

**4D Systems**

---

## **Application Note: 4D-AN-P2003**

# **Touch Detection on PICASO Modules**

Document Date: 15<sup>th</sup> December 2012

Document Revision: 1.0

---

## Description

This Application Note is dedicated to explaining how to use the 4-Wire Resistive Touch functionality on any PICASO 4D module. In order to carry out this application, the following items are required:

- Any 4D PICASO Screen Module
- 4D Programming Cable
- 4DWorkshop4 IDE Software Tool

---

## Application Overview

A dominant standout feature of certain 4D display modules is their ability to enable touch detection. This application note is a step by step procedure that explains the necessary coding practices required to enable and utilise the touch abilities on selected PICASO display modules.



---

## Setup Procedure

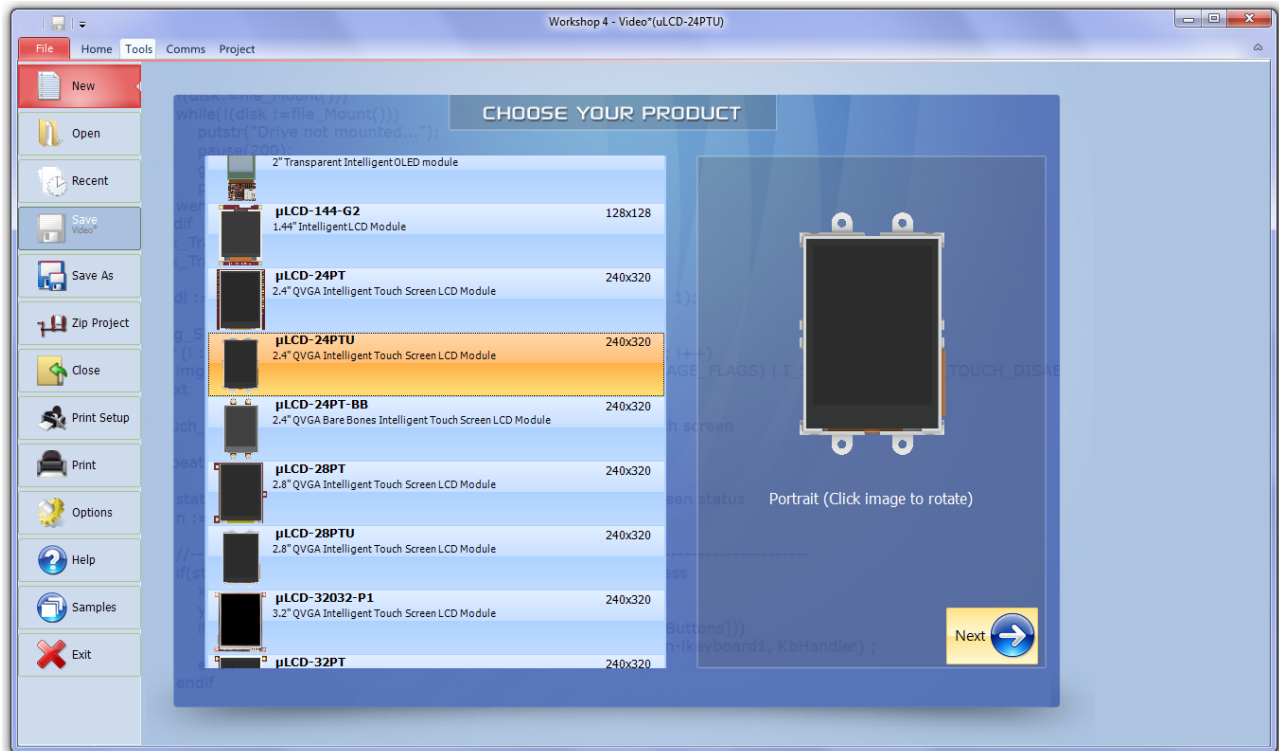
Firstly, you will need to download the 4DWorkshop4 IDE environment. This is where the end user application is developed and can be found from the 4D Systems website below:

