

# URTouch

Universal TFT resistive touchscreen library

## Manual

The logo for Rinky-Dink Electronics features the company name in a stylized, glowing cyan font with a 3D effect. The text is set against a dark background that includes a close-up image of a green printed circuit board (PCB) with various electronic components and traces visible.

Rinky-Dink Electronics

## Introduction:

This library was made to complement UTFT to provide touch screen functionality.

---

You can always find the latest version of the library at <http://www.RinkyDinkElectronics.com/>

For version information, please refer to **version.txt**.

## REGARDING CALIBRATION:

All touch screens will have slight variations. It is therefore important that you calibrate your particular touch screen for the best possible performance. The default calibration data supplied with the library *may* work on your screen, but **only if** your screen has a 320x240 resolution. Screens with other resolutions **MUST** be calibrated.

To calibrate your touch screen you will need to run the **URTouch\_Calibration** sketch supplied in the examples of the library.

Before you compile and upload the sketch there are a couple of things you must do.

1. Make sure you have uncommented the correct section for your development board
2. Make sure the UTFT display model code is correct for your display module
3. Make sure the TOUCH\_ORIENTATION define is correct. You can find a list of the correct parameter for all the tested displays in the ***URTouch\_Supported\_display\_modules*** PDF.

Further instructions will be given on screen when you run the sketch.

Remember that if you have more than one touch display module you may have to run the calibration on each module.

An on-line tool to verify your calibration data can be found at [http://www.RinkyDinkElectronics.com/t\\_cal\\_verify.php](http://www.RinkyDinkElectronics.com/t_cal_verify.php)

*Some touch screens, especially the larger ones (4.3" and larger), have some flaws where they have problems registering touch near the edges. The calibration sketch tries to take this into account when calibrating. Because of this some calibration points may take longer to register.*

It is also recommended that you power your development board using an external power source when running the calibration on 4.3" and larger screens.

## Defined Literals:

Orientation
For use with InitTouch()
PORTRAIT: 0
LANDSCAPE: 1

Precision
For use with setPrecision()
PREC_LOW: 1
PREC_MEDIUM: 2
PREC_HI: 3
PREC_EXTREME: 4

## Functions:

### **URTouch(TCLK, TCS, TDIN, TDOUT, IRQ);**

The main class of the interface.

Parameters:     TCLK: Pin for Touch Clock (D\_CLK)  
                  TCS: Pin for Touch Chip Select (D\_CS)  
                  TDIN: Pin for Touch Data input (D\_DIN)  
                  TDOUT: Pin for Touch Data output (D\_OUT)  
                  IRQ: Pin for Touch IRQ (DPenirq)  
Usage:           URTouch myTouch(15,10,14,9,8); // Start an instance of the URTouch class

### **InitTouch([orientation]);**

Initialize the touch screen and set display orientation. If the library is used together with UTFT the orientation should be set to the same orientation for both libraries.

Parameters:     orientation: <optional>  
                                PORTRAIT  
                                LANDSCAPE (default)  
Returns:        Nothing  
Usage:           myTouch.InitTouch();// Initialize the touch screen

### **dataAvailable();**

Check to see if new data from the touch screen is waiting.

Parameters:     None  
Returns:        Boolean: true means data is waiting, otherwise false  
Usage:           check = myTouch.dataAvailable() // See if data is waiting

### **read();**

Read waiting data from the touch screen. This function should be called if dataAvailable() is true. Use getX() and getY() to get the coordinates.

Parameters:     None  
Returns:        Nothing  
Usage:           myTouch.read(); // Read data from touch screen  
Notes:          After calling read(), raw data from the touch screen is available in the variables TP\_X and TP\_Y. Do not use these if you do not know how to handle the raw data. Use getX() and getY() instead.

### **getX();**

Get the x-coordinate of the last position read from the touch screen.

Parameters:     None  
Returns:        Integer  
Usage:           x = myTouch.getX(); // Get the x-coordinate

### **getY();**

Get the y-coordinate of the last position read from the touch screen.

Parameters:     None  
Returns:        Integer  
Usage:           y = myTouch.getY(); // Get the y-coordinate

### **setPrecision(precision);**

Set the precision of the touch screen.

Parameters:     precision: **PREC\_LOW**, **PREC\_MEDIUM**, **PREC\_HI**, **PREC\_EXTREME**  
Returns:        Nothing  
Usage:           myTouch.setPrecision(PREC\_MEDIUM); // Set precision to medium  
Notes:          Higher precision data will take longer to read, so take care when using PREC\_HI or PREC\_EXTREME with fast-moving input.