Application Report
SLAA384A–February 2008–Revised November 2010

4-Wire and 8-Wire Resistive Touch-Screen Controller Using the MSP430™

Neal Brenner, Shawn Sullivan
William Goh
Microcontroller Field Applications
MSP430 Applications

ABSTRACT

This application report describes how to use an MSP430™ microcontroller to drive and read a resistive touch screen. The hardware and software solutions provided enable the reading of user input through a 4-wire or 8-wire resistive touch screen with a low-cost low-power customizable microcontroller. The reference design includes MSP430 software and hardware schematics along with a demonstration PC application. The design was implemented using an MSP430F2012, but it can easily be modified to use any other MSP430 with an ADC.

Contents

List of Figures

List of Tables

MSP430 is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Principles of Operation

1.1 Resistive Touch-Screen Concept

A resistive touch screen is constructed with two transparent layers coated with a conductive material stacked on top of each other. When pressure is applied by a finger or a stylus on the screen, the top layer makes contact with the lower layer. When a voltage is applied across one of the layers, a voltage divider is created. The coordinates of a touch can be found by applying a voltage across one layer in the Y direction and reading the voltage created by the voltage divider to find the Y coordinate, and then applying a voltage across the other layer in the X direction and reading the voltage created by the voltage divider to find the X coordinate.

1.2 Detecting a Touch

To know if the coordinate readings are valid, there must be a way to detect whether the screen is being touched or not. This can be done by applying a positive voltage (V_{CC}) to Y+ through a pullup resistor and applying ground to X–. The pullup resistor must be significantly larger than the total resistance of the touch screen, which is usually a few hundred ohms. When there is no touch, Y+ is pulled up to the positive voltage. When there is a touch, Y+ is pulled down to ground as shown in Figure 1. This voltage-level change can be used to generate a pin-change interrupt.

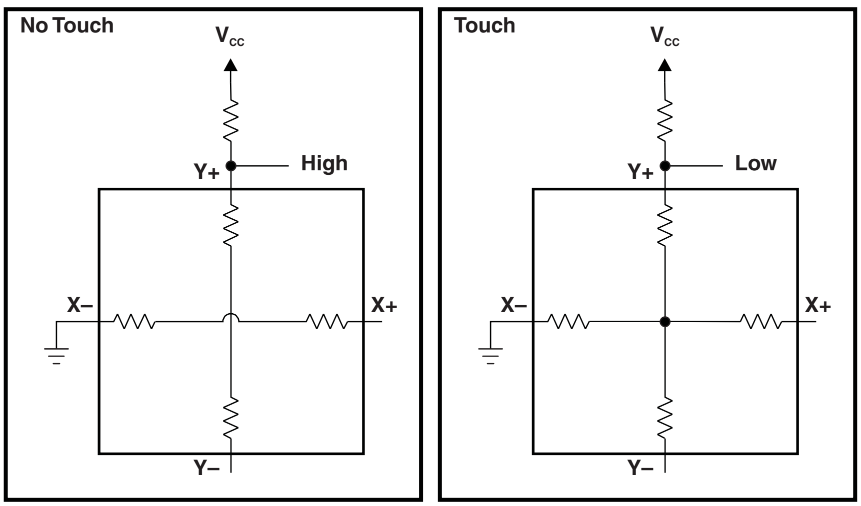

Figure 1. Touch Detection

1.3 Reading a 4-Wire Screen

A 4-wire resistive touch screen is constructed as shown in Figure 2.

