APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi DIP Switch

DOCUMENT DATE: **21st May 2019**
DOCUMENT REVISION: **1.1**

Description

This Application Note shows how to add and configure a DIP switch, one of the widgets available in Workshop. Before getting started, the following are required:

- Any of the following 4D Picaso display modules:

  gen4-uLCD-24PT gen4-uLCD-28PT gen4-uLCD-32PT
  uLCD-24PTU uLCD-28PTU uVGA-III

  and other superseded modules which support the ViSi Genie environment
- The target module can also be a Diablo16 display

  gen4-uLCD-24D series gen4-uLCD-28D series gen4-uLCD-32D series
  gen4-uLCD-35D series gen4-uLCD-43D series gen4-uLCD-50D series
  gen4-uLCD-70D series
  uLCD-35DT uLCD-43D Series uLCD-70DT

  Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor. The display module used in this application note is the uLCD-32PTU, which is a Picaso display. This application note is applicable to Diablo16 display modules as well.

- 4D Programming Cable / µUSB-PA5/uUSBPA5-II
  for non-gen4 displays (uLCD-xxx)
- 4D Programming Cable & gen4-IB / 4D-UPA / gen4-PA
  for gen4 displays (gen4-uLCD-xxx)
- micro-SD (µSD) memory card
- Workshop 4 IDE (installed according to the installation document)
- Any Arduino board with a UART serial port
- 4D Arduino Adaptor Shield (optional) or connecting wires
- Arduino IDE

When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.

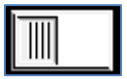
The ViSi-Genie project and the Arduino sketch are provided as examples to help you along this application note.

Content

Application Overview

This application note explains how to configure a DIP switch in the WYSIWYG screen, how to paste the generated code, and how to display the different states. The various orientations and states of a DIP switch are shown below.

DIP Switch

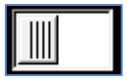
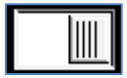
Vertical		Horizontal	
Appearance	**State**	**Appearance**	**State**
	0		0
	1		1

DIP Switch with n Positions

Vertical

Appearance	State
	0
	1
	n

Horizontal

Appearance	State
	0
	1
	n

Setup Procedure

For instructions on how to launch Workshop 4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

Create a New Project

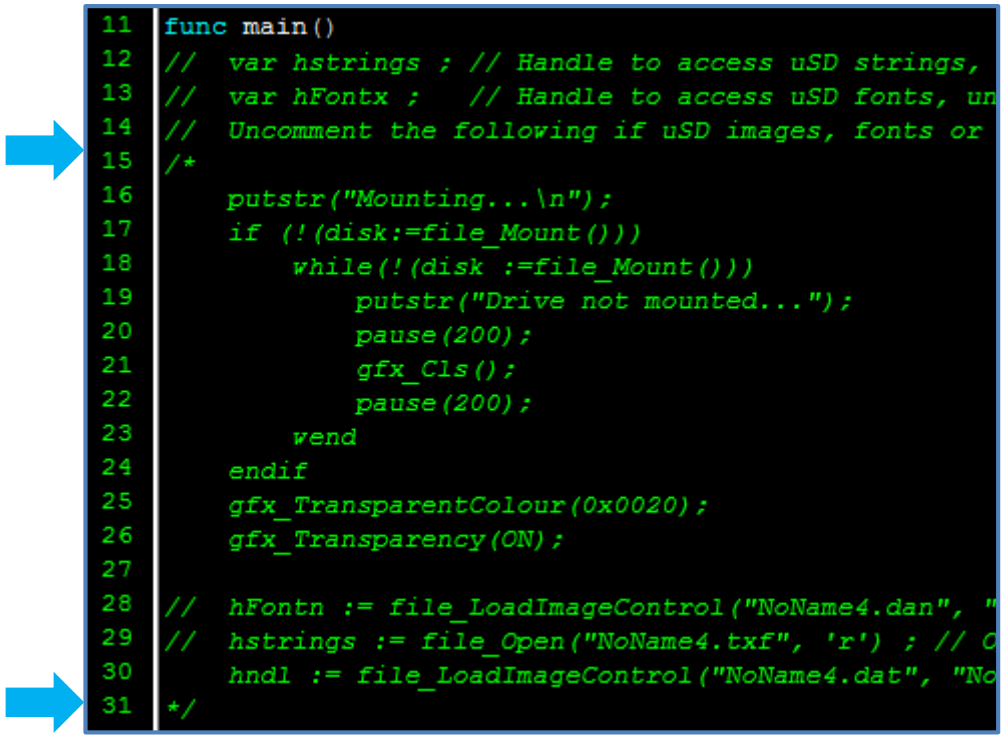
For instructions on how to create a new **ViSi** project, please refer to the section "**Create a New Project**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

