



ViSi: PIXXI PmmC Input Widgets

DOCUMENT DATE: **6th FEBRUARY 2020**
DOCUMENT REVISION: **1.0**



Description

This application note provides instructions for designing and using the PmmC knob and slider 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:
 - Pixxi28/44
- [Programming Adaptor for target display module](#)

Software

- [Workshop4](#)

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

- Knob_Slider_Digits.4DViSi

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

Content

Description 2

Content..... 2

Application Overview 3

Setup Procedure 3

Create a New Project 3

Design the Project..... 4

Adding a PmmC Knob 4

Adding a PmmC Slider 4

Adding a PmmC LedDigits..... 5

Programming the Display 5

 Paste the PmmC Knob Code 5

 Paste the PmmC Slider Code 6

 Paste the PmmC LedDigits Code 7

 Widget Handling 7

 Touch Detection 8

Run the Program..... 8

Proprietary Information..... 9

Disclaimer of Warranties & Limitation of Liability 9

Application Overview

This document is mainly focused on showing the simple use of the PmmC knob and slider widgets on the ViSi environment of the Workshop4 IDE. These type of widgets are very useful in generating Graphical User Interfaces without the use of any external memory storage device. The parameters used to compose this widgets during runtime resides in the user flash as data blocks.

Depending on the user's preferences, 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:

Pixxi Widgets Reference Manual

The simple project developed in this application note demonstrates a PmmC Knob and PmmC Slider used to update a PmmC LedDigits.

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 the following application note:

- [ViSi Getting Started - First Project for Pixxi Display Modules](#)

Create a New Project

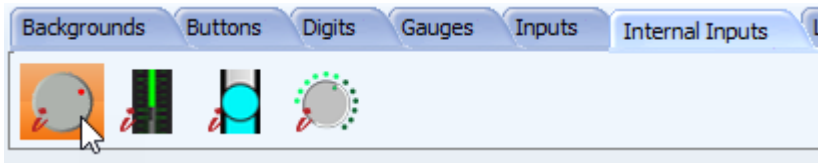
For instructions on how to create a new **ViSi** project, please refer to the section “**Create a New Project**” of the following application note:

- [ViSi Getting Started - First Project for Pixxi Display Modules](#)

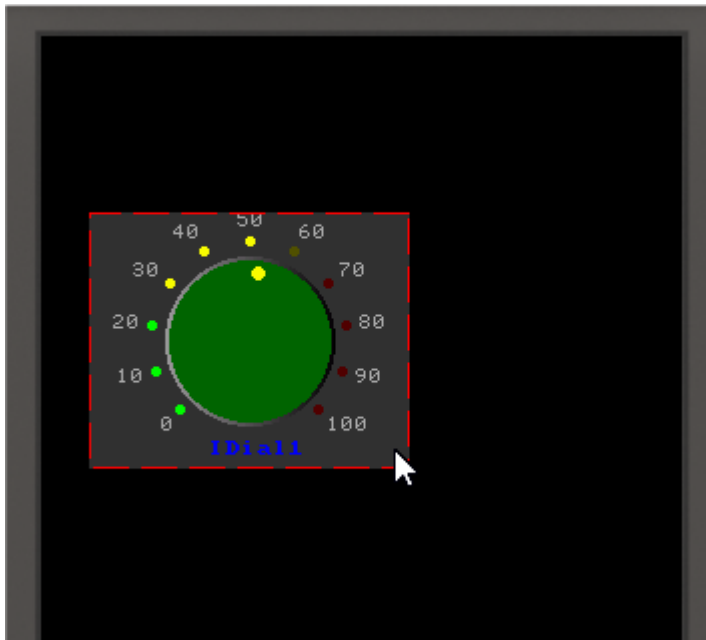
Design the Project

Adding a PmmC Knob

Add a PmmC Knob to the Form by clicking on the Internal Inputs tab, then selecting the PmmC Knob as shown below.



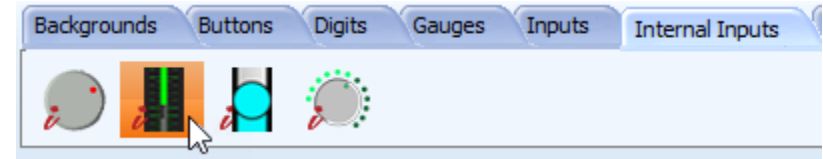
Click the WYSIWYG screen to place it, then drag it to the desired position.