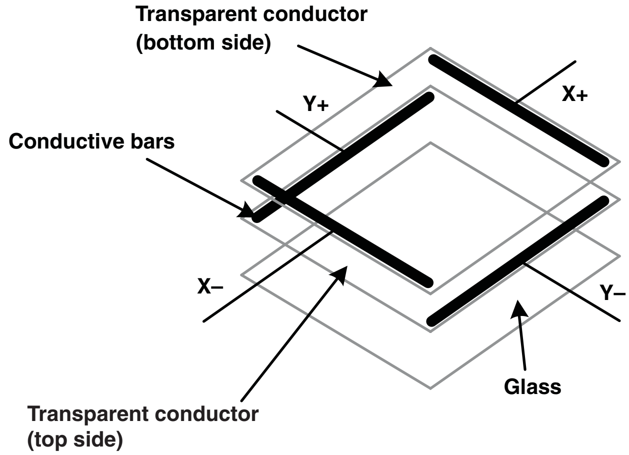

Figure 2. 4-Wire Touch-Screen Construction

The x and y coordinates of a touch on a 4-wire touch screen can be read in two steps. First, Y+ is driven high, Y– is driven to ground, and the voltage at X+ is measured. The ratio of this measured voltage to the drive voltage applied is equal to the ratio of the y coordinate to the height of the touch screen. The y coordinate can be calculated as shown in Figure 3. The x coordinate can be similarly obtained by driving X+ high, driving X– to ground, and measuring the voltage at Y+. The ratio of this measured voltage to the drive voltage applied is equal to the ratio of the x coordinate to the width of the touch screen. This measurement scheme is shown in Figure 3.

$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen} \qquad x = \frac{V_{Y+}}{V_{Drive}} \times width_{screen}$$

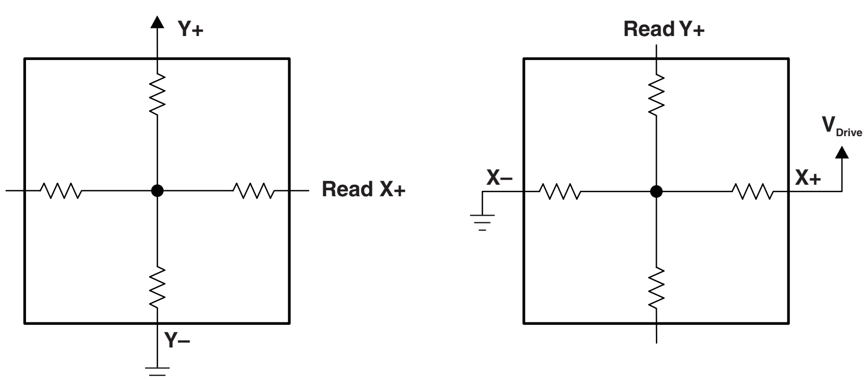

Figure 3. 4-Wire Touch Coordinate Reading

1.4 Reading an 8-Wire Screen

An 8-wire resistive touch screen is constructed as shown in Figure 4.

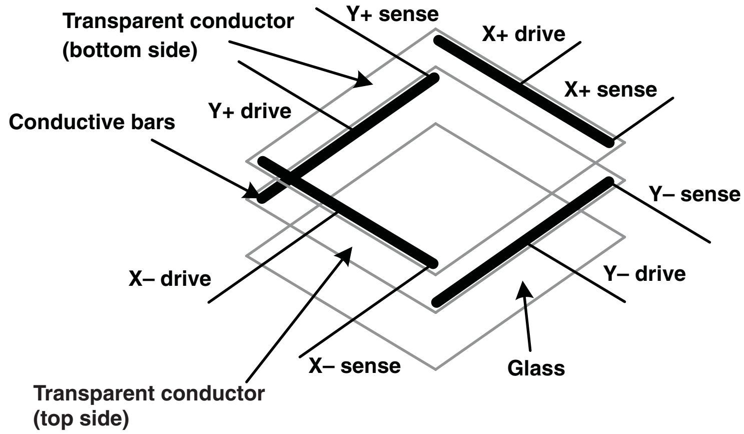

Figure 4. 8-Wire Touch-Screen Construction

In comparison to a 4-wire touch screen, an 8-wire touch screen adds sense wires to the end of each of the conductive bars. This allows any voltage offset created by the wiring or drive circuitry to be calibrated out during operation.

An 8-wire touch screen is calibrated by measuring voltage extremes on either coordinate. First, Y+ drive is driven high and Y– drive is driven low. The corresponding voltages measured at Y+ sense and Y– sense are denoted V_{Ymax} and V_{Ymin}. A similar procedure yields V_{Xmax} and V_{Xmin}. These are the maximum and minimum possible voltages across each coordinate.

