Application Report
SLAA481A–April 2011–Revised May 2012

Two-Dimensional Capacitive-Touch Implementation Using the High-Resolution Timer_D of the MSP430F5132

Zack Mak *MSP430 Systems Applications*

ABSTRACT

This application report describes the implementation of a two-dimensional capacitive-touch pad, using the high-resolution Timer_D of the MSP4305132 MCU.

Project collateral and source code discussed in this application report can be downloaded from the following URL: www.ti.com/lit/zip/SLAA481.

Contents

List of Figures

Trademarks

Code Composer Studio is a trademark of Texas Instruments.

1 Introduction

One of the challenges of using a relaxation oscillator to measure capacitive touch is that the typical change in capacitance is very small. In the conventional method where the number of pulses is recorded in a fixed observation window, the length of the window has to be made long enough in order to count a large number of pulses. A large number of pulses is proportional to high sensitivity. Therefore, there is an inverse speed vs sensitivity relationship.

The MSP430F5312 from Texas Instruments has the capability of generating as well capturing pulse widths to a fine resolution of a few nanoseconds using the Timer_D. This special capability is utilized to measure the capacitance change by measuring the very fine pulse width changes instead of the conventional method outline above. In this way, a fast scan rate with little or no compromise to sensitivity is achieved. In this implementation, an insulation thickness of up to 1 mm can be used with good sensitivity.

As an example for this application report, a wireless mouse pad remote controller is used to demonstrate how this new method of detection can be put to good use.

The following sections describe how the MSP430F5132 uses its Timer_D and other on-chip peripherals to realize a high-speed two-dimensional touchpad.

The demonstration board design builds on top of this and provides a complete RF4CE-based capacitive mouse pad remote controller system. The RF4CE details can be found in http://www.ti.com/corp/docs/landing/RF4CE/index.htm.

Some information in this document refers to *PCB-Based Capacitive Touch Sensing With MSP430* (SLAA363) and the *MSP430 Capacitive Single-Touch Sensor Design Guide* (SLAA379). It is recommended to read this application report with those two documents.

2 Hardware Implementation

2.1 System Diagram

Figure 1 shows the block diagram of the RF4CE-based capacitive touchpad remote control system, which consists of the transmitter and the receiver.

The transmitter is made up of the capacitive touch pad, 20 buttons, and the RF4CE wireless module.

The RF4CE USB dongle is the receiver, which provides an HID mouse interface to the PC. After the system is activated, the PC's mouse pointer can be controlled by the transmitter unit.

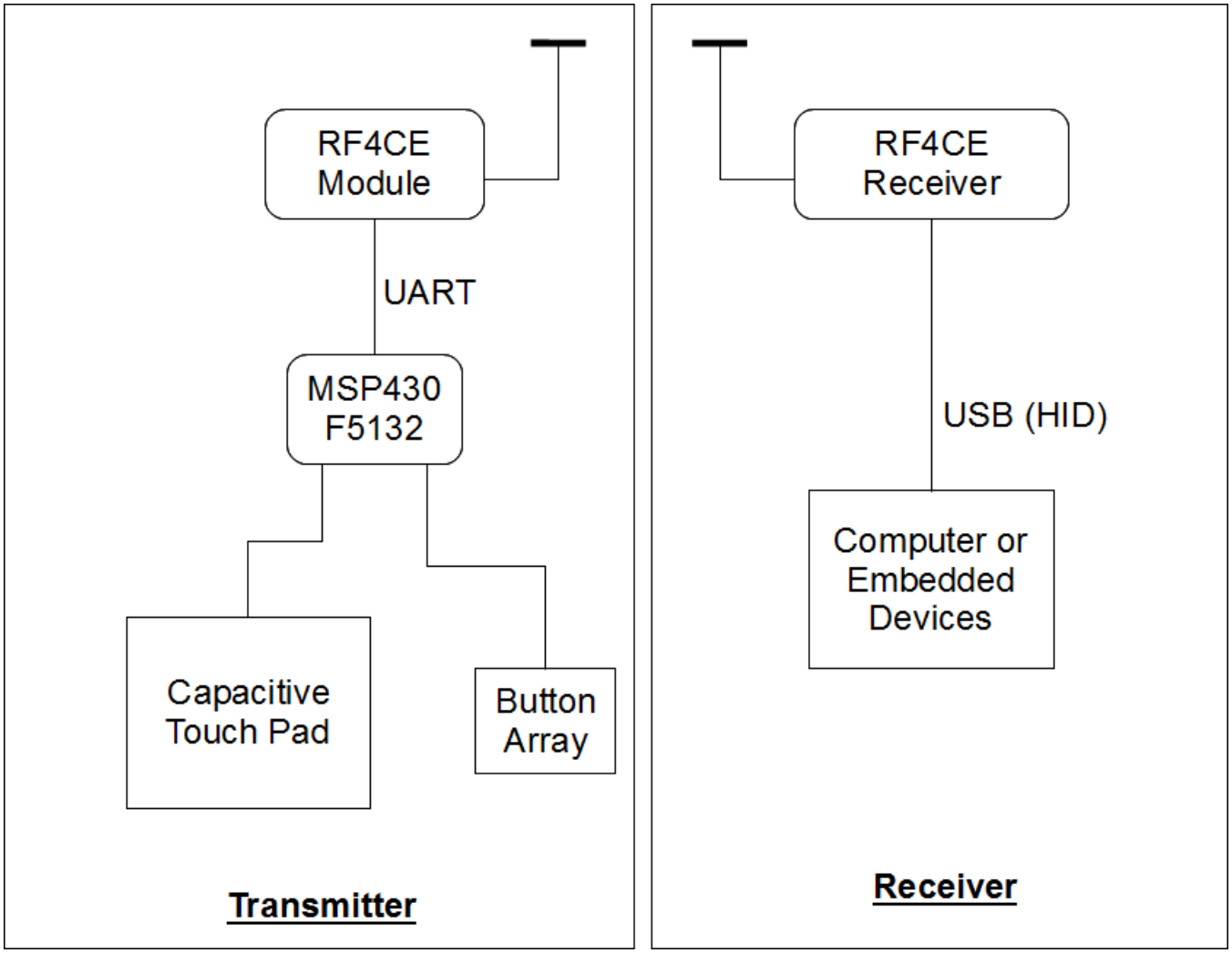


Figure 1. System Block Diagram

2.2 Touch Pad

2.2.1 Capacitive Touch Sensors on MSP430F5132

The connection of a single capacitive sensor with MSP430F5132 is shown in Figure 2. The Comparator_B, the resistor and the capacitance of the sensor pad forms a relaxation oscillator. The oscillation output is generated on CBOUT. The CBOUT is also connected to the Timer_D input that records the frequency of the CBOUT by measuring its period.

Figure 3 shows the working principle of the oscillation. In the Comparator_B of the MSP430F5132, the reference voltage is configurable. There are two reference voltage sources: CBREF0 and CBREF1. The reference voltage can be switched between these two voltage sources according to the output of CBOUT.

When CBOUT is high, the Comparator_B sets the internal reference voltage as CBREF1; CBx charges up. If CBx is higher then CBREF1, since it is connected to the inverting input, the CBOUT becomes low and also sets the internal reference voltage as CBREF0; CBx discharges. If CBx is lower then CBREF0, CBOUT goes high again. The frequency of the oscillation depends of the capacitance of the sensor pad, the resistor, and the internal reference voltage of the Comparator_B.

