH DETECTED! X=2305.0 Y=1320.0 (2859.0,1383.0) TIME=108.00ms - LONG TOUCH
```

**Figure 7-3. Output for Long Press**

## 8 Summary

This application note provides comprehensive instructions for implementing resistive touch screen control on the MSPM0L1306 microcontroller using GPIO and ADC interfaces. By documenting the control methodology, critical implementation considerations, and providing working software examples for simple touch detection and, click and drag detection, this guide enables developers to successfully port the resistive touch screen implementation with minimal modification.

## 9 References

- Texas Instruments, [MSPM0L1306-RESISTIVE-TOUCH-APPSW](#), webpage.
- Texas Instruments, [MSPM0L130x Mixed-Signal Microcontrollers](#), datasheet.

## 10 Revision History

| Changes from Revision * (August 2025) to Revision A (September 2026)                                                                                                    | Page |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| • Added details on the click and drag example.....                                                                                                                      | 1    |
| • Updated the introduction with details on the resistive touchscreen and MSPM0 microcontroller. Added details on the click and drag example.....                        | 2    |
| • Updated the numbering format for tables, figures, and cross-references throughout the document.....                                                                   | 2    |
| • Added 3rd pin for YP and YN. Updated the Launchpad Jumper image as 1 additional jumper is removed due to the choice of pins taken - and added reasoning for this..... | 3    |
| • Updated the images based on the touchscreen and setup used to implement the software.....                                                                             | 4    |
| • Added information on pin configuration calculations and note to consider when adding or removing the Pins.....                                                        | 6    |
| • Updated the figures with 3-pins for Y positive and Y negative due to difference in panel resistance.....                                                              | 7    |
| • Added details on the Click and Drag detection example, and pin mode switching.....                                                                                    | 9    |
| • Updated with detailed steps on the execution process to run the example.....                                                                                          | 10   |
| • Added touch detection results for 12-bit ADC resolution and the click and drag detection example.....                                                                 | 11   |

## 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, regulatory 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 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](#), [TI's General Quality Guidelines](#), 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025