The coordinates of a touch on an 8-wire touch screen can be read by first driving Y+ drive high, driving Y– drive to ground, and reading the voltage at X+ sense. Using the maximum and minimum results obtained during calibration, the y coordinate can be calculated as shown in the equations in Figure 5. The x coordinate can be obtained by driving X+ drive high, driving X– drive to ground, and reading the voltage at Y+ sense. This process is shown in Figure 5.

$$y = \frac{\left(\dfrac{V_{X+} - V_{Ymin}}{V_{Ymax} - V_{Ymin}}\right)}{V_{Drive}} \times \text{height}_{screen} \qquad x = \frac{\left(\dfrac{V_{Y+} - V_{Xmin}}{V_{Xmax} - V_{Xmin}}\right)}{V_{Drive}} \times \text{width}_{screen}$$

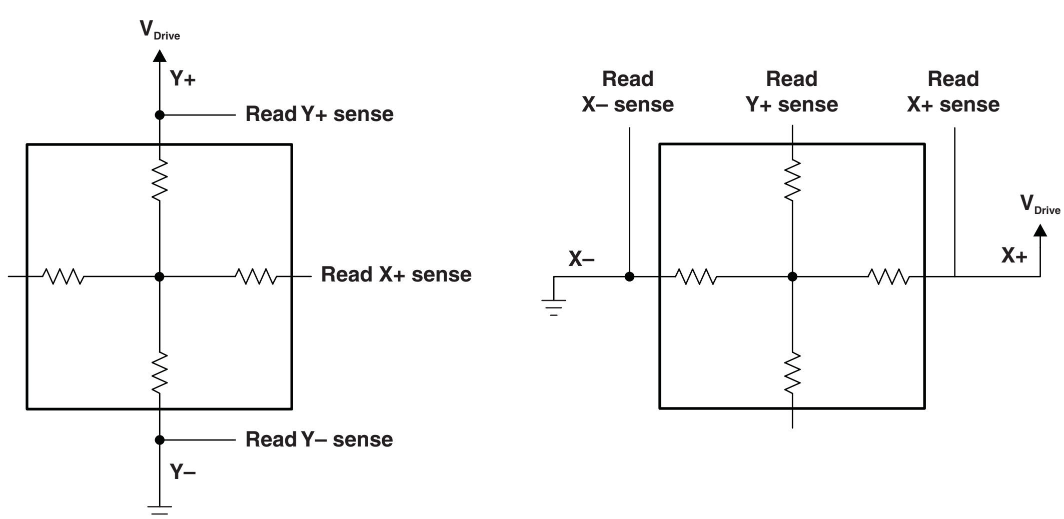

Figure 5. 8-Wire Touch Coordinate Reading

2 MSP430 Implementation

2.1 Solution Overview

Any MSP430 device with an ADC can be used as a precise, flexible, and low-power touch-screen controller. This application report provides a hardware and software proposal based on the MSP430F20x2 to implement both a 4-wire and 8-wire touch-screen controller. Both solutions wait in LPM4 (~0.1 μA) for the screen to be touched, and then transmit the coordinates of the touch over UART or I^2C. When the screen is no longer touched, the MSP430F20x2 is put back into LPM4. Table 1 shows the MSP430 resources used for each solution.

Table 1. Resources Used

Touch Screen	Communication	GPIO	Flash (bytes)	RAM (bytes)	Other
4 wire	UART	5	878	74	Two channels of ADC10, TACCR1
8 wire	UART	9	1178	82	Four channels of ADC10, TACCR1
4 wire	I^2C	6	992	75	Two channels of ADC10, USI
8 wire	I^2C	10	1292	83	Four channels of ADC10, USI

2.2 Detecting a Touch With the MSP430

The MSP430 2xx devices have programmable internal pullup and pulldown resistors on all GPIO pins. This feature can be used to detect a touch on the screen. The MSP430 GPIO pins connected to pins Y+ (Y+ drive for 8-wire screens) and X– (X– drive for 8-wire screens) from the touch screen are used. The X– pin is set to output low. The Y+ pin is set to input with the internal pullup resistor enabled as shown in Figure 6. The Y+ pin can be sampled to determine whether the screen is being pressed. If it is high, the screen is not being pressed; if it is low, the screen is being pressed. A high-to-low port pin interrupt can also be used to enable the MSP430 to sleep in LPM4 while waiting for a touch on the screen.

