APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi-Genie: Raspberry Pi Switching Banks

DOCUMENT DATE:	9th MAY 2020
DOCUMENT REVISION:	1.0

W W W . 4 D S Y S T E M S . C O M . A U

Description

This application note shows how to switch between banks of the Diablo16 processor using a Raspberry Pi Host.

Before getting started, the following are required:

Hardware

- Any 4D Systems display module powered by the Diablo16 processor
- Programming Adaptor for target display module
- uSD Card
- USB Card Reader
- Raspberry Pi

Software

- Workshop4
- This requires the **PRO** version of Workshop4

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

Content

Application Overview

The Diablo16 processor has six flash banks (Bank 0 to Bank 5), each of which has a capacity of 32 kB. As of WS4 version 4.5.0.8, it is now possible for the user to specify the destination flash bank of a ViSi-Genie program. This was not possible in previous versions of Worskhop4. Prior to version 4.5.0.8, bank 0 was the only possible flash memory destination of a ViSi-Genie program.

The purpose of this application note is to show how to switch between banks using a Raspberry Pi as a host controller. This application note uses the ViSi-Genie environment, together with Genie-Magic. Thus, the PRO version of WS4 is required. This application note is applicable to Diablo16 display modules only.

For more information on the basics of the multiple flash bank feature in ViSi-Genie, refer to the application note ViSi-Genie Flash Banks.

Setup Procedure

For instructions on how to launch Workshop4, how to open a **ViSi-Genie** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note.

- First ViSi-Genie Project for Diablo16

Create a New Project

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

- First ViSi-Genie Project for Diablo16

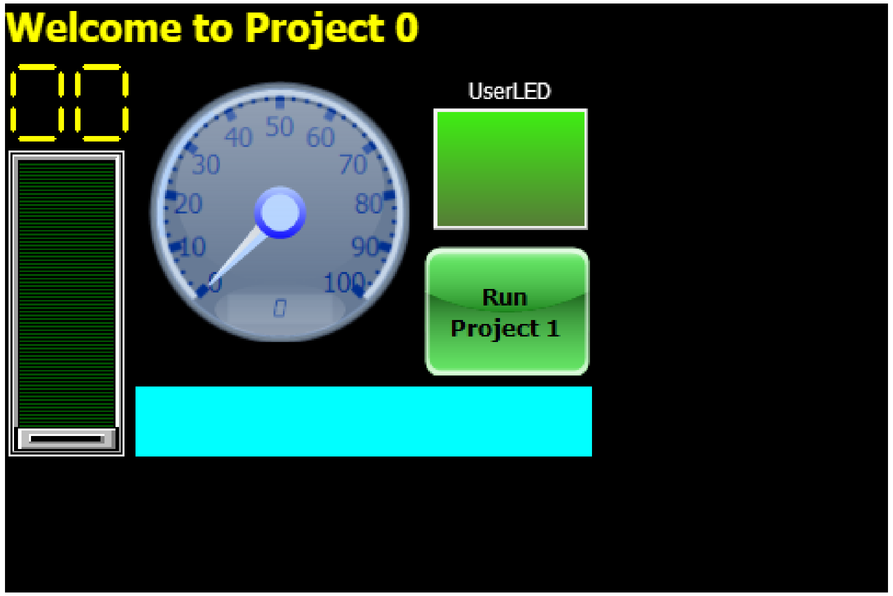
Design the Project

For this application note, a gen4-uLCD-35DCT-CLB will be used for the project. The same procedure is applicable for any Diablo16 displays. Also, this application note comes with zip files which contain demo projects needed for the discussions.