<http://www.4dsystems.com.au/>

## Simulation Procedure

### Initial Setup

Click "New" in the 4DWorkshop4 IDE, choose the product and Designer Environment,



### Enable Touch Detection

Layout the necessary scaffold for the program, which contains the following essential functions:

```
#platform "uLCD-24PTU"

#inherit "4DGL_16bitColours.fnc"

func main()

    gfx_ScreenMode(PORTRAIT) ; // change manually if orientation
    change

    print("Hello World") ; // replace with your code

    repeat // maybe replace
    forever // this as well

endfunc
```

At the start of the program, insert the following function to enable the Touch feature:

```
touch_Set(TOUCH_ENABLE);
```

This touch\_Set() function can also be used to disable Touch detection at any stage if required, by changing the inserted variable.

```
touch_Set(TOUCH_DISABLE);
```

## Set a Detection Region

Now that touch has been enabled, it needs to be refined to a specific area of the screen. Generally speaking, applications will require a different action for a touch in certain areas. If this is not so, the entire screen can be used as the touch detection region. It should be noted that setting a specific touch detection region can be done in one of two ways. The following two sections cover the first approach. To set a specific touch region using an explicit command, use the following code:

```
touch_DetectRegion(x1, y1, x2, y2);
```

Insert the desired parameters for the X and Y coordinates of the screen. For example; if a screen with a resolution of 240x320 is being used, then the X and Y figures must lie within these bounds. The next two lines explicitly show the touch boundaries for detection being set.

```
touch_DetectRegion(0, 0, 240, 320); //enable entire area
```

```
touch_DetectRegion(10, 10, 30, 30); //enable a small 20x20 square
```

An alternative method to enable the entire active area can be done in one command. This method is recommended over manually setting the detection region to the same area as the screen resolution. Use the following command:

```
touch_Set(TOUCH_REGIONDEFAULT);
```

## Detect Touch in a Specific Region

Now that the touch detection area has been set, it needs to be constantly checked for a change in state, or a 'touch'. The status of a touch response is retrieved by using the following command:

```
touch_Get(TOUCH_STATUS);
```

Using the touch\_Get() function returns a value depending on the current state. Integers 0 to 3 or their MACRO equivalents are returned based on the following results:

- 0 = NOTOUCH**
- 1 = TOUCH\_PRESSED**
- 2 = TOUCH\_RELEASED**
- 3 = TOUCH\_MOVING**

In this way, it can be determined when there is activity on the screen.

## Simple Example

The example program below sets a detection region right in the middle of the screen that is a 20x20 square region. This means that all other areas of the screen are disabled.

```
#platform "uLCD-24PTU"

/*****
* Filename: TouchDetectRegion.4dg
* Created: 9th November 2011
* Author: 4D team
* Description: detect a touch for a defined area
*****/

#inherit "4DGL_16bitColours.fnc"

func main()

    touch_Set(TOUCH_ENABLE);

    repeat

        touch_DetectRegion(110,150,130,170); // 20x20 square centre screen

        if(touch_Get(TOUCH_STATUS) == TOUCH_PRESSED)
            gfx_Cls();
            print("There is a Touch!");
            pause(1000);
            gfx_Cls();
            pause(1000);
        endif

    forever

endfunc
```