As the resistor and the internal reference voltage are fixed, the only variable is the capacitance of the sensor pad. If a finger is placed on the sensor pad, the capacitance of the sensor pad increases, which increases the charging/discharging time of the CBx. Therefore, the frequency of CBOUT decreases.

For more information, see the *Section 3: Measuring a Capacitive Touch Sensor Using the MSP430* in *PCB-Based Capacitive Touch Sensing With MSP430* (SLAA363).

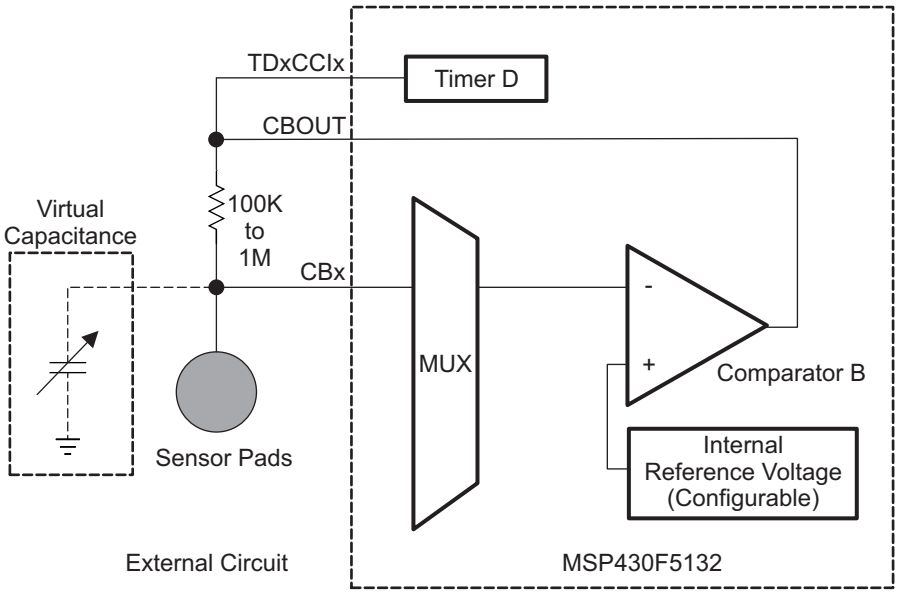


Figure 2. Single Sensor Block Diagram

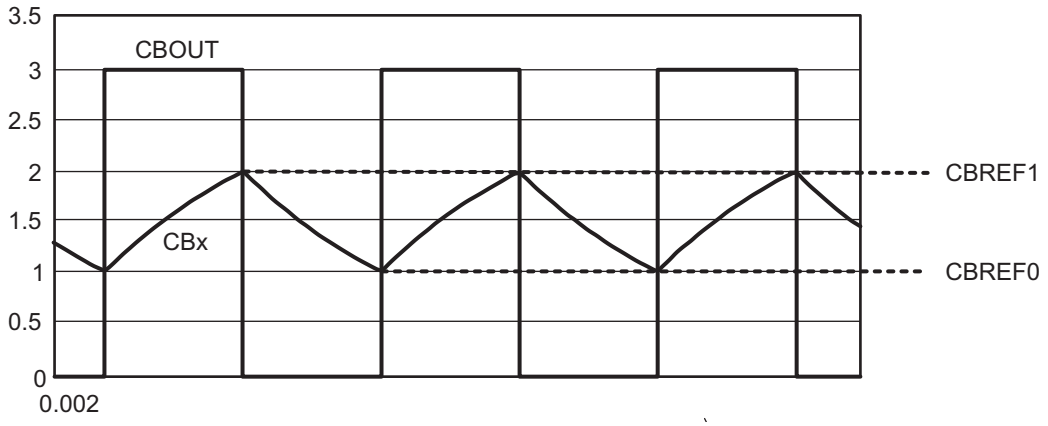


Figure 3. Waveform of CBOUT and the Sensor

2.2.2 Finger Detection Using Timer_D

In a relaxation oscillator implementation, the frequency output of Comparator_B (CBOUT) varies by the base capacitance of the sensor pad, the resistor, and the internal reference voltages. This frequency can be between 100 kHz and 1 MHz depending on the hardware design and the internal reference setting.

When the capacitive pad is touched, a small variation is seen in the frequency. In conventional designs, the Comparator_B output becomes the clock of Timer_A or Timer_B. A separate timer, for example the watchdog timer, sets a fixed interrupt period during which the number of pulses is recorded. To have enough sensitivity, typically, the interrupt period has to be long enough to ensure hundreds of pulses are captured.

Using the Timer_D in high-resolution mode, pulse widths can be measured to a resolution of down to 4 ns. This fine granularity means that instead of counting a large number of pulses, the pulse widths of a small number of pulses, for example eight pulses, can be captured.

The Timer_D is configured in captured mode. Each rising edge of the CBOUT is captured and the period between two consecutive rising edges is later calculated.

The lower part of Figure 4 shows that the Timer_D captures N+1 rising edges to measure the period of N pulses. The timer count, the difference in Timer value T1 and T2, represents the period of one pulse. When there is no object placed on the sensor pad, the timer count represents the base capacitance of the sensor pad. When there is an object, such as a finger, placed on the sensor pad, the oscillating frequency reduces, and thus the pulse period increases. The timer count becomes higher. If the change of the timer count is big enough to a certain level, a valid touch is detected.

The MSP430F5132 DMA is used to store the captured timer values into a buffer, so the whole measurement is done automatically without MCU operations. As Timer_D captures the rising edges of CBOUT, it also triggers the DMA to store the timer values to a desired buffer. When the measuring process is finished, the MCU processes the buffer.