Design the Project

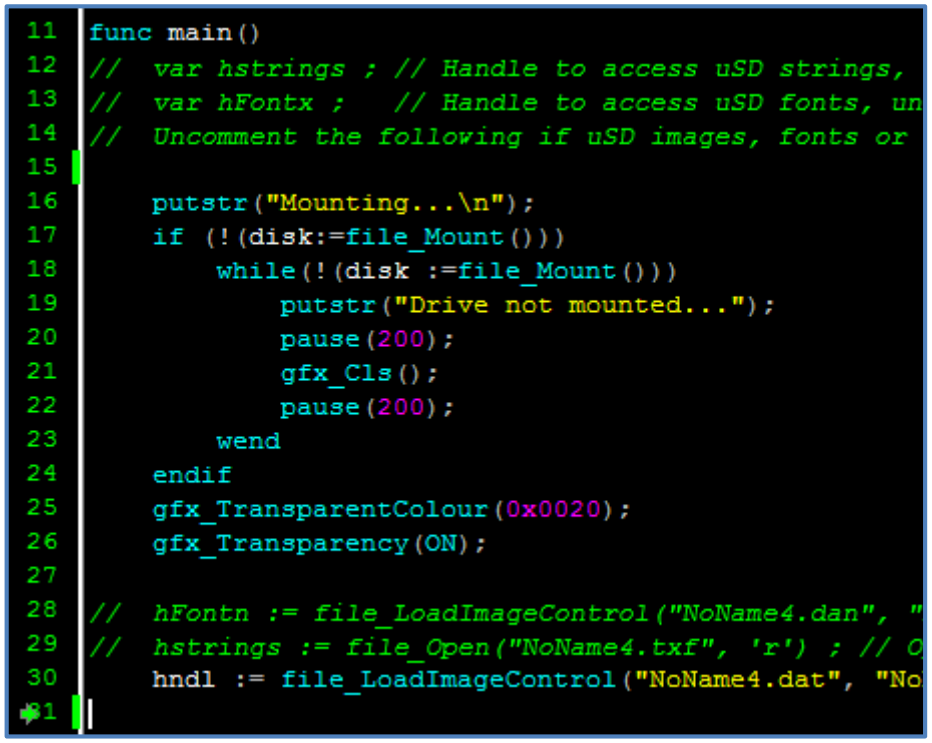
Uncomment the uSD Card Initialization Routine

Remove the block comment symbols as shown below.



```
11  func main()
12  //  var hstrings ; // Handle to access uSD strings,
13  //  var hFontx ;   // Handle to access uSD fonts, un
14  //  Uncomment the following if uSD images, fonts or
15  /*
16      putstr("Mounting...\n");
17      if (!(disk:=file_Mount()))
18          while(!(disk :=file_Mount()))
19              putstr("Drive not mounted...");
20              pause(200);
21              gfx_Cls();
22              pause(200);
23          wend
24      endif
25      gfx_TransparentColour(0x0020);
26      gfx_Transparency(ON);
27
28  //  hFontn := file_LoadImageControl("NoName4.dan", "
29  //  hstrings := file_Open("NoName4.txf", 'r') ; // C
30      hndl := file_LoadImageControl("NoName4.dat", "No
31  */
```

The code screen will be updated accordingly, showing the block as an actual part of the code for compilation.



```
11  func main()
12  //  var hstrings ; // Handle to access uSD strings,
13  //  var hFontx ;   // Handle to access uSD fonts, un
14  //  Uncomment the following if uSD images, fonts or
15
16      putstr("Mounting...\n");
17      if (!(disk:=file_Mount()))
18          while(!(disk :=file_Mount()))
19              putstr("Drive not mounted...");
20              pause(200);
21              gfx_Cls();
22              pause(200);
23          wend
24      endif
25      gfx_TransparentColour(0x0020);
26      gfx_Transparency(ON);
27
28  //  hFontn := file_LoadImageControl("NoName4.dan", "
29  //  hstrings := file_Open("NoName4.txf", 'r') ; // C
30      hndl := file_LoadImageControl("NoName4.dat", "No
31
```

Leave lines 28 and 29 as they are, since they are not needed in this application.