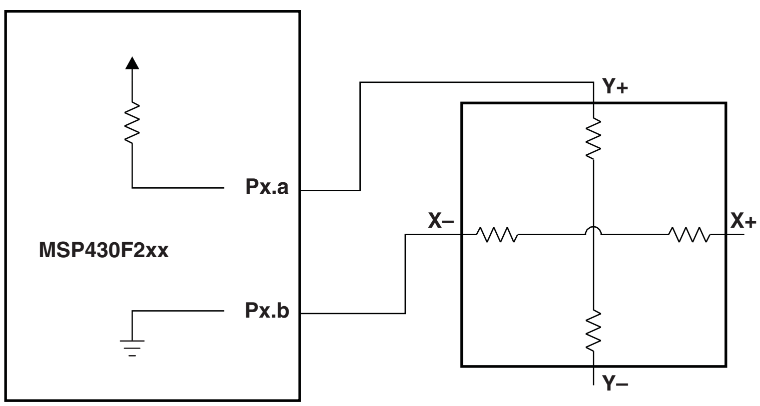

Figure 6. MSP430 Touch Detection Connections

2.3 Hardware

The schematic for the demonstration hardware is shown in Figure 7. The demonstration hardware was designed to interface to any of the Touch International (http://www.touchinternational.com/) 8-wire resistive touch screens (TI-8 Touch).