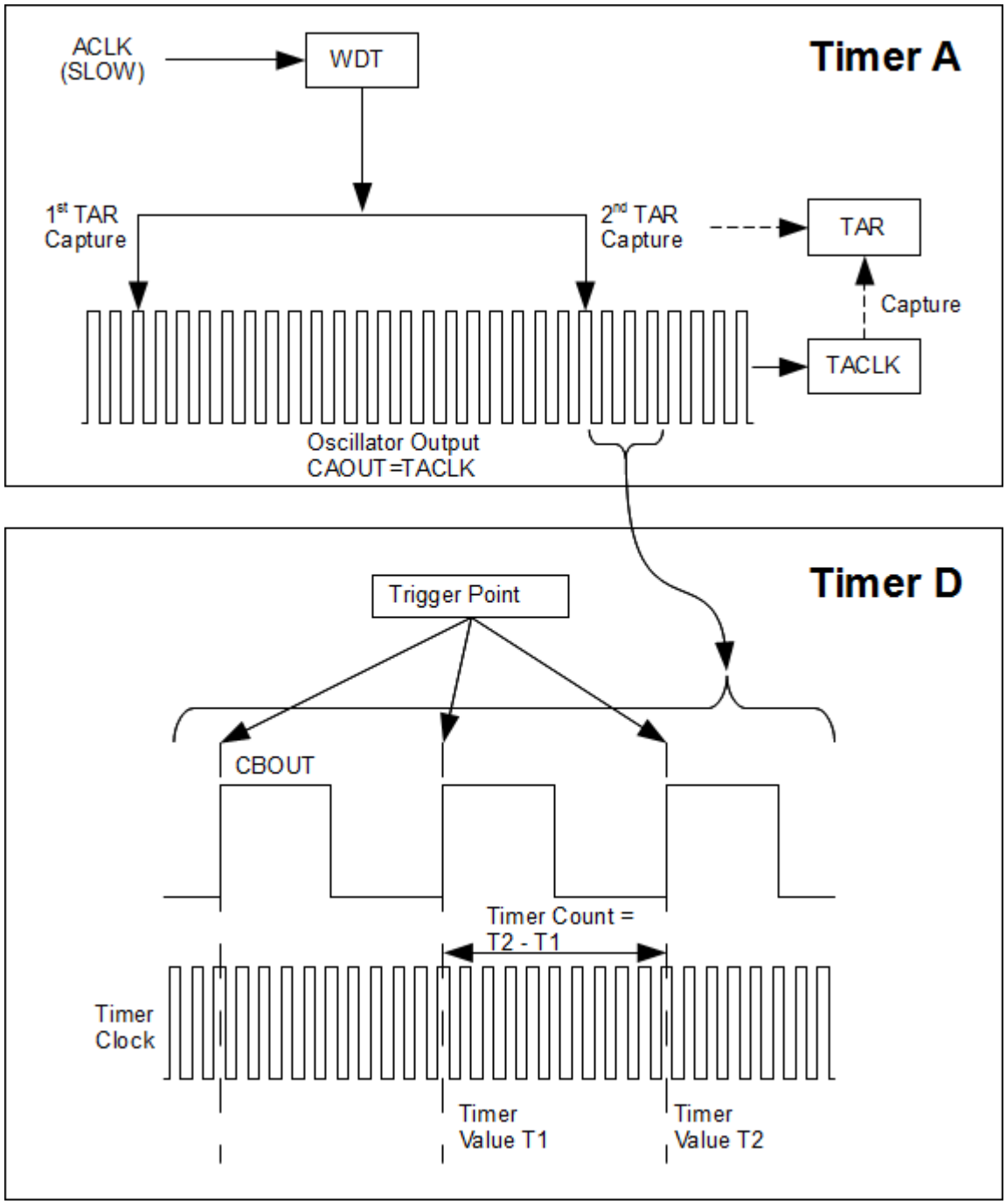


Figure 4. Measurement of Waveform Using Timer_A vs Timer_D

2.2.3 Touch Pad Using Multiple Sensors

One capacitive sensor can act as a simple button; multiples of sensors can form a slider and provide 1D position detection.

Comparator_B of the MSP430F5132 supports multiple input configurations using a MUX of the Comparator_B. It supports up to 16 sensor pads. An example connection is shown in Figure 5.

The capacitive touch pad consists of multiple capacitive touch sensors that are configured as two sliders, which are used to detect an object in X and Y axis on the pad (see Figure 6).

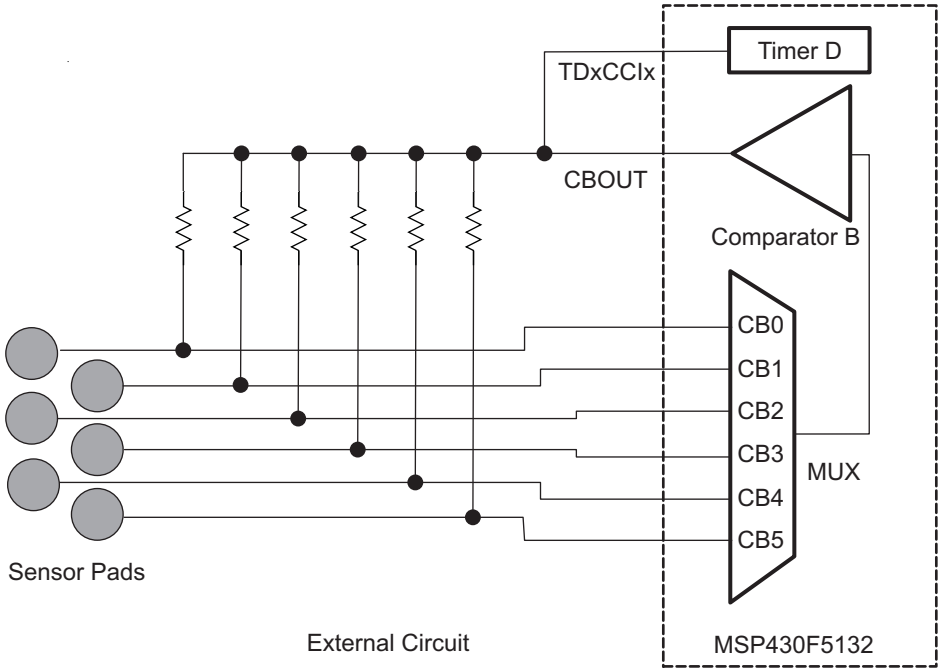


Figure 5. Multiple Sensor Block Diagram

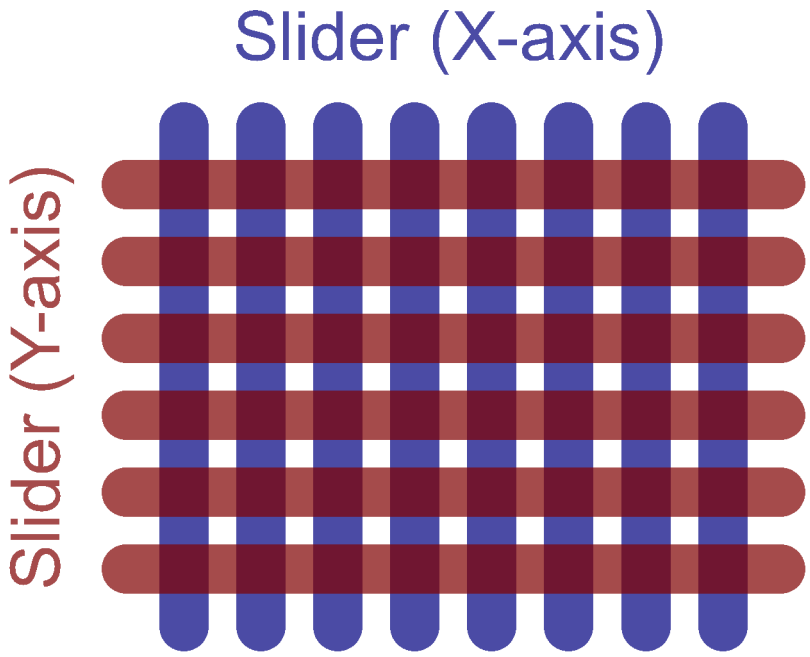


Figure 6. Two Sliders Form a 2D Touch Pad

3 Software Implementation

The software provides basic movement detection in order to send data to the host for moving the mouse pointer. It also provides a tapping function to simulate mouse clicks.

On the transmitter side, the MSP430F5132 continuously measures every sensor pad of the touch pad. After getting the pulse widths of CBOUT using Timer_D, the finger's position and its movement are worked out. If it detects a moving finger on the touch pad, data is sent to the RF4CE transmitter through the universal asynchronous receiver/transmitter (UART).

On the receiver side, the RF4CE USB dongle receives the data, translates it into USB HID protocol, and sends it to the host.

The software flow of the MSP430F5132 is shown in Figure 7.