Opening Project Files

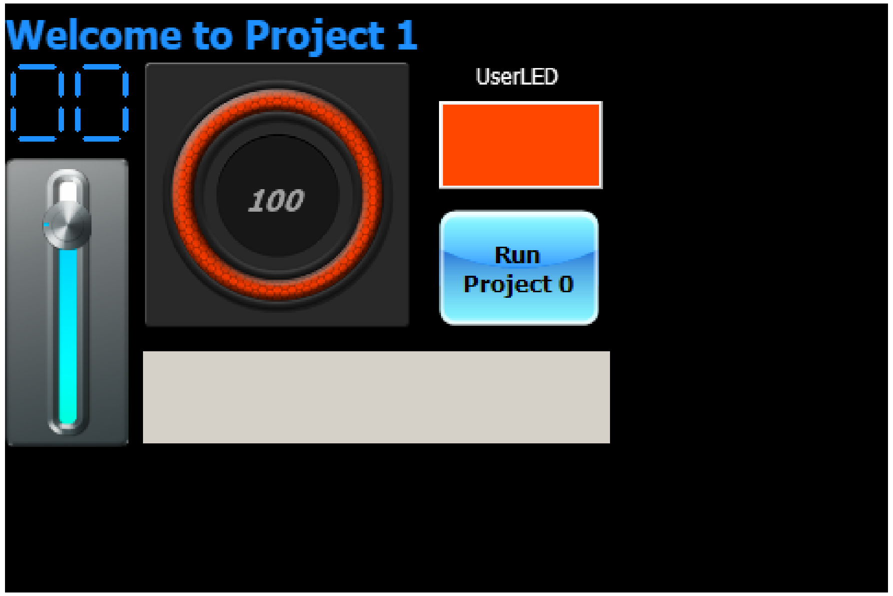
Open the project files inside the zip files “BnkPi0.zip” and “BnkPi1.zip”.

Note that the projects contain magic objects, so WS4 PRO is needed to open them. The project “BnkPi0” should contain the objects shown below. The target bank for project “BnkPi0” is bank 0.



Note that these objects do not occupy the whole screen area. This is in consideration of easily testing the project with smaller displays. You may resize the project if you desire.

The project “BnkPi1”, on the other hand, is shown below. The target bank for this project is bank 1.

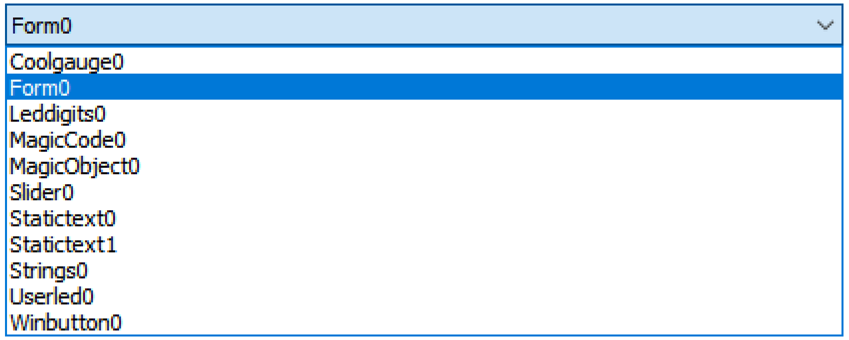


Again, please note that these objects do not occupy the whole screen area in consideration of easily testing the project with smaller displays. You may resize the project if you desire.

Both projects will communicate with the Raspberry Pi at 115200 baud.

Project 0 Objects

The first project contains the following objects:


Form0

Coolgauge0
Form0
Leddigits0
MagicCode0
MagicObject0
Slider0
Statictext0
Statictext1
Strings0
Userled0
Winbutton0

The input objects present in the project are a Winbutton and a Slider.

The Winbutton sends an event to the host controller that should give the Raspberry Pi a signal that the user wants to navigate to the other project.

Note: It is important for applications which utilize multiple flash banks with different ViSi Genie projects that the host controller knows what flash bank the display is currently at.

If the Winbutton's event is successfully received, the host will reply by issuing a WRITE_OBJ command to a MagicObject. The MagicObject will then run the appropriate flash bank stated in the host's command if the flash bank is valid. Otherwise, the display will reset.

```
func rMagicObject0(var action, var object, var newVal, var *ptr)

    var returnFB := 0;
    if(action == WRITE_OBJ)
        seroutX(ACK);
        pause(5);
        returnFB := flash_Run(newVal);
```

```
        if(returnFB)
            gfx_MoveTo(0,0);
            if(returnFB == -1)
                print("invalid bank number");
            else if(returnFB == -2)
                print("no valid program in the selected bank");
            else
                print("run bank not successful, unknown error");
            endif

            print("\nrestarting...");
            pause(2000);
            SystemReset();
        endif
    else if(action == READ_OBJ)
         SendReport(REPORT_OBJ, tMagicObject, 0,flash_Bank()) ;
         // let the host know the current bank
    endif
endfunc
```

As seen in the code above, the MagicObject can also be read and it will send a REPORT_OBJ stating the current flash bank that is running.

```
SendReport(REPORT_EVENT, tMagicObject, 0, flash_Bank()) ;
```

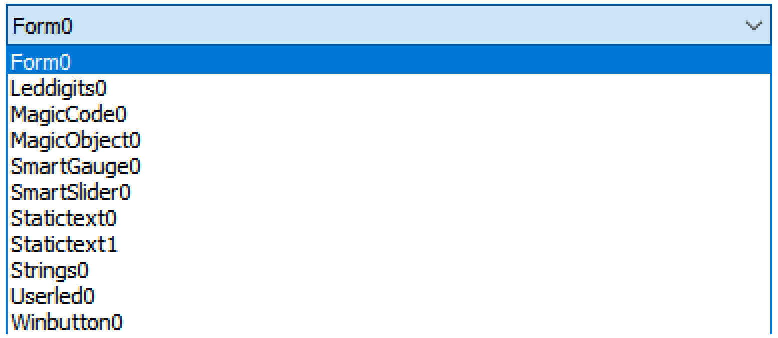
The code above can also be found in MagicCode0 which is set to Post Genie Initialize. This sends a signal to the host that the display is ready and what bank it is currently running.

The Slider will simply report an OnChanged Event. Once received, the host will update the LedDigits object accordingly.

The other remaining objects act as outputs that will be controlled by the host.

Project 1 Objects

The second project contains the following objects:



```
Form0
Form0
Leddigits0
MagicCode0
MagicObject0
SmartGauge0
SmartSlider0
Statictext0
Statictext1
Strings0
Userled0
Winbutton0
```

The input objects present in the project are a Winbutton and a SmartSlider.

It can be noticed that a SmartSlider and a SmartGauge have replaced the Slider and the Coolgauge from the previous project.

This project is very similar to the previous one. The SmartSlider doing the Slider's function while the SmartGauge performing the Coolgauge's function.

The Winbutton will again send a REPORT_EVENT to let the host know that it should move to the other project.

The MagicCode and MagicObject from the first project is also included in this one.

The other remaining objects, similar to the previous project, will act as outputs that will be controlled by the host.

Design the Raspberry Pi Projects

Raspberry Pi 3 was used for this project, but this application note procedure should work similarly with other versions. The Raspberry Pi used in this application runs on Raspbian OS.

The latest Raspbian OS can be downloaded here. You can refer to their documentation on how to load the operating system to your Raspberry Pi's SD card.

Once the Raspbian OS is loaded into the Raspberry Pi, you will need to either use a monitor or keyboard to control the RPi directly or use SSH to access it remotely. For the latter, you will need to enable SSH before using it.

Accessing Raspberry Pi

If a monitor and keyboard is accessible, it is easiest to enable SSH by running Raspberry Pi Configuration. Then, you will be able to remotely control it afterwards if you ensure that your computer and the Raspberry Pi is connected to the same network either using Wi-Fi or Ethernet.

You can refer to Raspberry Pi's documentations for the detailed procedure.

The next steps will be using SSH remote access, but should you choose to access the Raspberry Pi directly. The general procedure should be very similar.

Disabling Serial Console

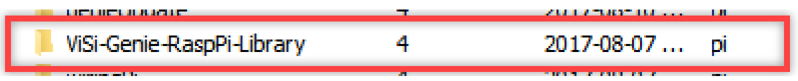
By default, the Serial Port is being utilized by Raspbian as a Serial Console. You will need to disable this feature when using the serial port for other applications like this application note project.

Detailed procedure can be found here.

Download and Install geniePi Library

The geniePi library is available here.

You can simply download the library as a zip file or use clone the repository using git. In this application note, the zip file is downloaded and copied to the Raspberry Pi using FTP.



```
ViSi-Genie-RaspPi-Library    4    2017-08-07 ...    pi
```

Once you have a copy of the library in your Raspberry Pi, you can install it by using the commands:

```
cd ViSi-Genie-RaspPi-Library
make
sudo make install
```

C Code Decision

A typical Raspberry Pi ViSi-Genie application writes values to the display widgets and receives events from user interaction from the display.

This application writes to a Coolgauge and a UserLed continuously when the display is at the first project and writes to a SmartGauge and a UserLed continuously when the display is at the second project.

This is being handled by a thread that runs parallel to the main thread.

```
if (!changingBanks) {

  if (currentBank == PROJECT0) {
    //write to Coolgauge0
    genieWriteObj(GENIE_OBJ_COOL_GAUGE, 0x00, gaugeVal[0]);
  } else if (currentBank == PROJECT1) {
    //write to SmartGauge0
    genieWriteObj(GENIE_OBJ_ISMARTGAUGE, 0x00, gaugeVal[1]);
  }

  gaugeVal[currentBank] += step[currentBank];
  //increment or decrement
  genieWriteObj(GENIE_OBJ_USER_LED,0,gaugeVal[currentBank]%2);

  if (gaugeVal[currentBank] == 99) step[currentBank] = -1;
  if (gaugeVal[currentBank] == 0) step[currentBank] = 1;

  usleep(10000); //10ms delay
}
```

The code checks if the display is not changing banks before writing to the appropriate objects (Gauges and Userled).

```
else {
  int i;
  for (i = 0; changingBanks ; i++) {

    printf(".");
    usleep(10000);

    if (changingBanks && (i % 50 == 0))
    {
      currentBank = genieReadObj(GENIE_OBJ_MAGIC, 0x00);

      if (currentBank != -1) {
      // Query current bank if post genie report event      //
current bank wasn't receive after a second);

        changingBanks = false;
        writePrevValues();
      }
    }
  }
}
```

Otherwise, every 500ms that the display couldn't notify the Raspberry Pi that it is ready, the Raspberry Pi will send a READ_OBJ command. If the display replies, the Raspberry Pi will update the display with the previous values of the active bank.

The REPORT_EVENT commands sent by the display are handled as described in the previous sections. This one checks if the event is from Slider0 then update the LedDigits with the value received.

```
if (reply->object == GENIE_OBJ_SLIDER) {
//check if the object byte is that of a slider

  if (reply->index == 0) {
  //check if the index byte is that of Slider0

    sliderVal = reply->data; //write to the LED digits object
    genieWriteObj(GENIE_OBJ_LED_DIGITS, 0x00, sliderVal);
  }
}
```

This one checks if the event is from SmartSlider0 then update the LedDigits with the value received.

```
else if (reply->object == GENIE_OBJ_ISMARTSLIDER)
{
  if (reply->index == 0) {
    //check if the index byte is that of SmartSlider0
    // write to the LED digits object

    smartSliderVal = reply->data;
    genieWriteObj(GENIE_OBJ_LED_DIGITS,0, smartSliderVal);
  }
}
```

Both above codes are updating a global variable that may be used when writing to the display the previous values when a bank switch is done.

This code checks if the event is from Winbutton0. If true, the Raspberry Pi will tell the display that it is now ready for switching banks by sending a WRITE_OBJ command to the Magic Object.

```
else if (reply->object == GENIE_OBJ_WINBUTTON)
{
  if (reply->index == 0) {
  //check if the index byte is that of Winbutton0
    changingBanks = true;
    printf("\nCurrent Bank: Flashbank%d\n", currentBank);
    currentBank =(currentBank==PROJECT0)?PROJECT1:PROJECT0;
    printf("Switching to FlashBank%d\n", currentBank);
    genieWriteObj(GENIE_OBJ_MAGIC, 0, currentBank);
  }
}
```

Lastly for event handling:

```
else if (reply->object == GENIE_OBJ_MAGIC) {
  if (reply->index == 0) {
    changingBanks = false;
    currentBank = reply->data;
    writePrevValues();
  }
}
```

A REPORT_EVENT is sent by the display to the host at the start of both ViSi-Genie projects to let the host know that it is ready and what bank it is running from.

The code above handles this event and sets the current bank accordingly and also writes the previous values to the display. This will also set the flag *changingBanks* to false allowing the gauge thread to update the widgets accordingly.

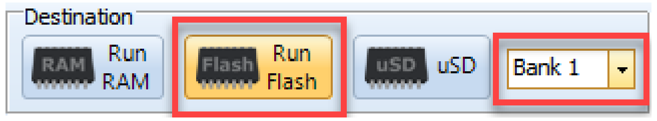
Run the Program

For instructions on how to save a **ViSi-Genie** 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 application note

First ViSi-Genie Project for Diablo16

Downloading the ViSi-Genie Projects

Upload the project to the respective flash banks.

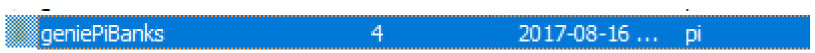


You will need to copy both projects' GCI and DAT files to your uSD Card.

The first project should occupy Bank0 while the second project should occupy Bank1.

Compile and Run the C Program

Included with this application note are the files geniePiBanks.c and Makefile. Create a directory geniePiBanks.



geniePiBanks 4 2017-08-16 … pi

```
cd ~
mkdir geniePiBanks
cd geniePibanks
```

After navigating inside of that folder, copy the Makefile and C file. Compile the C code by using the command:

```
make
```

Proprietary Information

Disclaimer of Warranties & Limitation of Liability