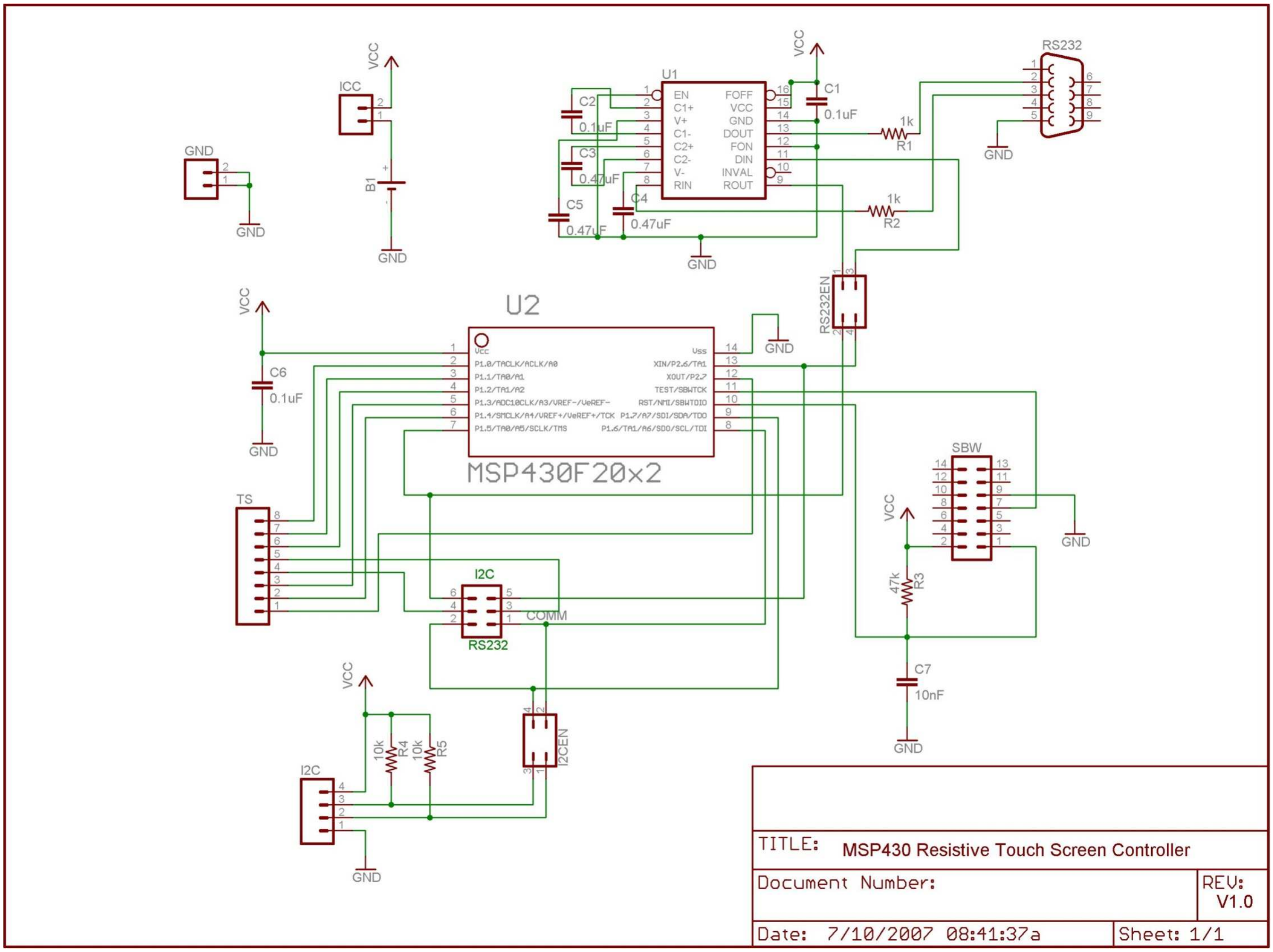

Figure 7. Schematic

2.4 Software

The software flow charts for the 4-wire and 8-wire software are shown in Figure 8.

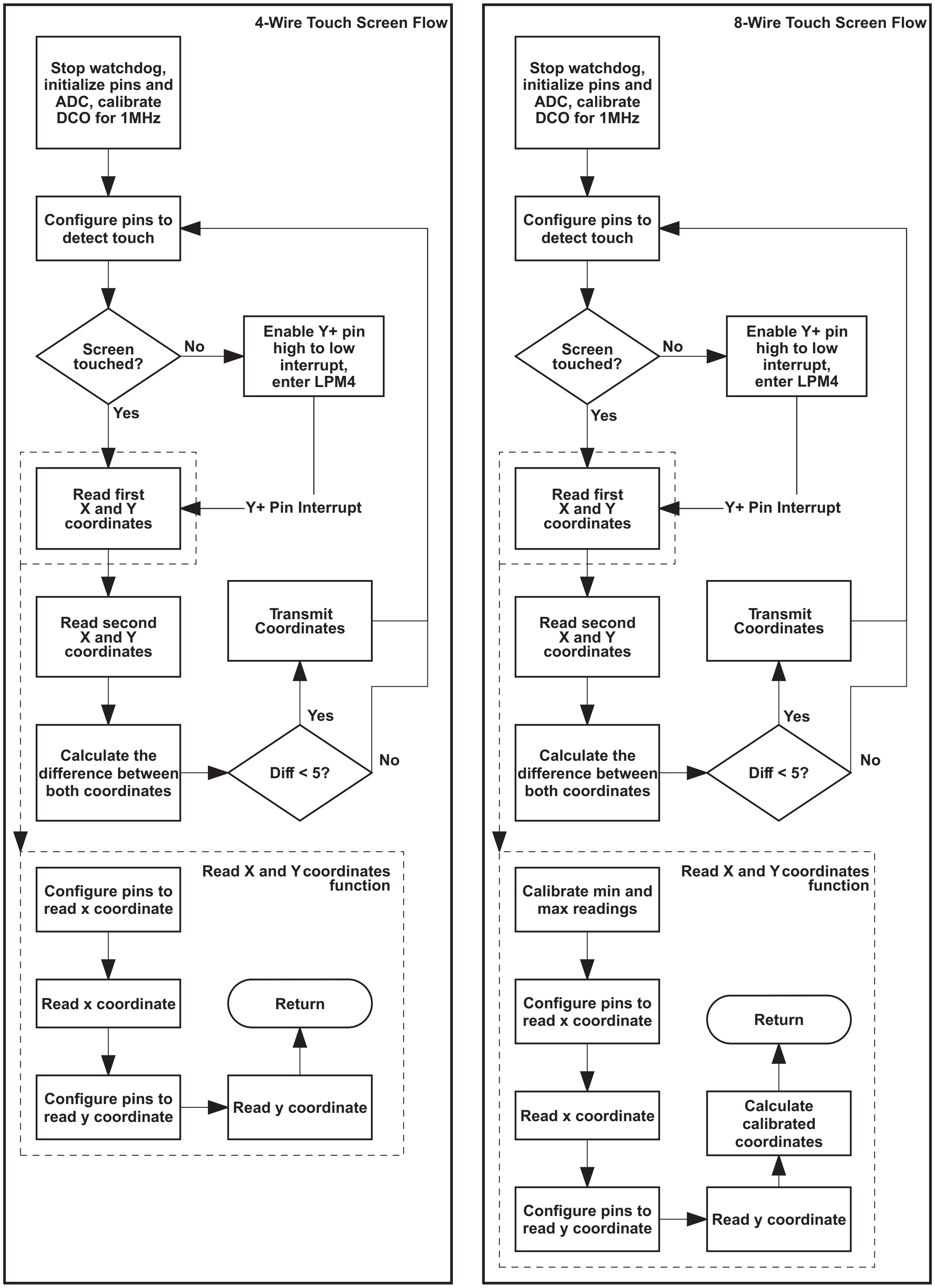

Figure 8. Software Flow Charts

2.4.1 Coordinate Reading

Once a touch is detected, the MSP430 reads the x and y coordinates of the touch. For 8-wire screens, first the minimum and maximum x and y values are read to calibrate the screen.