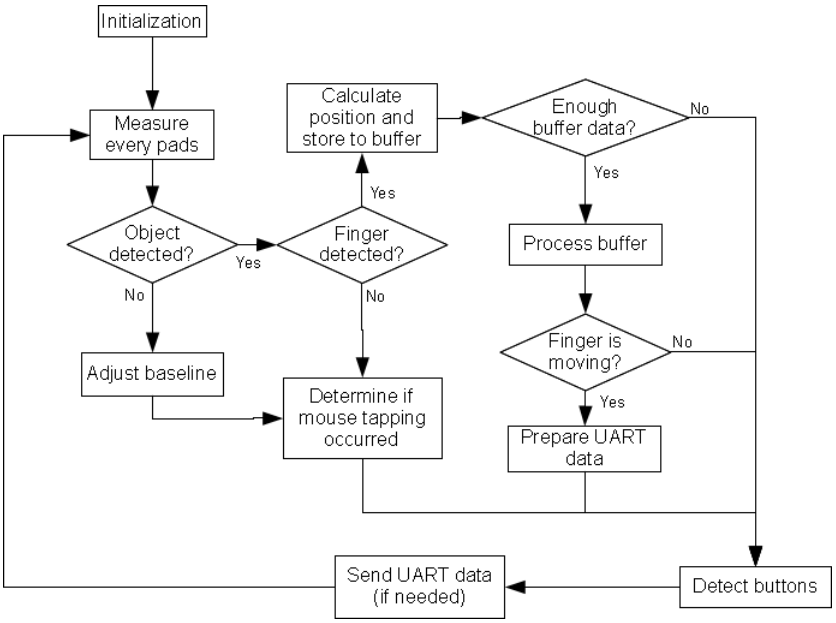
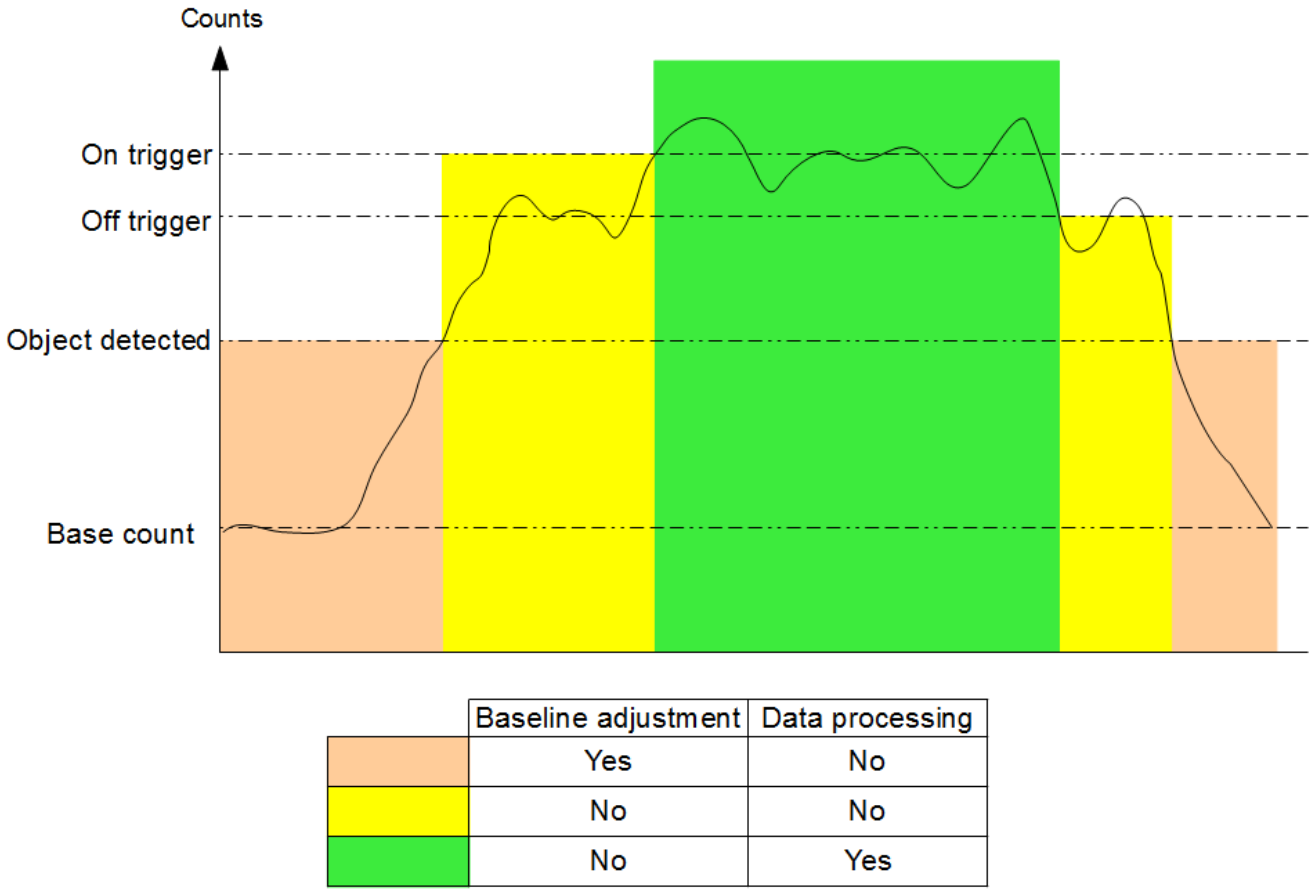


Figure 7. Software Flowchart

3.1 Finger Position Detection

As described in Section 2.2.2, once the timer count values are obtained, there are two values to be measured. The timer count of the base capacitance, the **base count**, is the reference count of the sensor pad. And the timer count of the measured capacitance, the **measured count**, is used for comparing with the base count. The difference between the measured count and the base count, the **signal count** where **signal count = measured count – base count**, is used for calculation. If the signal count is high enough to a certain level, a valid touch is detected.

The signal count is not always stable due to environmental noise and small jitter in the timer clock. Implementing simple buttons using capacitive sensors is rather easy, because the buttons are independent of each other. Applying a low-pass filter to the signal count, and using independent on and off thresholds is usually enough. For sliders and touchpads, the values of the signal count of the sensor pads are required for calculating object position. Therefore, more filters are used for signal conditioning and position calculation.



	Baseline adjustment	Data processing
	Yes	No
	No	No
	No	Yes

Figure 8. Hysteresis is used for capacitive buttons by using on/off triggers

3.1.1 Filtering Signal Count

Implementing a low-pass filter to the signal count is an easy and efficient way to reduce high-frequency jitter. For low-frequency jitter, hysteresis is used so that a small range of variation of the signal count is filtered. It helps to maintain the filtered signal count and the calculated position stable.