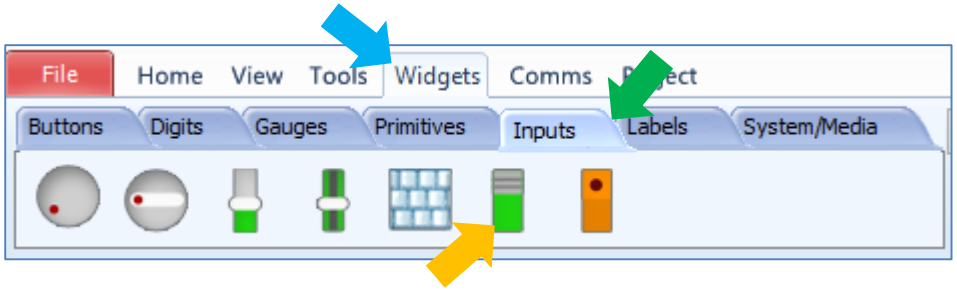
The function **file_LoadImageControl(**fname1,fname2,mode**)** in line 30 creates an image control list. It requires two files – fname1 and fname2, the .dat file and .gci file, respectively. These files are created by Workshop. The

GCI file contains all the graphics for the images and/or videos created by Workshop. The DAT file contains one line for each image or video, that names the object and gives its starting offset within the GCI and its initial X/Y position. The function returns a handle (pointer to the memory allocation) to the image control list that has been created. This handle will be used to access and display objects, as will be shown later.

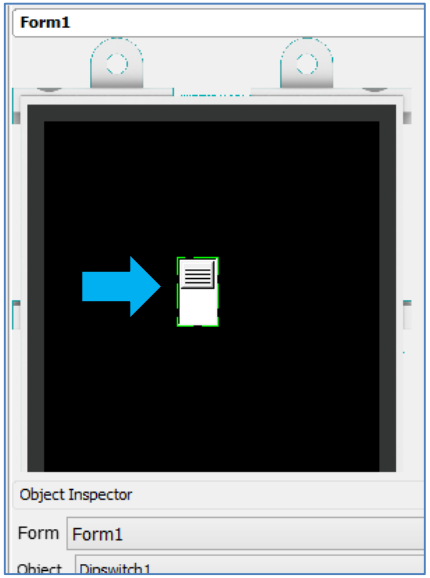
```
29  //  hstrings := file_Open("NoName1.txf", 'r') ; // Open handle to access t
    hndl := file_LoadImageControl("NoName1.dat", "NoName1.gci", 1);
31
```

Add a DIP Switch

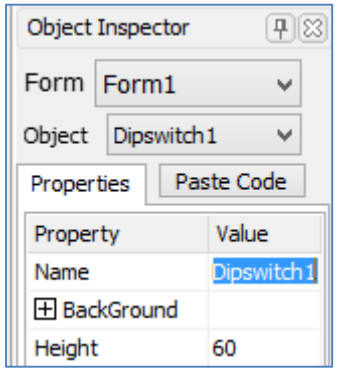
Go to the Widgets menu, select the Inputs pane, and click on the DIP switch icon.

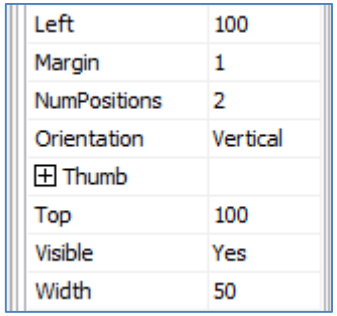


Once the DIP switch is selected, click on the WYSIWYG screen to place it.



The Object Inspector shows the different properties of the DIP switch object. Apply the following property values to the DIP switch.



