touchIt 1.0 User Guide

Version 1.1 28/06/2016

Contents

Version 1.0 (01/06/16) – Initial release.
Version 1.1 (28/06/16) – Command 0x02 “Set Address” missing from documentation.

Introduction

The touchIt device is a compact sensor based on capacitive touch matrix technology. Designed to enable sensitive position data for such things as robotic end effectors, miniature track pads and wearable gesture interfaces, it features:

1) Fine touch and tap detection with a resolution of up to 400dpi

2) Coarse area touch and tap detection of up to 16 locations

3) Interrupt output configurable per touch type

4) LED touch indicator

5) Individually addressable

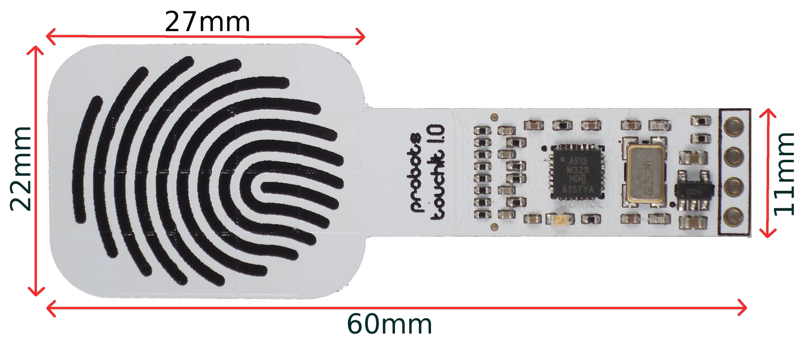
6) 3.3V / 5V operation @ < 5mA draw.

Interrupt detection of changes in each mode is enabled with a configuration command sent from an I2C master device. Fine (x/y) or Coarse (0-15) position data is retrieved from the I2C slave either through a polling loop or event-driven using the sensor's interrupt output, indicating a touch has occurred.

Thank you for using touchIt!

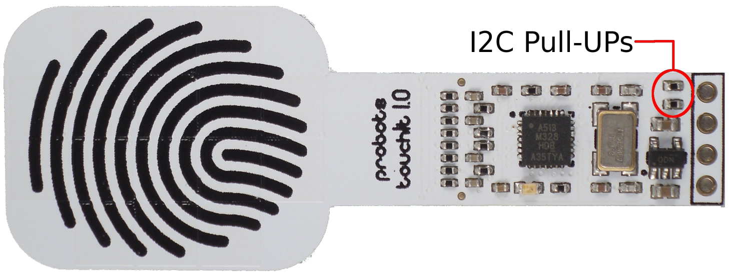
Dimensions

The overall dimensions for touchIt are 60mm x 22mm x 0.4mm



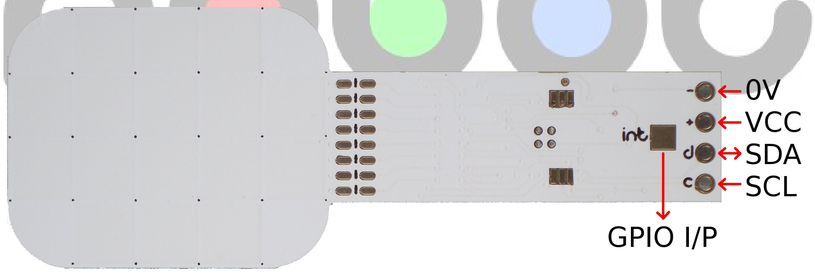
Wiring

The touchIt device can be controlled by anything which has a spare I2C port. Connect I2C serial clock (SCL) and serial data (SDA) to the pins labelled 'c' and 'd' respectively. There are 4K7 pull-up resistors on the SCL and SDA lines on touchIt, so no external pull-ups are required on the control device; however these will become weaker when more devices are placed on the bus so may need to be removed if the number is too great.