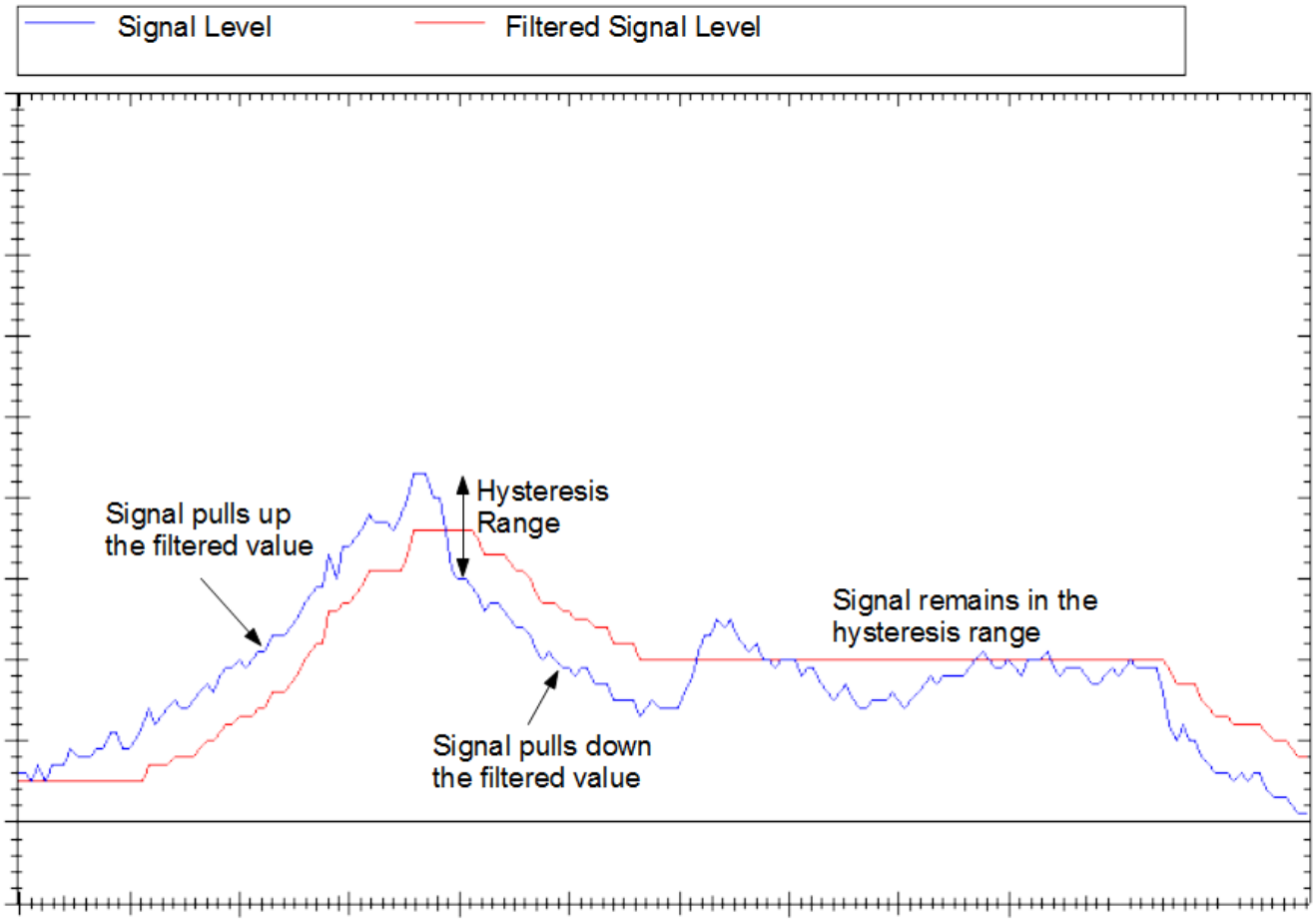


Figure 9. Filtering Signal Level Using Hysteresis

3.1.2 Calculate Finger Position on a Slider

Sliders are implemented by placing sensor pads close together. To calculate finger position on a slider, a weighted averaging method is used. Each pad is assigned with a weight incrementally according to the sensor pad's position. If a sensor pad gets the highest signal count between all pads, the calculated position approaches to the weight of that sensor pad. The resolution of the slider depends on the step of the weight between adjacent pads.

$$\frac{\sum_{i=1}^{i=n+1}\left(i \times signal\ count[i-1]\right)}{\sum_{i=1}^{i=n+1}\left(signal\ count[i-1]\right)} \times gain \qquad \left(weitht = i \times gain\right)$$

Figure 10. Weighted averaging for calculating position.

If a finger is placed at the border of the both sides of the slider, due to no sensor pads outside the slider, the calculated position is invalid. It is required that the signal count of the second sensor pad is high enough to make sure that the finger is inside the slider in order to have valid calculating results.

Apart from detecting finger position, finger movement can also be tracked by detecting finger position continuously. It is also desirable to filter the movement data in order to provide a smoother and better user experience. Low-pass filter can be used to filter the position data by storing the previous position data and averages them with the current position so that the mouse pointer moves smoothly over time. If a finger is placed and stays at the same position on the slider, the position data may have some jitter. A dead zone can be used so that this small range of unstable movement is ignored. If the finger moves away from this dead zone, it is defined as finger movement and the dead zone should be updated.

3.1.3 Multiple Finger Detection on a Slider

It is also possible to detect multiple fingers on the slider. The signal counts of each pad can be visualized in a bar chart. Figure 11 shows the signal data of eight pads when there are two fingers placed on the slider. It shows 2 peaks in the chart. The MCU detects and records the peaks. Then the finger positions can be calculated by weighted averaging method. Different from single finger calculation, not all the sensor pads are used in the calculation. Only the data from the peak and the adjacent sensors are used.