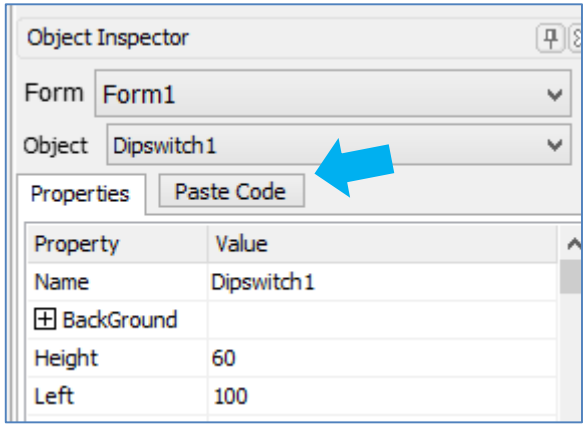


Insert the DIP Switch Code

Go to the code area and place the cursor just after the handle assignment statement (line 32 in this example).

```
28  //  hFontn := file_LoadImageControl("NoName1.dan
29  //  hstrings := file_Open("NoName1.txf", 'r') ;
30      hndl := file_LoadImageControl("NoName1.dat",
31
32
33
34      repeat
35      forever
36  endfunc
```

Having selected the DIP switch object, go to the Object Inspector and click on the Paste Code button.



The code will be updated accordingly.

```
30      hndl := file_LoadImageControl("NoName1.dat", "NoName1.gci", 1);
31
32
33      // Dipswitch1 1.0 generated 8/13/2014 11:14:31 AM
34      img_ClearAttributes(hndl, iDipswitch1, I_TOUCH_DISABLE); // set
35      img_Show(hndl,iDipswitch1) ; // show initial state at 0
36      // Determine new position
37      state := (y - 55) / 25 ;          // (y - top_left) / (height_wid
38      // if (state > 1) state := 1 ;      // if positions > 2 and heigh
39      img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, state) ; // where st
40      img_Show(hndl,iDipswitch1) ;
41
42
43      repeat
44      forever
```

A new block for the DIP switch is generated, along with comments for each line. Additional explanations are given below.

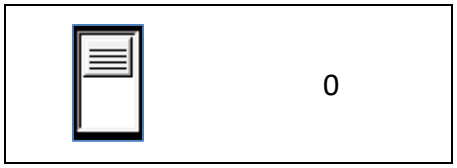
The statement in line 34

```
img_ClearAttributes (hndl, iDipswitch1, I_TOUCH_DISABLE);
```

enables the DIP switch object, Dipswitch1, for touch detection. Not doing this will make Dipswitch1 unresponsive to touch.

The command **img_Show**(hndl,iDipswitch1) displays the DIP switch at the default initial state – state 0.

Appearance	State



Comment out the statements in lines 37, 39, and 40 for now.

```
33      // Dipswitch1 1.0 generated 5/12/2013 12:29:58 PM
34      img_SetWord(hndl, iDipswitch1, IMAGE_FLAGS, (img_Ge
35      img_Show(hndl,iDipswitch1) ; // show initial state
36      // Determine new position
37      //state := (y - 100) / 30 ;          // (y - top_lef
38      // if (state > 1) state := 1 ;      // if positions
39      //img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, state
40      //img_Show(hndl,iDipswitch1) ;
41
```

Change the DIP Switch State From 0 to 1

Appearance	State		Appearance	State
	0			1

To change the state of the DIP switch, we use the command:

```
img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, 1) ;
```

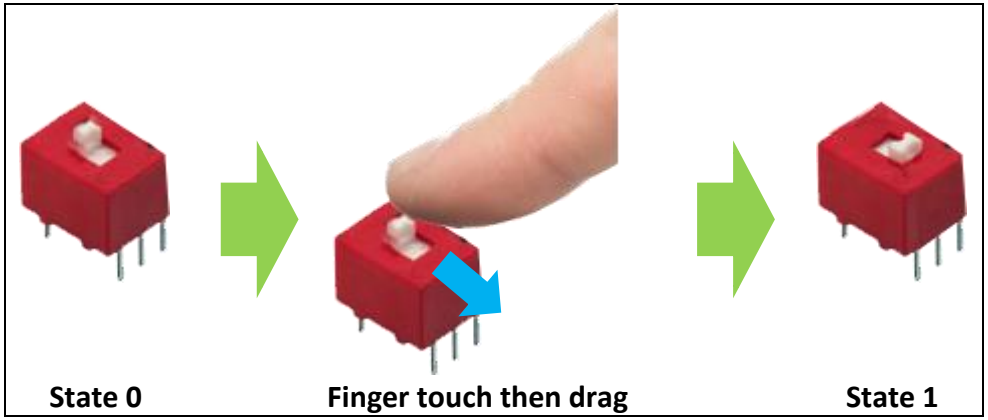
Here the value of IMAGE_INDEX for iDipswitch1 is set to 1. The IMAGE_INDEX is the current frame of the object. There are two frames for iDipswitch1 in this case – frame 0 and frame 1, each representing the object at its different states. The following code will illustrate this.

