APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi: Diablo16 and Pixxi Internal (PmmC) Widgets

DOCUMENT DATE: **24th JUNE 2020**
DOCUMENT REVISION: **1.0**

Description

This application note provides instructions for designing and using the Pixxi or Diablo16 Internal (PmmC) widgets for a ViSi application.

Before getting started, the following are required:

Hardware

- Any 4D Systems display module powered by any of the following processors:
  - Diablo16 (with PmmC version 2.2 or higher)
  - Pixxi28/44
- Programming Adaptor for target display module

Software

- Workshop4 IDE

This application note comes with one (1) ViSi project:

- PmmC_Widgets.4DViSi

Note: Using a non-4D programming interface could damage the processor and void the warranty.

Content

Application Overview

This document is mainly focused on showing the simple use of the PmmC widgets on the ViSi environment of the Workshop4 IDE. This type of widget is very useful in generating Graphical User Interfaces without the use of any external memory storage device. The parameters of each widget of this type resides in the user code as data blocks. Therefore, each widget adds a small amount to the program/code size.

These widgets can be designed and manipulated using the properties tab in the Object Inspector of the ViSi environment. For more details, please refer to the following manual:

- **Workshop4 Widgets Reference Manual**

The simple project developed in this application note demonstrates a Internal Button Widget acting as a momentary switch for controlling a Internal LED widget.

Setup Procedure

For instructions on how to launch Workshop4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of any of the following application notes:

- **ViSi Getting Started - First Project for Picaso and Diablo16**
- **ViSi Getting Started - First Project for Pixxi Display Modules**

Create a New Project

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

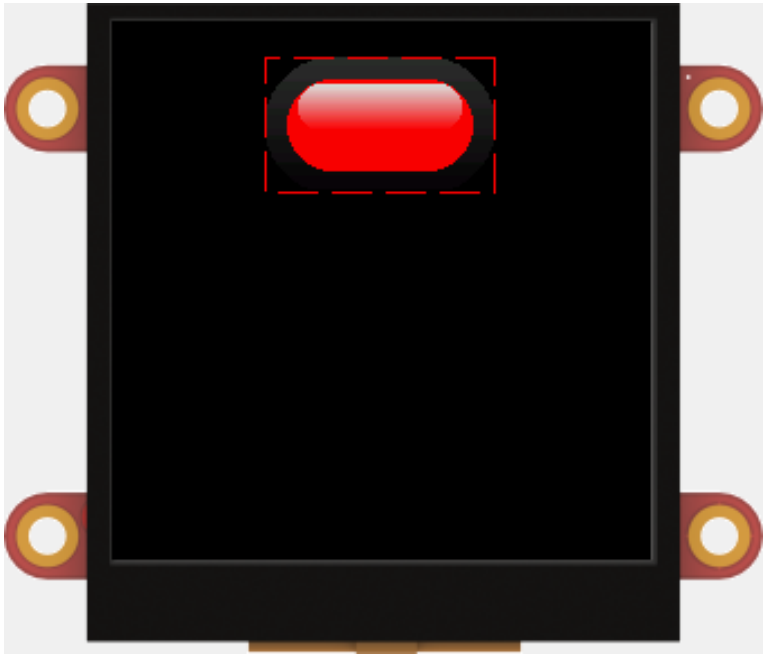
- **ViSi Getting Started - First Project for Picaso and Diablo16**
- **ViSi Getting Started - First Project for Pixxi Display Modules**

Design the Project

Adding an Internal Led

Add a Internal Led to the Form by clicking on the Digits tab, then selecting the Internal Led as shown below.

Click the WYSIWYG screen to place it, then drag it to the desired position.