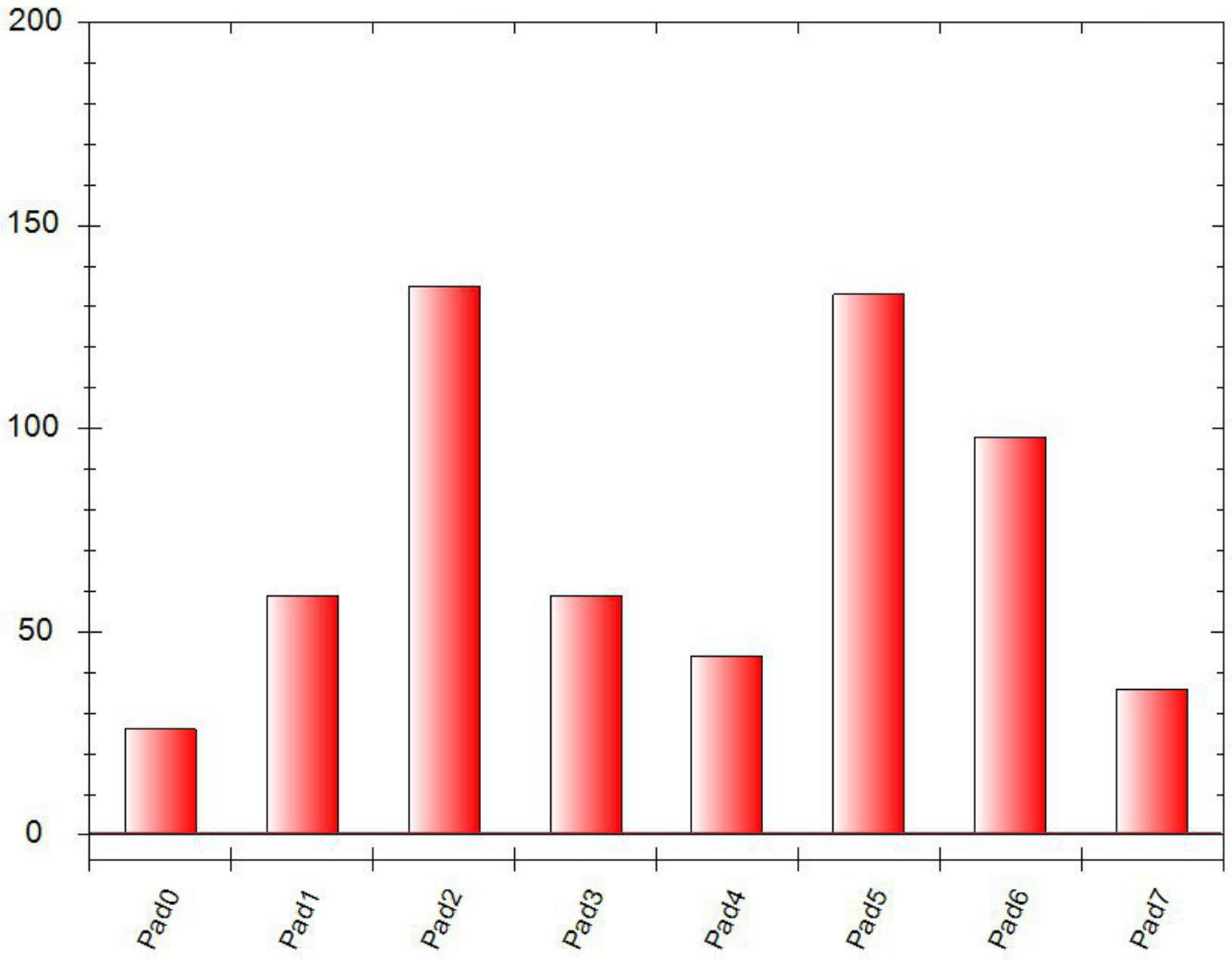


Figure 11. Visualized Signal Counts of All Pads of a Slider

3.1.4 Finger Detection on Slider Type Touchpad

A touchpad can be made by combining two sliders in X and Y axis. A single finger position on the pad can be found by simply combining the measuring results of the X and Y sliders to get an [X, Y] coordinate.

For multiple finger detection, the story is different. In some cases, different finger positions or movement get the same measuring results from the sliders. That confuses the MCU to find the actual position of each finger. Figure 12 shows a scenario that different finger positions causes the same calculation result.

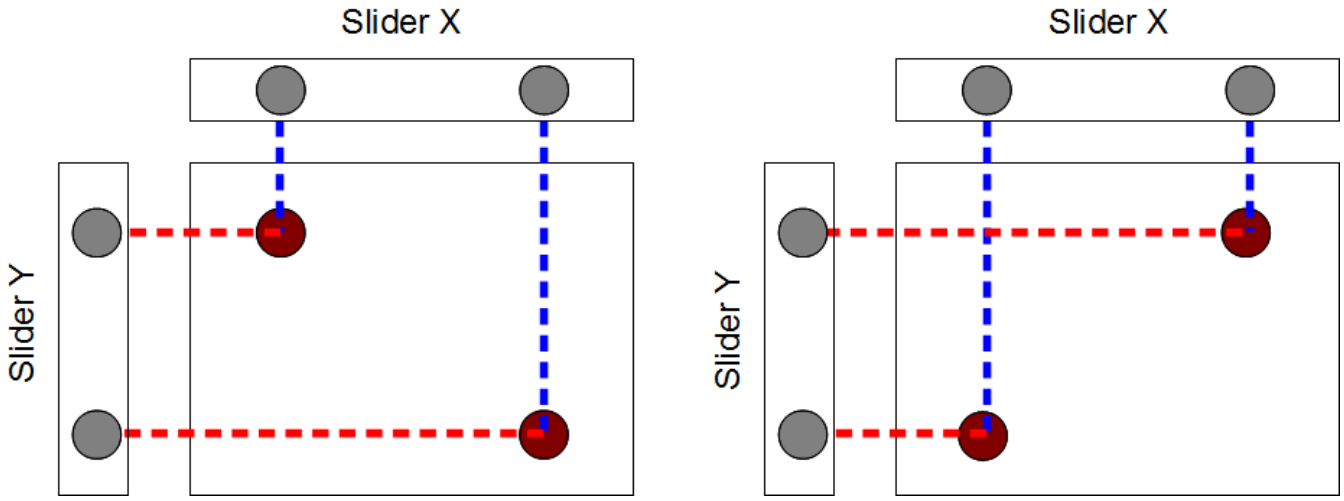


Figure 12. Example of Same Measuring Result of Different Finger Positions

Although this limitation prevents slider type touchpad from implementing true multi-touch function, it is still possible to implement simple multi-touch gestures on the touchpad when only one slider is performing multiple finger detection. For PC usage, some useful gestures such as vertical scrolling, page zooming, or right clicking can be implemented by enabling multiple finger detection only on X slider.

3.2 Mouse Tapping

Mouse clicking is simulated by tapping fingers on the capacitive touchpad. Mouse taps are determined by how long the finger stays on the touchpad as shown in Figure 13.

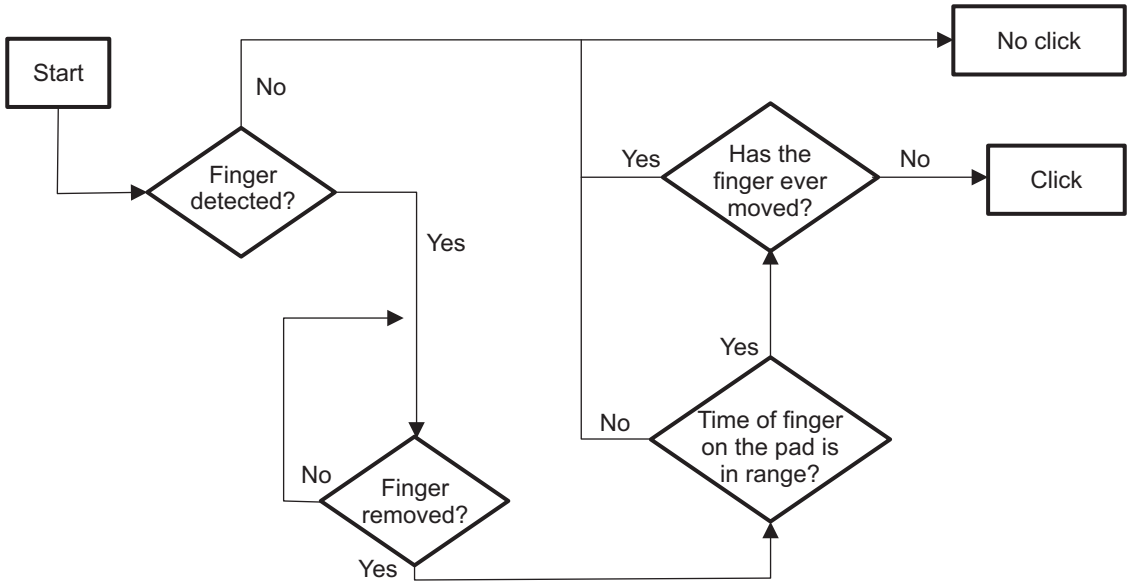


Figure 13. Flowchart of Determination of a Mouse Click

3.3 Software Considerations

There are some considerations for modifying the sample code.

- The sample code is styled in the hierarchy of *touchpad.c→slider.c→basic_cap_sense.c→hardware.c*. Programmers can exclude some files if any specific functions are not needed.
- The profile_X.c stores the hardware configuration for easy implementation of different hardware. Programmers can make use of the macro in profile.c and the macro setting of each *build configuration* of the CCS project to set different hardware profile in one single CCS project.
- The structure struct_cap_group is suitable for single button or slider configuration. Here is the example configuration of a slider using four sensor pads connecting to CB3, CB7, CB5 and CB11, respectively. The map_x is the array storing the CB input channel number corresponding to the slider.

```
#define NUM_PAD_X    4
const unsigned char    map_x[NUM_PAD_X] = { 3, 7, 5, 11 };
unsigned int base_level_x[NUM_PAD_X];
unsigned int count_x[NUM_PAD_X];
int cal_level_x[NUM_PAD_X];
int adaptor_x[NUM_PAD_X];

struct_cap_group map_data_x = {
NUM_PAD_X,
map_x,
count_x,
base_level_x,
cal_level_x,
adaptor_x,
      :
      :
};
```

- The sample code runs in an infinite for-loop for polling. If there is no finger detected, the MSP430F5132 is placed in LPM3 mode to save power.