To read the coordinates, the pins are configured as previously described to drive the X lines and set the Y sense pin as analog ADC inputs. Four ADC conversions are taken and averaged together to obtain the x coordinate reading. For 8-wire screens, this value is calibrated as previously described. The same process is repeated to obtain the y coordinate.

2.4.2 Communication Details

After a touch coordinate is read, it is transmitted back to a host. The zip file associated with this document has example code using master I^2C or an 8N1 9600 baud UART. The Timer_A module is used to implement the UART transmit function as described in *Implementing a UART Function With Timer_A3* (SLAA078).

Each coordinate is sent in a packet of four bytes. The first byte is a synch/control byte. The MSB is always 1 to differentiate it from the data bytes. If the screen is still being touched, a value of 0xFF is sent as the control byte to indicate that the data following is valid. If the screen is not still being touched, a value of 0x80 is sent to indicate that the data following is not valid, and this is the final pair of coordinates for the touch. The 10-bit x and y coordinates are split and packed into three data bytes. The three MSBs of the x and y are packed together in the first byte, and the seven LSBs of the x and y coordinates are sent in the third and fourth, respectively. The communication format is shown in Figure 9.

Control Byte
0xFF (touch) or 0x80 (no touch)

Data Byte 1							
0	Y9	Y8	Y7	0	X9	X8	X7

Data Byte 2							
0	X6	X5	X4	X3	X2	X1	X0

Data Byte 2							
0	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Figure 9. Communication Format

3 Demonstration System

To demonstrate the hardware and software solution proposed in this document, a PC application is provided in the accompanying zip file. This section discusses how to setup and run the demonstration.

3.1 Hardware Setup

The demonstration software uses the RS232 port of a PC, so the 4-wire or 8-wire RS232 software must be loaded into the hardware. At the beginning of the file named main.c, set the WIRES #define to match the number of wires on your touch screen. The COMM #define must be set to UART. The configuration for an 8-wire demonstration is:

```
// Define # of wires, 8 or 4 #define WIRES 8 // Define communication interface, I2C
or UART #define COMM UART
```

To enable RS232 communication on the board, five jumpers must be put in place as shown by the red boxes in Figure 10. There should be no jumpers connected on the I2CEN header.

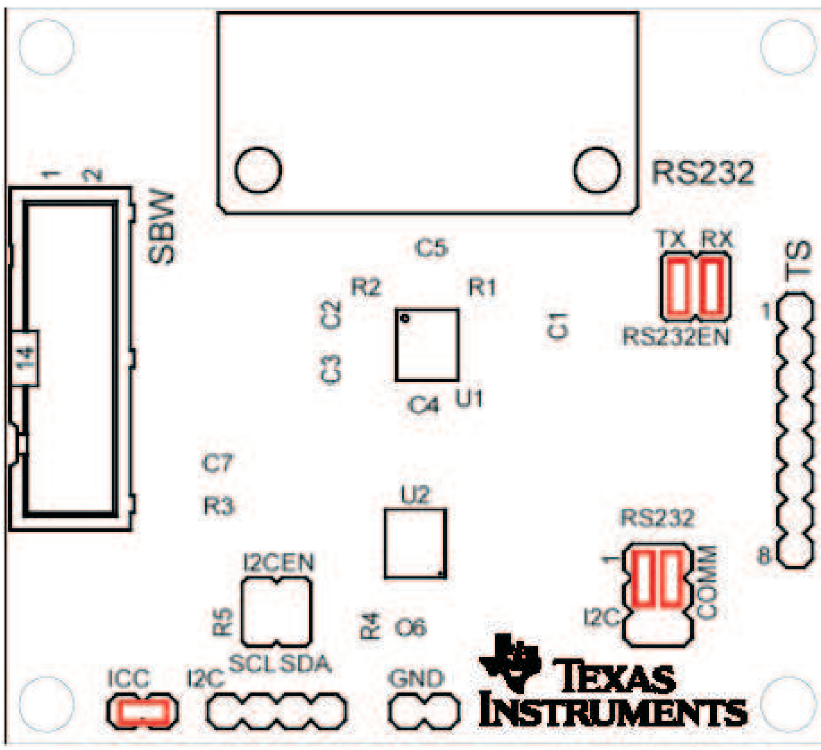

Figure 10. Hardware Setup

The touch screen should be connected to the TS header, and an RS232 cable should be connected between the PC COM1 port and the RS232 connector on the board. Finally, the board should be powered by 1.8 V to 3.6 V, either through the JTAG header or by a CR2032 battery on the back of the board.

3.2 PC Software

The PC application is provided in the associated zip file. Run setup.exe to install the software. The setup program may download and install additional components required by the software. Once the proper software is loaded on the board and the hardware is configured as previously described, launch the PC software that was just installed called Touch-Screen Reader. A window similar to the one shown in Figure 11 appears.

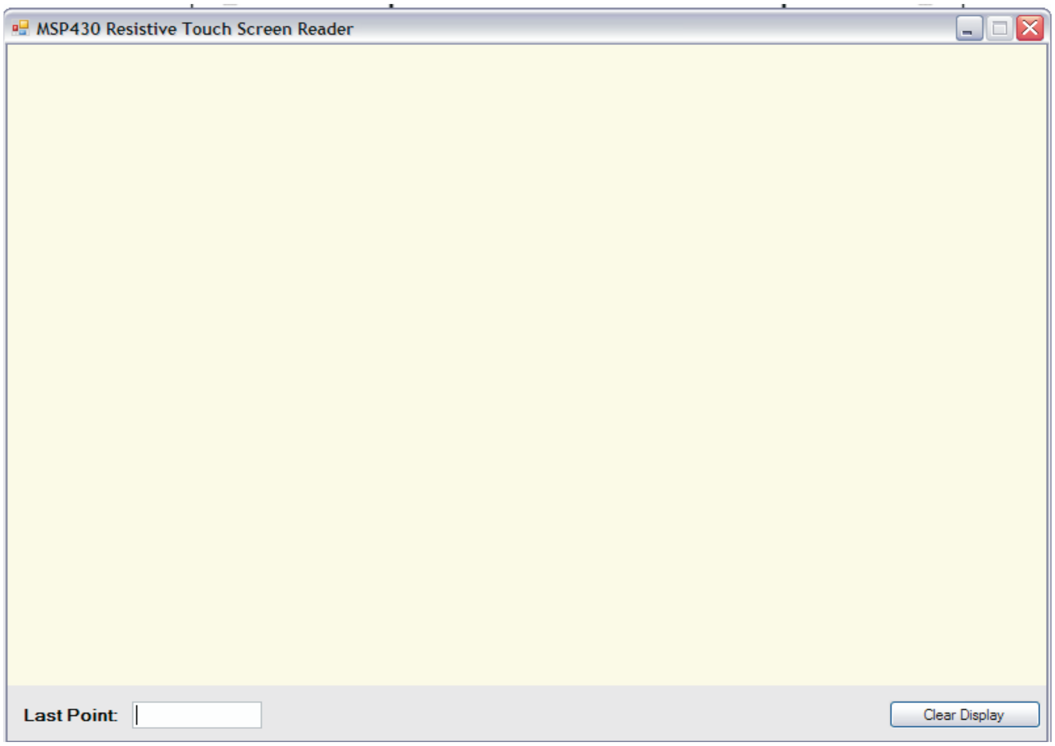

Figure 11. PC Software

The PC software opens COM1 and a connection to the board. As you touch the touch screen, the software draws the points you touch, and the coordinate of the last point are displayed in the text box at the bottom of the application. The Clear Display button at the bottom of the application clears the previous points drawn in the window.

4 References

1. MSP430x20x1, MSP430x20x2, MSP430x20x3 data sheet (SLAS491)
2. *MSP430x2xx Family User's Guide* (SLAU144)
3. *Implementing a UART Function With Timer_A3* (SLAA078)

Revision History

Changes from Original (February 2008) to A Revision **Page**

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.