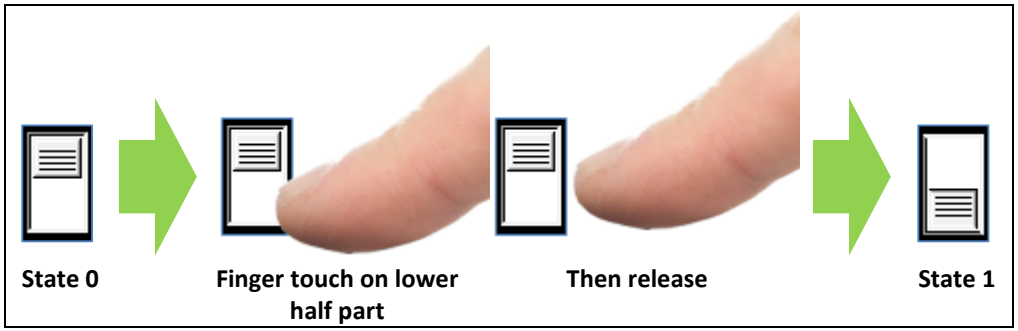
```
//set iDipswitch1 to display frame 1
img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, 1) ;

//display iDipswitch1
img_Show(hndl, iDipswitch1);

//add a delay
pause(2000);

//set iDipswitch to display frame 0
img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, 0) ;

//display iDipswitch1
img_Show(hndl, iDipswitch1);

repeat
forever
```

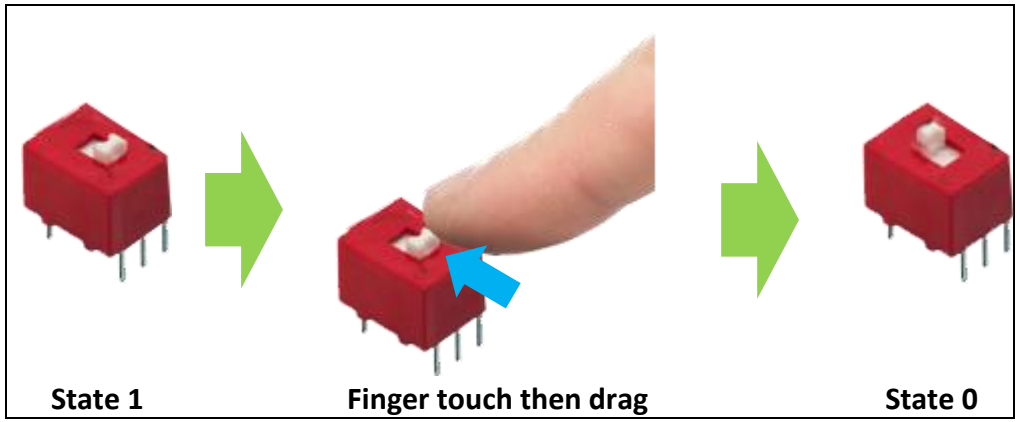
Insert the code above to your main program. When compiled, it should display the DIP switch at state 1 for 2 seconds, then at state 0 forever.