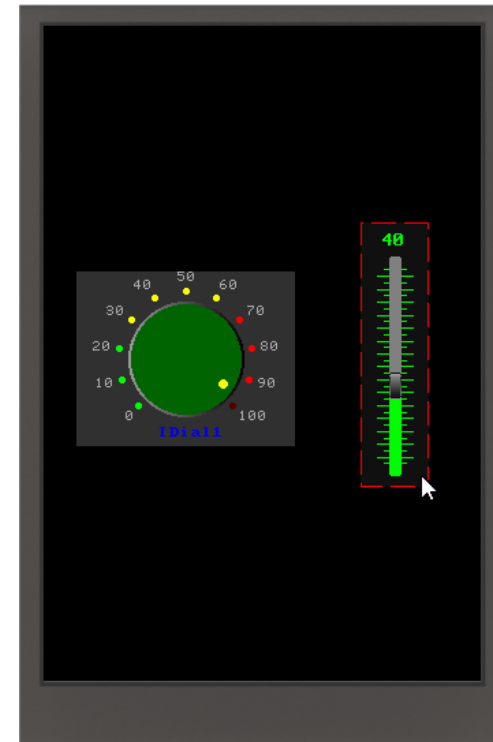
The Knob widget can be modified as needed through the **Object Inspector**.

Adding a PmmC Slider

Add an PmmC Slider to the Form by clicking on the Internal Inputs tab, then selecting the PmmC Slider as shown below.



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



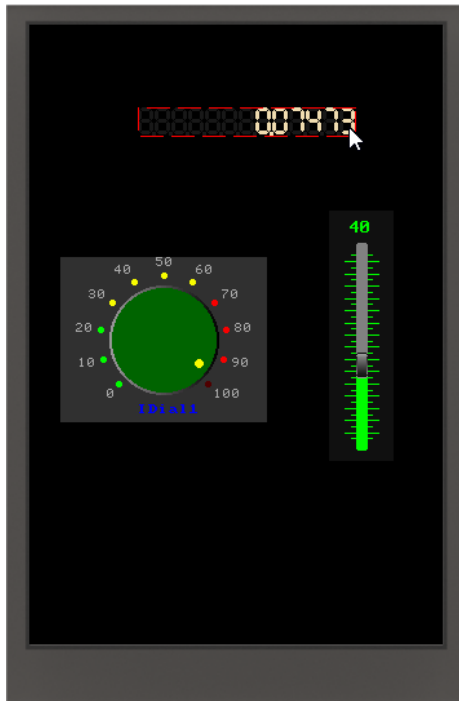
The Slider widget can be modified as needed through the **Object Inspector**.

Adding a PmmC LedDigits

Add an PmmC LedDigits to the Form by clicking on the Digits tab, then selecting the PmmC LedDigits as shown below.



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



The LedDigits 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.

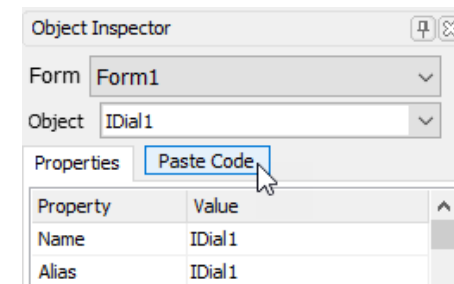
Paste the PmmC Knob Code

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

Place the blinking cursor inside the repeat-forever loop.

```
repeat
|
forever
endfunc
```

Go to the Object Inspector, choose IDial1 and click **Paste Code**



This will paste the function for rendering the Knob widget.

```
repeat
// IDial1 1.0 generated 06/02/2020 2:09:21 PM
gfx_Dial(posn, IDial1RAM, IIDial1) ; // where posn is 0 to 100
```

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

```
// IDial1 Data Start
word IIDial1 24, 180, 160, 128, 104, 244, 40, 135, 405, 0, 100, 0x3186,
DARKGREEN, 2, BLACK, 0x9CD3, 200, 300, 0x0280, 0x5280, 0x5000, LIME,
YELLOW, RED, 12, 2, 8, YELLOW, 3, 30, DARKGRAY, FONT1, 16, 10, 0,
FONT2, BLUE, -20, 50, CIDial1, 0x0
byte CIDial1 "IDial1", 0
// IDial1 Data End
```

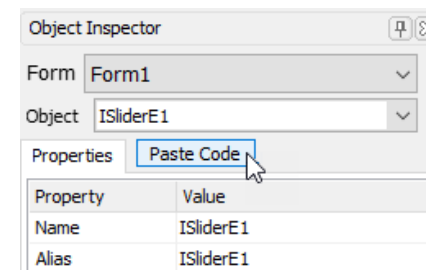
Notice the fixed integers and colour included in the data block for the knob function `gfx_Dial`. These parameters are based on the values in the Object Inspector and may have to be regenerated by doing a new paste code action if the widget parameters are changed.