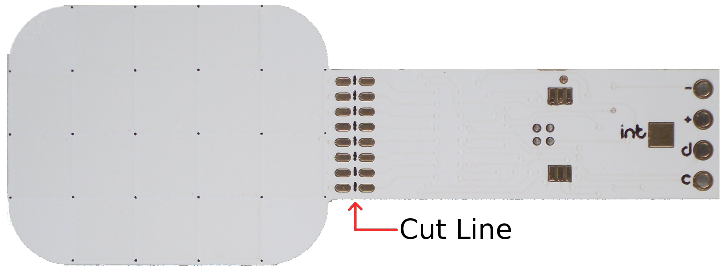


Power to the device can be 3.3V or 5V and its positive and negative rails should be connected to pins '+' and '-'.

To the rear of the device is a pad labelled 'int' or Interrupt; if required, this should be connected to a spare GPIO input on the host device. It is driven high/low so no external pull up/down resistors are required.



It is possible to relocate the sense elements from the control circuit by cutting along the dashed line at the rear of the device with scissors or a scalpel. This may be desirable in an application where space is limited. An 8-way 1.27mm pitch ribbon cable (recommended) should then be soldered to the pads either side of the separation to reconnect the circuit. Keep the cable short and avoid loose or twisted wires which could cross-couple signals and reduce performance.



Programming

All programming is done over an I2C interface. The touchIt sensor is configured as an I2C Slave device and requires an I2C Master device to retrieve data at a recommended bitrate of 100kHz. This can be accomplished using the slave's interrupt output to trigger a read, or through a less-efficient continuous poll loop.

There are 5 commands available to the I2C Master device. All commands contain a Cyclic-Redundancy Check byte (CRC) to improve immunity over long wire lengths and prevent erratic behaviour.

I2C Format: <I2C START><Transmit><I2C REPEATED START><Receive><I2C STOP>

Transmit format (Master->Slave)
<'#'><CMD><LEN><DATA><CRC>

<'#'>	Start of transmit command
<CMD>	Command number
<LEN>	Length of data bytes to send
<DATA>	Data bytes to send (if LEN > 0)
<CRC>	Generated CRC

Receive format (Slave->Master)
<'$'><CMD><LEN><DATA><CRC>

<'$'>	Start of receive command
<CMD>	Command number
<LEN>	Length of data bytes to send
<DATA>	Data bytes to send (if LEN > 0)
<CRC>	Check CRC

The CRC is generated with an XOR of all bytes except the CRC itself. The method below is used to generate and validate outgoing and incoming packets respectively:

Transmitted "Set Config" Bytes: <'#'><0x01><0x01><0x03><0x20>

```
CRC ^= '#';      (CRC = 0x23)
CRC ^= 0x01;     (CRC = 0x22)
CRC ^= 0x01;     (CRC = 0x23)
CRC ^= 0x03;     (CRC = 0x20) – Set as transmitted 5th byte
```

Received "Set Config" Bytes: <'$'><0x01><0x01><0x03><0x27>

```
CRC ^= '$';      (CRC = 0x24)
CRC ^= 0x01;     (CRC = 0x25)
CRC ^= 0x01;     (CRC = 0x24)
CRC ^= 0x03;     (CRC = 0x27) – Check against received 5th byte
```

Command Set

0x00 – Get Version

Returns the touchIt Firmware Version.

Out: <'#'><0x00><0x00><CRC>

In: <'$'><0x00><0x02><0x01><0x00><CRC>

The data received above indicates the firmware is at version 1.0

0x01 – Set Configuration

Configure the events which cause the interrupt output to pull low. The return value will be the configuration byte if successfully stored in non-volatile eeprom, zero otherwise.

Out: <'#'><0x01><0x01><CFG><CRC>

In: <'$'><0x01><0x01><CFG><CRC>

The Configuration (CFG) byte can be one or a combination (logical OR) of four types, but fine and coarse events cannot be mixed (0x01 + 0x04 or 0x02 + 0x04):

0x01 (XMOVE):	Int output low when a change is detected in the Fine X axis.
0x02 (YMOVE):	Int output low when a change is detected in the Fine Y axis.
0x04 (TMOVE):	Int output low when a change is detected in the Coarse area.
0x08 (TAP):	Int output low when a tap is detected in either Fine of Coarse modes.

0x02 – Set Address

Set a unique 7-bit address for touchIt to allow multiple units to then share the same I2C bus.

Out: <'#'><0x02><0x01><ADD><CRC>
In: None, unit resets after address change, and master then uses the new address.

Address must be between 1 and 127.