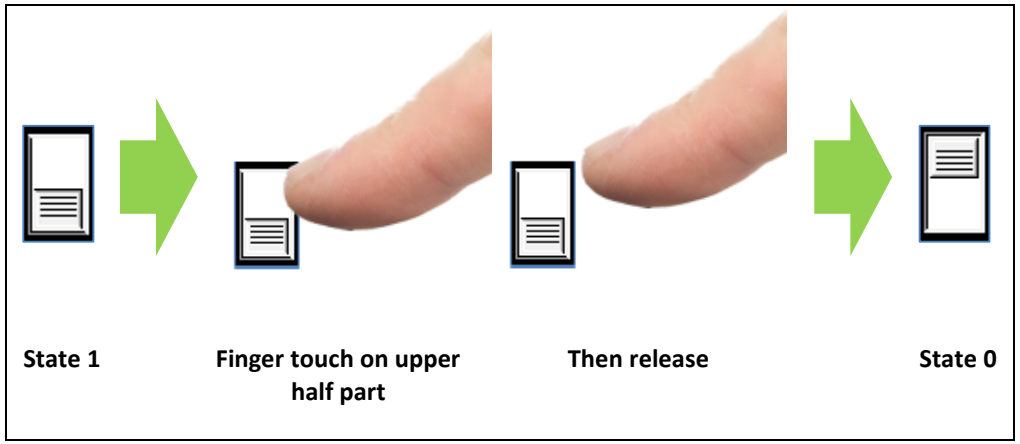
Control the DIP Switch with Touch

In this section we will configure the DIP switch to respond to touch. To do this, we will try to simulate how a real DIP switch works.









Enable Touch Detection

Before using the touch feature, enable it with the function:

```
touch_Set(TOUCH_ENABLE);
```

To disable the feature, use the function:

```
touch_Set(TOUCH_DISABLE);
```

The touch detection feature runs in the background and disabling it when not in use will free up extra resources for the 4DGL CPU cycles.

Check Touch Status

Now that the screen is enabled for touch detection, it needs to be constantly checked for a change in state. 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

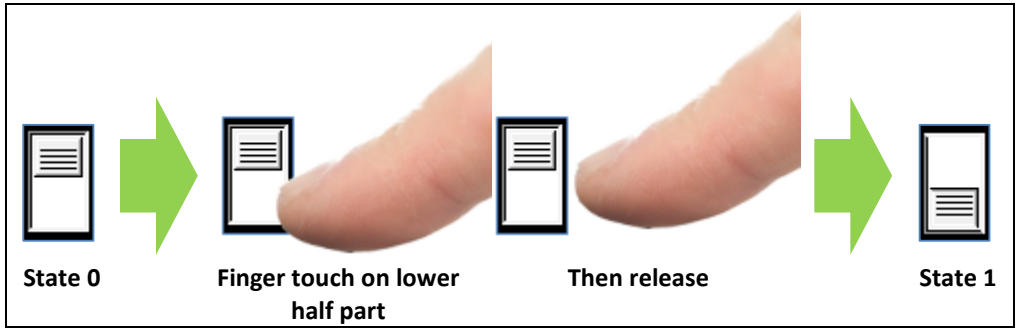
Check if the DIP Switch is Touched

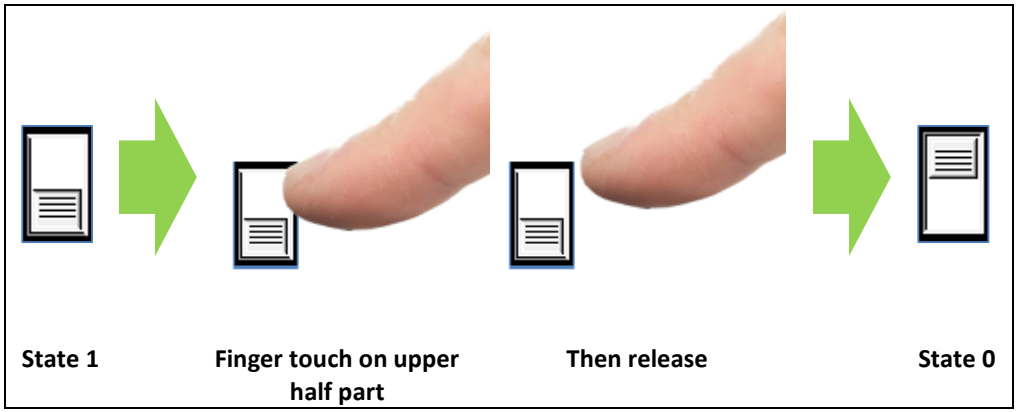
Of course the DIP Switch is only a part of the screen. When the screen is touched, we need to know if the point of touch is within the region of interest, which is the DIP switch. One way to do this is to use the function, **img_Touched**(handle, index)**.** This function returns back the index if the image is touched or returns -1 if not.

```
n := img_Touched(hndl, iDipswitch1)
if(n == iDipswitch1)
      print("iDipswitch1 is touched");
if(n == -1)
      print("Touch is outside iDipswitch1);
endif
```