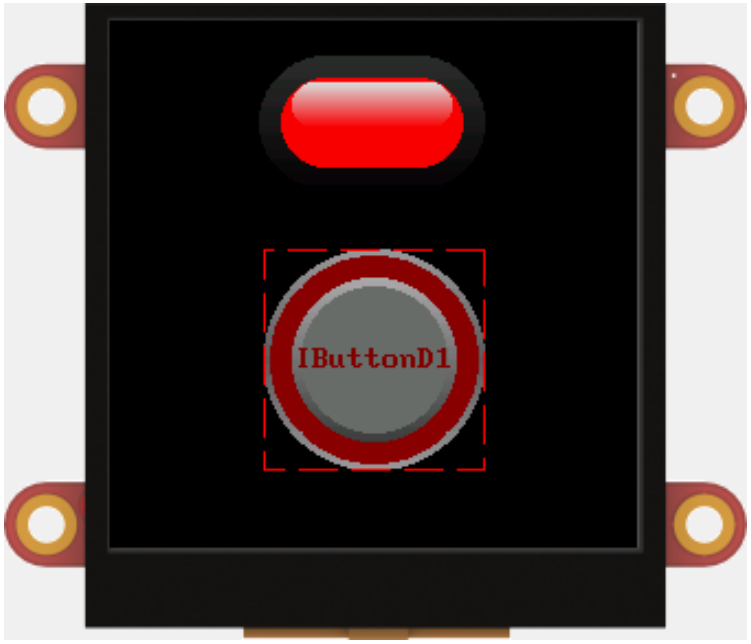


The Led widget can be modified as needed through the **Object Inspector**.

Adding an Internal Button

Add a Internal Button to the Form by clicking on the Buttons tab, then selecting the Internal Button as shown below.

Click the WYSIWYG screen to place it, then drag it to the desired position.



The Button widget can be modified as needed through the **Object Inspector**.

Programming the Display

It is always ideal to prepare the graphics interface before moving to programming the display. This lessens the number of times the users will have to proceed on doing a **Paste Code** action in order to regenerate the widget parameters in the code editor.

When the user interface is near its final stages, it is the best time to proceed with programming.

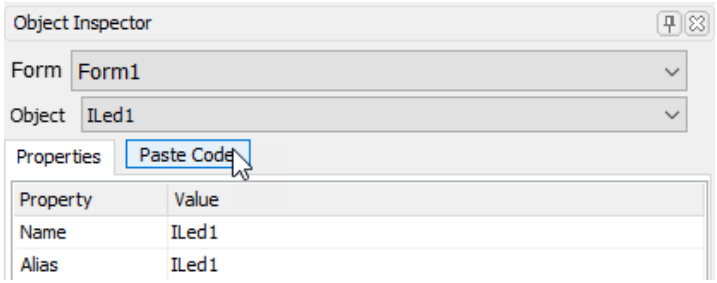
Paste Internal Led Code

Similar to the GCI widgets, ViSi provides Paste Code feature for Internal widgets.

Place the blinking cursor inside the repeat-forever loop.

```
    repeat
    |
    forever
endfunc
```

Go to the Object Inspector, choose ILed1 and click **Paste Code**



This will paste the function for rendering the Led widget.

```
    // ILed1 1.0 generated 05/02/2020 11:41:48 AM
    gfx_Led(value, ILed1RAM, IILed1) ;
```

The parameters and variables required by the function are also automatically inserted near the start of the code.

```
#platform "pixxiLCD-35P4CT"


// Program Skeleton 1.5 generated 05/02/2020 9:26:25 AM

// #MODE RUNFLASH uncomment and set Destination to Flash to run from Flash, refer

#inherit "4DGL_16bitColours.fnc"

#DATA
    // ILed1 Data Start
    word IILed1 108, 80, 103, 61, 0x2965, BLACK, 0xDEFB, 0xF800, 0x5800, 20,
        30, 10, 20, 0x1
    // ILed1 Data End
#END
var ILed1RAM[WIDGET_RAM_SPACE] ;
```

Notice the fixed integers and colour included in the data block for the ILed1 function gfx_Led. These parameters are based on the values in the Object Inspector and may have to be regenerated by doing a new paste code action if the widget parameters are changed.