Paste the PmmC Slider Code

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

```
repeat
// IDial1 1.0 generated 06/02/2020 2:09:21 PM
gfx_Dial(posn, IDial1RAM, IIDial1) ; // where posn is 0 to 100
|
```

Go to the Object Inspector, choose `ISliderE1` and click **Paste Code**



This will paste the function for rendering the Slider widget.

```
// ISliderE1 1.0 generated 06/02/2020 2:09:41 PM
gfx_Slider5(posn, ISliderE1RAM, IISliderE1) ; // where posn is 0 to 100
```

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

```
// ISliderE1 Data Start
word IISliderE1 232, 144, 194, 50, 0x7, 0, 100, 0x1082, GRAY, LIME, 30, 30,
2, 2, 10, LIME, 5, LIME, FONT3, LIME, 0x1082, 0x9CD3, 0x20, 0x1082,
0x9CD3, 0x40
// ISliderE1 Data End
```

Notice the fixed integers and colour included in the data block for the slider function `gfx_Slider5`. 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 PmmC LedDigits Code

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

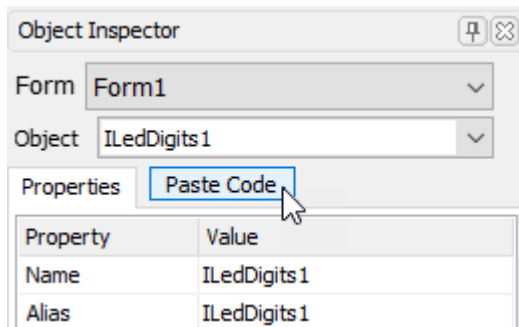
```
repeat
  // IDial1 1.0 generated 06/02/2020 2:09:21 PM
  gfx_Dial(posn, IDial1RAM, IIDial1) ;    // where posn is 0 to 100

  // ISliderE1 1.0 generated 06/02/2020 2:09:41 PM
  gfx_Slider5(posn, ISliderE1RAM, IISliderE1) ;    // where posn is 0 to 100

  |

  forever
endfunc
```

Go to the Object Inspector, choose ILedDigits1 and click **Paste Code**



This will paste the function for rendering the LedDigits widget.

```
// ILedDigits1 1.0 generated 06/02/2020 2:09:51 PM
gfx_LedDigits(posn, ILedDigits1RAM, IILedDigits1) ;
```

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

```
// ILedDigits1 Data Start
word IILedDigits1 116, 56, 117, 69, 3, 0, 0, 3, WHEAT, 0x18C3, 0x0
// ILedDigits1 Data End
```

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.

```
hndl := widget_Create(2);
```

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, IDial1RAM);
widget_Add(hndl, 1, ISliderE1RAM);
```

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);
widget_SetAttributes(hndl, 1, WIDGET_F_TOUCH_ENABLE);
```

In order to show the widgets on the screen, their functions must be called at least once before the repeat-forever loop.

```
gfx_Dial(posit, IDial1RAM, IIDial1) ;
gfx_LedDigits(posit, ILedDigits1RAM, IILedDigits1) ;
gfx_Slider5(posn, ISliderE1RAM, IISliderE1) ;
```

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. The X and Y coordinates are also constantly stored for later calculations.

```
repeat
  touchStatus := touch_Get(TOUCH_STATUS);
  x := touch_Get(TOUCH_GETX);
  y := touch_Get(TOUCH_GETY);
```

If a touch press or touch moving event 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 (touchStatus == TOUCH_PRESSED || touchStatus == TOUCH_MOVING)
  n := widget_Touched(hndl, ALL); // scan widget list, looking for a touch
```

If the touch function returns index 0, this means that the knob is touched. The angular position is calculated using the **gfx_XYrotToVal** function. The position is then used to update the knob widget and the LedDigits widget.

```
if (n == 0)
  // Calculate Angular Position
  posit := gfx_XYrotToVal(x-106, y-246, XYROT_SOUTH, 45, 315, 0, 100);

  // Update Knob
  gfx_Dial(posit, IDial1RAM, IIDial1); // where posit is 0 to 100
  // Update LedDigits
  gfx_LedDigits(posit, ILedDigits1RAM, IILedDigits1);
```

Otherwise if the touch function returns index 1, this means that the slider is touched. The linear position is calculated using the **gfx_XYlinToVal** function.

The position is then used to update the slider widget and the LedDigits widget.

```
else if (n == 1)
  // Calculate Linear Position
  posn := gfx_XYlinToVal(x, y, 0, 176, 327, 0, 100);

  // Update Slider
  gfx_Slider5(posn, ISliderE1RAM, IISliderE1); // where posn is 0 to 100
  // Update LedDigits
  gfx_LedDigits(posn, ILedDigits1RAM, IILedDigits1);
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 the following application notes:

- [ViSi Getting Started - First Project for Pixxi Display Modules](#)

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.