Check if What Part of the DIP Switch is Touched

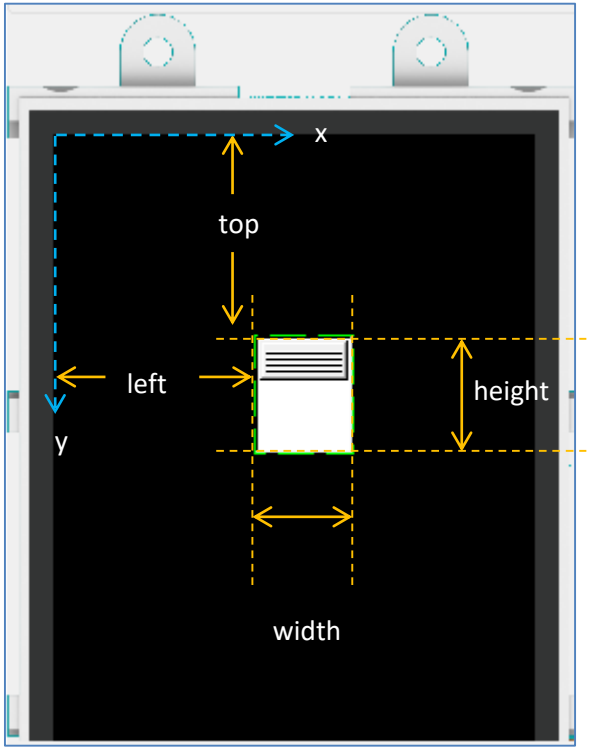
Finally, before deciding which state to display, it necessary that we determine where the last point of touch occurred. To illustrate:

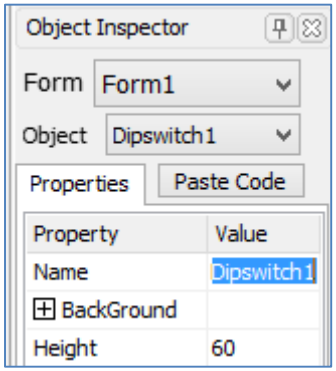


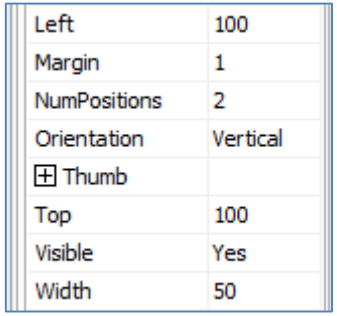


The model above can be coded as follows:

```
// Determine new position
DIPstate :=  (y - 100) / (60/2) ; // (y - top) / (height/ positions)
```







The formula

```
DIPstate := (y - top) / (height/ positions)
```

will assign either 0 or 1 to the variable DIPstate depending on the location of touch and properties of the DIP switch object.

y	y value of touch point
top	location of the DIP switch along the y axis
height	height of the DIP switch
positions	number of positions of the DIP switch

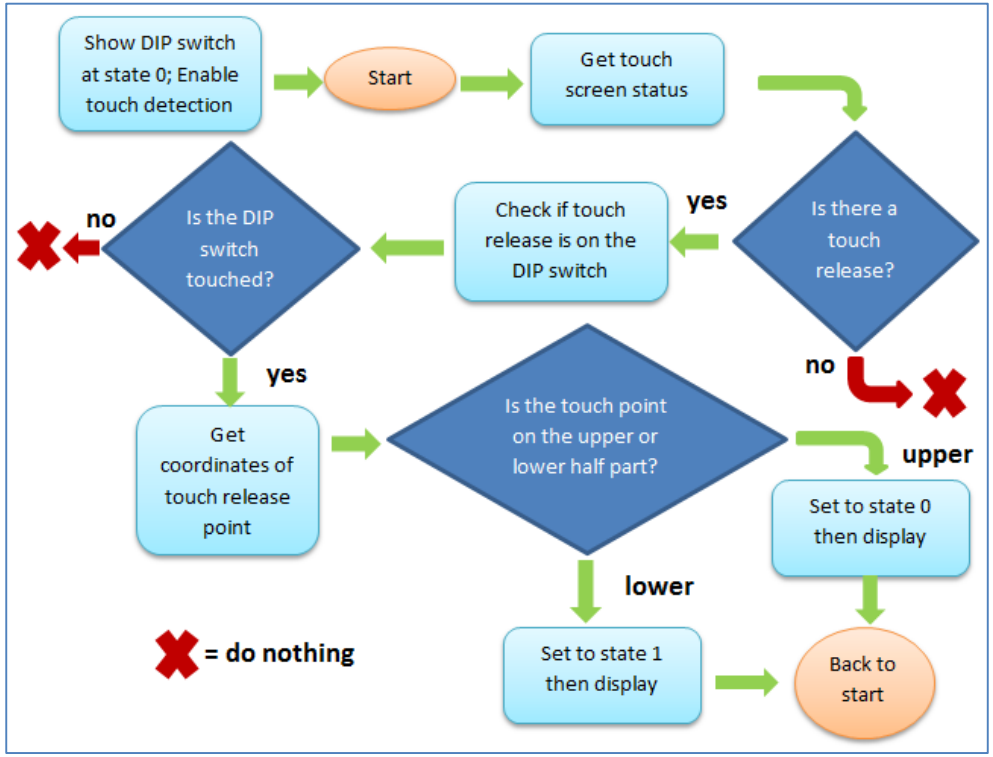
For horizontally oriented DIP switch objects, the formula is

```
DIPstate := (x - left) / (width/ positions)
```