4 Loading the Example Code With Code Composer Studio™

The example code for the application can be compiled with Code Composer Studio 4.1.3. This code can be downloaded from the following URL: http://www-s.ti.com/sc/techlit/slaa481.zip.

To run the code:

1. Unpack the .zip file.
2. Launch Code Composer Studio.
3. Import the project folder *CCS Project\Remote Control* into the workspace (see Figure 14).

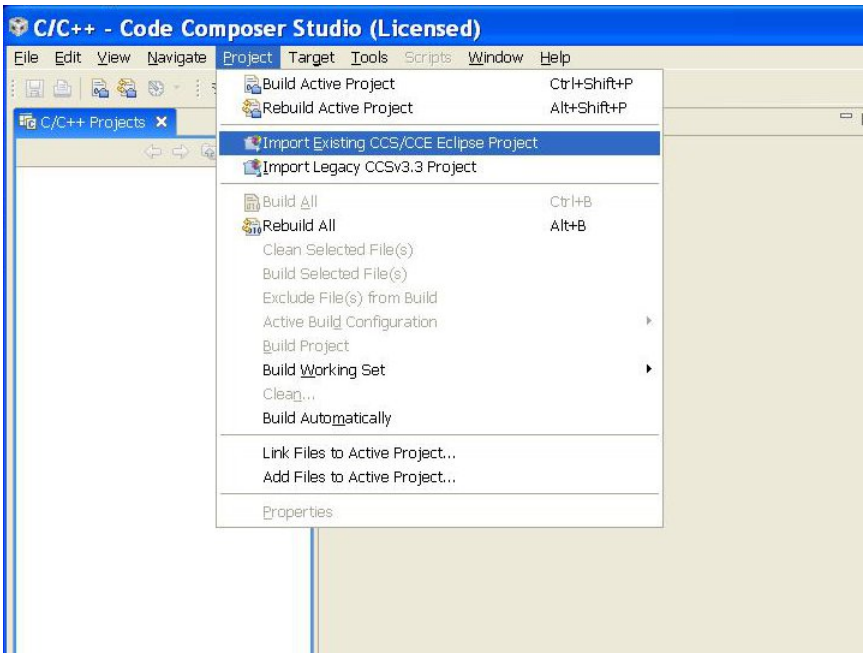


Figure 14. Import Existing Code Composer Studio Project

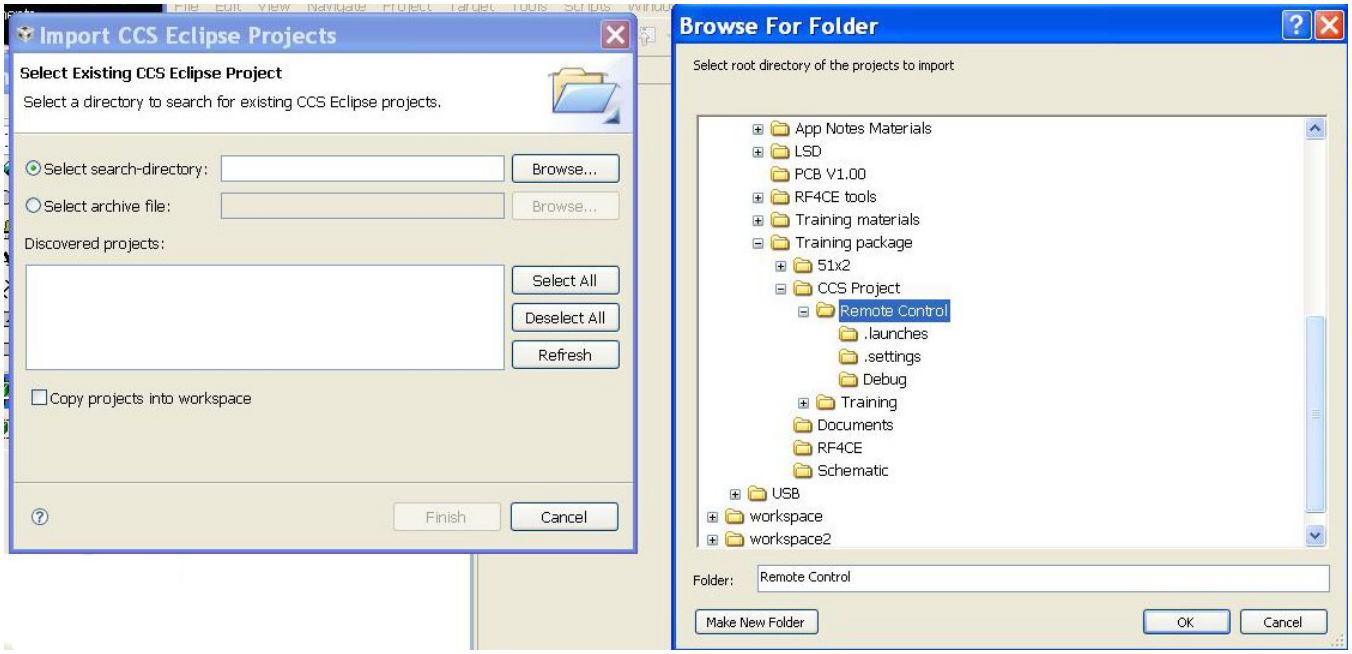


Figure 15. Select Folder

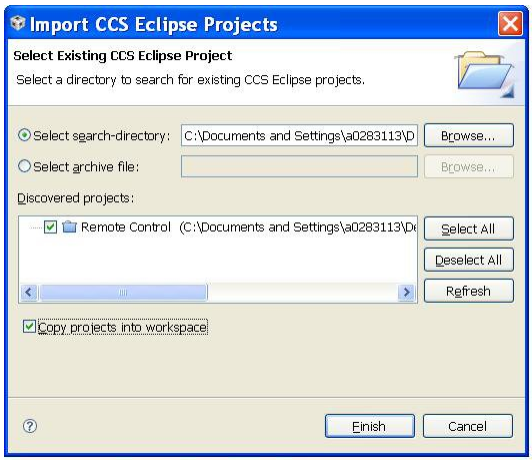


Figure 16. Choose 'Copy Projects Into Workspace'

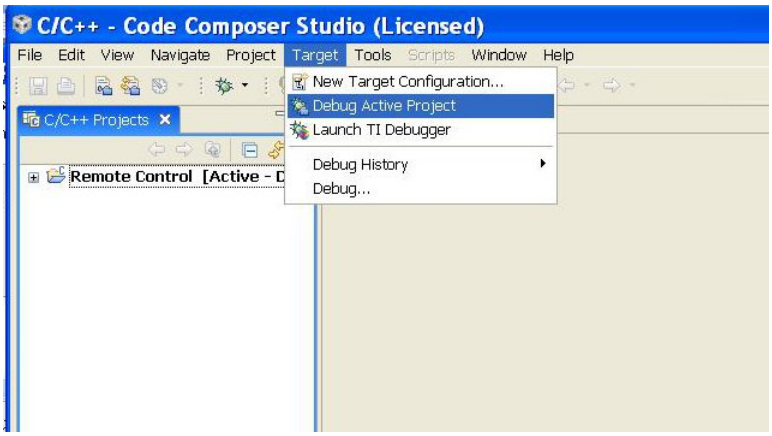


Figure 17. Compile and Debug the Project

5 Circuit Diagram

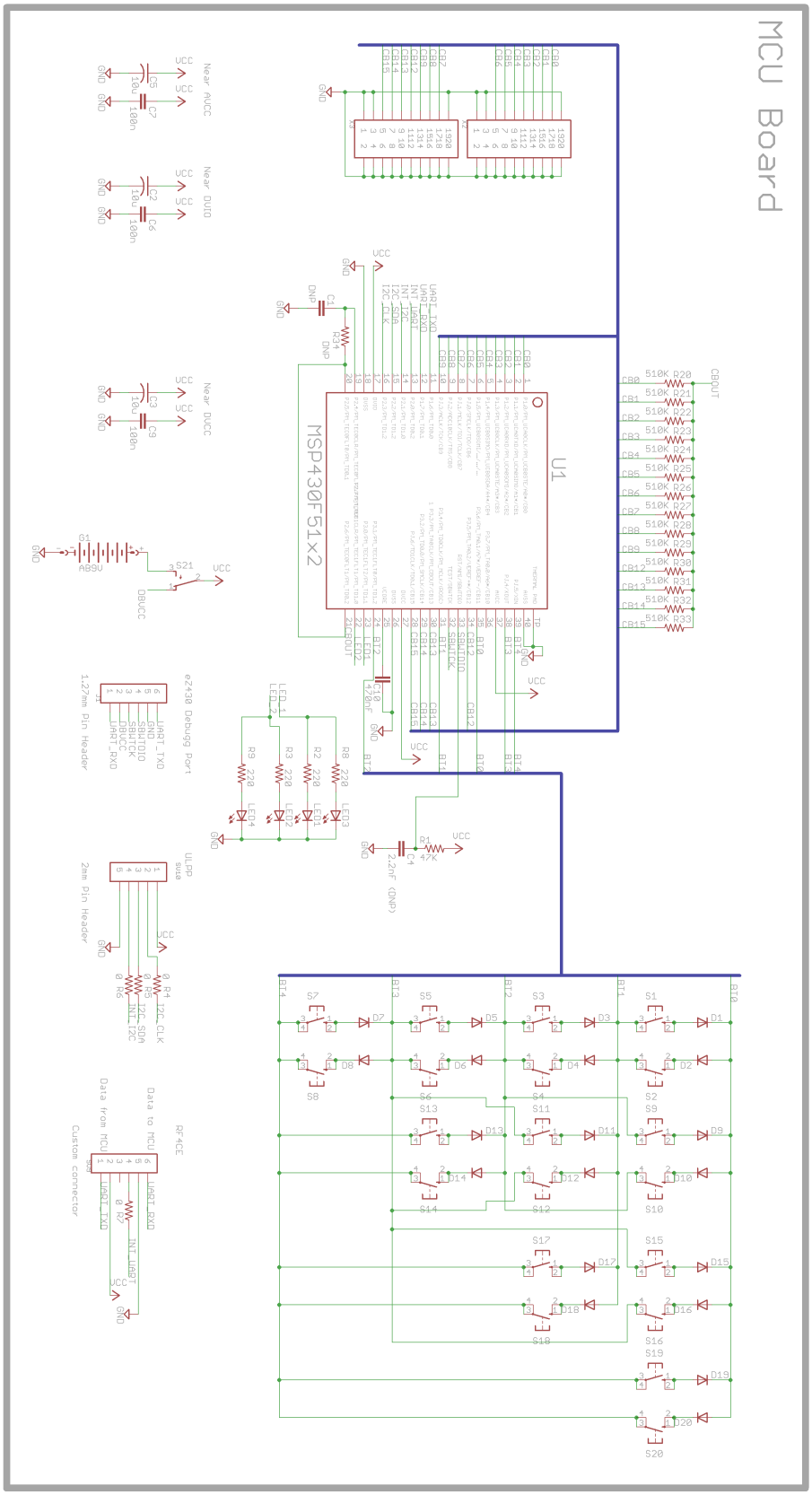


Figure 18. Schematic (MCU board)

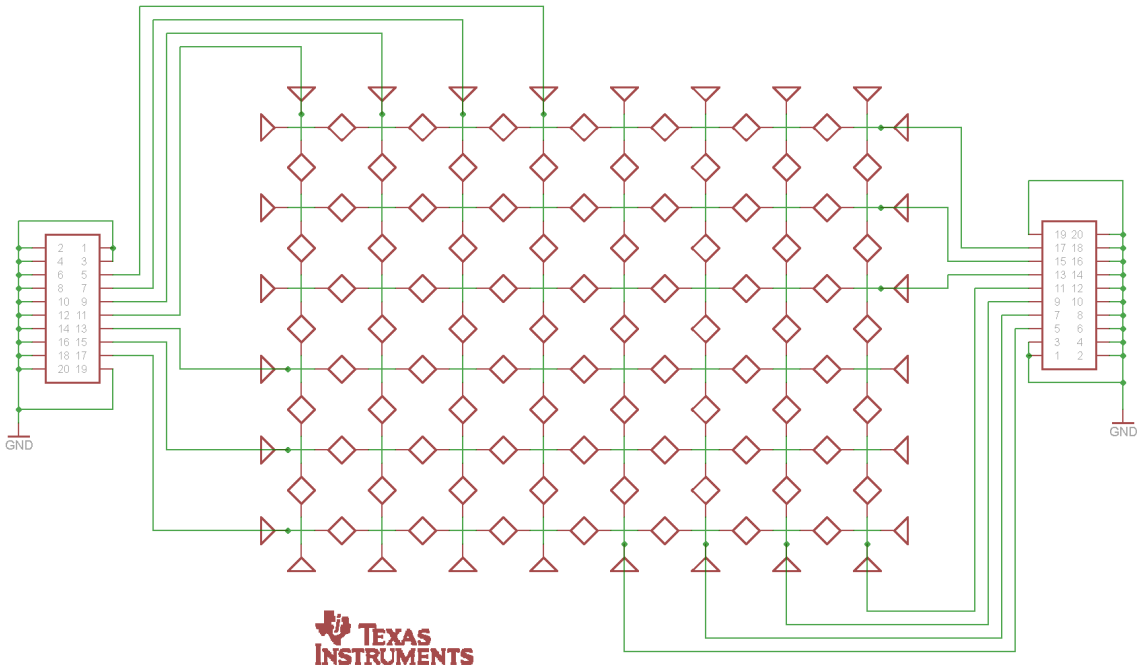


Figure 19. Schematic (Touch pad)

Figure 20. The Remote Control and the USB Receiver

6 References

1. *PCB-Based Capacitive Touch Sensing With MSP430* (SLAA363)
2. *MSP430 Capacitive Single-Touch Sensor Design Guide* (SLAA379)
3. *MSP430F51x1, MSP430F51x2 Mixed Signal Microcontroller Data Sheet* (SLAS619)
4. *MSP430x5xx and MSP430x6xx Family User's Guide* (SLAU208)

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