Paste the Internal Button Code

Place the blinking cursor in the place where you want the code is pasted.

```
repeat

// ILed1 1.0 generated 05/02/2020 11:41:48 AM
gfx_Led(value, ILed1RAM, IILed1) ;

|

forever
```

Go to the Object Inspector, choose IButtonD1 and click **Paste Code**

This will paste the function for rendering the Button widget.

```
// IButtonD1 1.0 generated 05/02/2020 11:42:00 PM
gfx_Button4(state, IButtonD1RAM, IIButtonD1) ;
```

The parameters and variables required by the function are also automatically added to the data block near the start of the code.

```
#DATA
    // IButtonD1 Data Start
    word IIButtonD1 110, 228, 50, 0x9CD3, 0x5ACB, 0x70, DARKRED, RED, 0xAD55,
        0x39E7, 0x20, 0x40, 0x6B6D, CIButtonD1, MAROON, RED, FONT3, 1
    byte CIButtonD1 "IButtonD1",0
    // IButtonD1 Data End
    // ILed1 Data Start
    word IILed1 108, 80, 103, 61, 0x2965, BLACK, 0xDEFB, 0xF800, 0x5800, 20,
        30, 10, 20, 0x1
    // ILed1 Data End
#END
var IButtonD1RAM[WIDGET_RAM_SPACE] ;
var ILed1RAM[WIDGET_RAM_SPACE] ;
```

Notice the fixed integers and colour included in the data block for the IButtonD1 function gfx_Button4. These parameters are based on the values in the Object Inspector and may have to be regenerated by doing a new paste code action if the widget parameters are changed.

Widget Handling

The best way to manage multiple widgets is to setup a widget handler. The function **widget_Create** generates a widget control capable of holding count elements and returns a handle (pointer to the memory allocation) to the image control list. This handle will be used to access and display objects, as will be shown later.

```
func main()
    var touchState;
    var state, value, n;

    hndl :=  widget_Create(1);  // Generate widget control handle
```

With the handle established, the widgets can then be added into the handle through the **widget_Add** function. This will add the Button widget entry as index 0.

```
    widget_Add(hndl, 0, IButtonD1RAM);  // Add Widget to handle
```

The function **widget_SetAttributes** is then used to enable the touch detection attribute on the button widget.

```
    widget_SetAttributes(hndl, 0, WIDGET_F_TOUCH_ENABLE);   // Enable touch for Button
```

Touch Detection

The touchscreen for the pixxiLCD display is enabled through the **touch_Set** function.

```
touch_Set(TOUCH_ENABLE) ;   // Enable touch
```

In the repeat loop, the touchscreen status is constantly checked for changes.

```
repeat
    touchState := touch_Get(TOUCH_STATUS);   // Get Touch Status
```

If a touch press is detected, the conditional proceeds to check which widget is touched through the function **widget_Touched**. This function will scan all of the widget in the handle with enabled touch detection attribute.

```
    if(touchState == TOUCH_PRESSED)
        n := widget_Touched(hndl, ALL); //scan widget list, looking for a touch
```

If the touch function returns index 0, this means that the button is pressed. The button is updated to show its pressed state and the LED is turned on.

```
if(touchState == TOUCH_PRESSED)
    n := widget_Touched(hndl, ALL); //scan widget list, looking for a touch
    if(n == 0)  // Button is touched
        state := 1;
        gfx_Button4(state, IButtonD1RAM, IIButtonD1) ;  // Draw button released
        gfx_Led(state, ILed1RAM, IILed1) ;      // Update LED widget
    endif
```

Similarly, when the touch is released over the button, the button is updated back to its normal state and LED is turned off.

```
else if(touchState == TOUCH_RELEASED)
    n := widget_Touched(hndl, ALL); //scan widget list, looking for a touch
    if(n == 0)  // Button is touched
        state := 0;
        gfx_Button4(state, IButtonD1RAM, IIButtonD1) ;  // Draw button released
        gfx_Led(state, ILed1RAM, IILed1) ;      // Update LED widget
    endif
endif
```

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, and how to compile and download a program, please refer to the section "**Run the Program**" of any of the following application notes:

- **ViSi Getting Started - First Project for Picaso and Diablo16**
- **ViSi Getting Started - First Project for Pixxi Display Modules**

Proprietary Information

Disclaimer of Warranties & Limitation of Liability