The formulas above are quite handy when dealing with objects with more than two positions or states. Of course you can have your own way of determining where the point of touch occurred within the object.

An Example

Below is a code for a program that initially displays a DIP switch at state 0. The state will then change depending on what part of the switch is touched. The code comes with a process flow chart to help explain the touch detection part.



A Workshop file is attached containing the same code as shown below. For the touch detection part, note that only the TOUCH_RELEASED state was used in this program. There are two other states, namely TOUCH_PRESSED and TOUCH_MOVING. You can easily modify the program and experiment with these states.

```
#platform "uLCD-32WPTU"

// Program Skeleton 1.0 generated 5/12/2013 12:28:03 PM

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "DIPSwitchTutorialConst.inc"

func main()

var state, x, y, n, DIPstate;

    putstr("Mounting...\n");
    if (!(disk:=file_Mount()))
        while(!(disk :=file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif

    gfx_TransparentColour(0x0020);
    gfx_Transparency(ON);
    gfx_Cls();

    hndl := file_LoadImageControl("DIPSWI~1.dat", "DIPSWI~1.gci", 1);

    // Dipswitch1 1.0 generated 5/12/2013 12:29:58 PM
    img_ClearAttributes(hndl, iDipswitch1, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    img_Show(hndl,iDipswitch1) ; // show initial state at 0
    touch_Set(TOUCH_ENABLE);

    repeat          //Start (process flow chart)

    state := touch_Get(TOUCH_STATUS);   //get touch screen status
```

```
//-------------------------------------------------------------
    if(state == TOUCH_RELEASED)         //if there's a release

        n := img_Touched(hndl,iDipswitch1);
        if(n == iDipswitch1)            //check if object is touched
            x := touch_Get(TOUCH_GETX);
            y := touch_Get(TOUCH_GETY);
            DIPstate :=  (y - 100) / (60/2);
            img_SetWord(hndl, iDipswitch1, IMAGE_INDEX, DIPstate); //change state
            img_Show(hndl, iDipswitch1);                           //show state

            gfx_MoveTo(160,125);        //move origin to point (160,125)
            print([BIN2Z]DIPstate);     //print the value of DIPstate using a two-bit binary format
        else
            gfx_MoveTo(160,125);        //move origin to point (160,125)
            print(n);                   //print -1, return value of img_Touched()
        endif                           //if object is not touched

    endif
    //-------------------------------------------------------------

    //-------------------------------------------------------------
    if(state == TOUCH_PRESSED)         //if there's a press
        x := touch_Get(TOUCH_GETX);    //do nothing
        y := touch_Get(TOUCH_GETY);    //just get touch coordinates
    endif
    //-------------------------------------------------------------

    //-------------------------------------------------------------
    if(state == TOUCH_MOVING)         //if there's movement
        x := touch_Get(TOUCH_GETX);   //do nothing
        y := touch_Get(TOUCH_GETY);   //just get touch coordinates
    endif
    //-------------------------------------------------------------

    forever     //back to start (process flow chart)

endfunc
```

Run the Program

For instructions on how to save a **ViSi** project, how to connect the target display to the PC, how to select the program destination (this option is not available for Goldelox displays), and how to compile and download a program, please refer to the section "**Run the Program**" of the application note

ViSi Getting Started - First Project for Picaso and Diablo16

The uLCD-32PTU and uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability