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 UTouch_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
UTouch_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

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 Arduino/chipKit using an external power source
when running the calibration on 4.3” and larger screens.
Defined Literals:

<table>
<thead>
<tr>
<th>Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>For use with InitTouch()</td>
</tr>
<tr>
<td>PORTRAIT: 0</td>
</tr>
<tr>
<td>LANDSCAPE: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>For use with setPrecision()</td>
</tr>
<tr>
<td>PREC_LOW: 1</td>
</tr>
<tr>
<td>PREC_MEDIUM: 2</td>
</tr>
<tr>
<td>PREC_HI: 3</td>
</tr>
<tr>
<td>PREC_EXTREME: 4</td>
</tr>
</tbody>
</table>
Functions:

**UTouch(TCLK, TCS, TDIN, TDOUT, IRQ);**

The main class of the interface.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCLK:</td>
<td>Pin for Touch Clock (D_CLK)</td>
</tr>
<tr>
<td>TCS:</td>
<td>Pin for Touch Chip Select (D_CS)</td>
</tr>
<tr>
<td>TDIN:</td>
<td>Pin for Touch Data input (D_DIN)</td>
</tr>
<tr>
<td>TDOUT:</td>
<td>Pin for Touch Data output (D_OUT)</td>
</tr>
<tr>
<td>IRQ:</td>
<td>Pin for Touch IRQ (D_Penirq)</td>
</tr>
</tbody>
</table>

**Usage:**

```java
myTouch(15,10,14,9,8); // Start an instance of the UTouch 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.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orientation:</td>
<td>&lt;optional&gt; PORTRAIT LANDSCAPE (default)</td>
</tr>
</tbody>
</table>

**Returns:**

Nothing

**Usage:**

```java
myTouch.InitTouch(); // Initialize the touch screen
```

**dataAvailable();**

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns:**

Boolean: true means data is waiting, otherwise false

**Usage:**

```java
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.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns:**

Nothing

**Usage:**

```java
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.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns:**

Integer

**Usage:**

```java
x = myTouch.getX(); // Get the x-coordinate
```

**getY();**

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

**Returns:**

Integer

**Usage:**

```java
y = myTouch.getY(); // Get the y-coordinate
```

**setPrecision(precision);**

Set the precision of the touch screen.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision:</td>
<td>PREC_LOW, PREC_MEDIUM, PREC_HI, PREC_EXTREME</td>
</tr>
</tbody>
</table>

**Returns:**

Nothing

**Usage:**

```java
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.