## Multiple Touch Zones on a Single Display Screen

It can be difficult to use the above method of specifying explicit touch coordinates for a given area if there are multiple points on a screen that require a different action from a touch response. To scan for touch in multiple areas, the entire active area should be enabled and used in conjunction with a series of 'if' statements that each check for touch according to certain coordinates. The following example illustrates this concept. Five targets are drawn on the screen. Touch is then scanned in their locations, which upon activity will display the appropriate message verification.

```
#platform "uLCD-24PTU"

/*****
* Filename: MultiTouchDetect.4dg
* Created: 9th November 2011
* Author: 4D team
* Description: detect a touch for a defined area
*****/

#inherit "4DGL_16bitColours.fnc"

var x,y;

func main()

    touch_Set(TOUCH_ENABLE);
    repeat

        if(touch_Get(TOUCH_STATUS) == TOUCH_PRESSED)
            x := touch_Get(TOUCH_GETX);
            y := touch_Get(TOUCH_GETY);
            if( (x >= 105 && x <= 135) && (y >= 145 && y <= 175) )
                gfx_Cls();
                print("There is a Touch in the centre target!");
                pause(1000);
                gfx_Cls();
            endif
        endif

    forever

endfunc
```

## Complete Example Application

A fully developed application below illustrates detecting touch in multiple regions on the screen using a switch statement for control flow. There are associated print statements and targets displayed on the screen to assist with experimenting with touch coordinate detection.

```
#platform "uLCD-24PTU"

/*****
 * Filename: TouchDetect.4dg
 * Created: 9th November 2011
 * Author: 4D team
 * Description: detect a touch for a defined area
 *****/

#inherit "4DGL_16bitColours.fnc"

var x,y;

func main()

    touch_Set(TOUCH_ENABLE);

    repeat

        gfx_Hline(30,30-12,30+12,WHITE);    //target #1 top left
        gfx_Vline(30,30-12,30+12,WHITE);
        gfx_CircleFilled(30,30,3,WHITE);
        gfx_Circle(30,30,12,WHITE);

        gfx_Hline(30,210-12,210+12,WHITE);    //target #2 top right
        gfx_Vline(210,30-12,30+12,WHITE);
        gfx_CircleFilled(210,30,3,WHITE);
        gfx_Circle(210,30,12,WHITE);

        gfx_Hline(160,120-12,120+12,WHITE);    //target #3 centre
        gfx_Vline(120,160-12,160+12,WHITE);
        gfx_CircleFilled(120,160,3,WHITE);
        gfx_Circle(120,160,12,WHITE);

        gfx_Hline(290,30-12,30+12,WHITE);    //target #4 bottom left
        gfx_Vline(30,290-12,290+12,WHITE);
        gfx_CircleFilled(30,290,3,WHITE);
        gfx_Circle(30,290,12,WHITE);

        gfx_Hline(290,210-12,210+12,WHITE);    //target #5 bottom right
        gfx_Vline(210,290-12,290+12,WHITE);
        gfx_CircleFilled(210,290,3,WHITE);
        gfx_Circle(210,290,12,WHITE);

        if(touch_Get(TOUCH_STATUS) == TOUCH_PRESSED)
            x := touch_Get(TOUCH_GETX);
            y := touch_Get(TOUCH_GETY);
            switch

                case ( (x >= 10 && x <= 40) && (y >= 10 && y <= 40) )
                    gfx_Cls();
                    print("There is a Touch in the top left hand target!");
```

```
                pause(1000);
                gfx_Cls();
            break;

            case( (x >= 200 && x <= 230) && (y >= 10 && y <= 40) )
                gfx_Cls();
                print("There is a Touch in the top right hand target!");
                pause(1000);
                gfx_Cls();
            break;

            case( (x >= 105 && x <= 135) && (y >= 145 && y <= 175) )
                gfx_Cls();
                print("There is a Touch in the centre target!");
                pause(1000);
                gfx_Cls();
            break;

            case( (x >= 10 && x <= 40) && (y >= 280 && y <= 310) )
                gfx_Cls();
                print("There is a Touch in the bottom left hand
target!");
                pause(1000);
                gfx_Cls();
            break;

            case( (x >= 200 && x <= 230) && (y >= 280 && y <= 310) )
                gfx_Cls();
                print("There is a Touch in the bottom right hand
target!");
                pause(1000);
                gfx_Cls();
            break;

        endswitch

    endif

    forever

endfunc
```



## Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

---

## Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.