0x03 – Get Position (Fine mode)

Retrieve X/Y coordinates at the point of touch following an interrupt event or via direct polling. Suitable Interrupt configurations are XMOVE, YMOVE, TAP.

Out: <'#'><0x03><0x00><CRC>

In: <'$'><0x03><0x04><X MSB><X LSB><Y MSB><Y LSB><CRC>

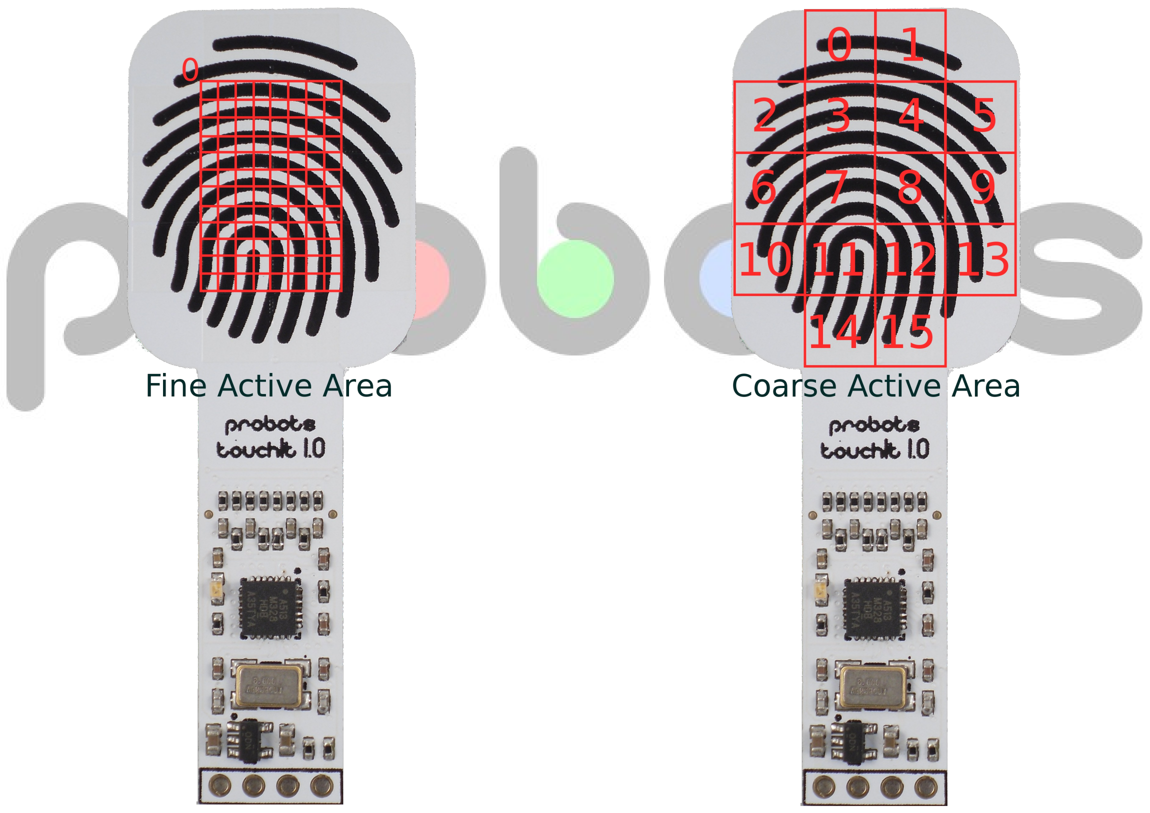
Fine X/Y position data is obtained from the sensor by averaging the measurements from adjacent capacitive plates, so the active area is smaller than Coarse mode but has a much greater resolution.

0x04 – Get Touch (Coarse mode)

Retrieve the numbered area at the point of touch following an interrupt event or via direct polling. Suitable Interrupt configurations are TMOVE, TAP.

Out:	<'#'><0x04><0x00><CRC>
In:	<'$'><0x04><0x01><AREA><CRC>

Coarse position data is obtained from the magnitude of the measurements taken at each individual capacitive plate, therefore the active areas are as shown below:


0
Fine Active Area
probots touchit 1.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Coarse Active Area
probots touchit 1.0