



# ViSi-Genie Magic Password Implementation

DOCUMENT DATE: **30<sup>th</sup> APRIL 2019**  
DOCUMENT REVISION: **1.1**



## Description

This application note shows how to use Magic Objects like Magic Code and Keyboard + ColorPicker Event Magic to be able to print characters on string object from keyboard object and to create a simple Password Implemented Application .

This application note requires:

- Any of the following 4D Picaso and gen4 Picaso display modules:

[gen4-uLCD-24PT](#)    [gen4-uLCD-28PT](#)    [gen4-uLCD-32PT](#)  
[uLCD-24PTU](#)    [uLCD-32PTU](#)    [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and other superseded modules which support the ViSi Genie environment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](http://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 / uUSB-PA5/uUSB-PA5-II](#) for non-gen4 displays(uLCD-xxx)
- [4D Programming Cable & gen4-PA](#) / [gen4-IB](#) / [4D-UPA](#) 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.

## Content

<b>Description</b> .....	<b>2</b>
<b>Content</b> .....	<b>3</b>
<b>Application Overview</b> .....	<b>3</b>
<b>Setup Procedure</b> .....	<b>4</b>
<b>Create a New Project</b> .....	<b>4</b>
<i>Create a New Project</i> .....	<b>4</b>
<b>Design the Application</b> .....	<b>5</b>
<i>Create a new Form</i> .....	<b>5</b>
<i>Object Properties</i> .....	<b>6</b>
StaticText0 and Statictext2 .....	<b>6</b>
StaticText1 .....	<b>7</b>
Strings .....	<b>7</b>
Keyboard .....	<b>8</b>
<i>Adding MagicCode</i> .....	<b>9</b>
<i>MagicCode0</i> .....	<b>9</b>
<i>Adding Keyboard + ColorPicker Event Magic Object</i> .....	<b>10</b>
<i>MagicKbClrEvent0</i> .....	<b>10</b>
<i>MagicKbClrEvent1</i> .....	<b>11</b>
<b>Build and Upload the Project</b> .....	<b>12</b>
<b>Proprietary Information</b> .....	<b>13</b>
<b>Disclaimer of Warranties &amp; Limitation of Liability</b> .....	<b>13</b>

## Application Overview

The application developed in this document works in two forms. First form is to Set/Save a desired password, the second form compares any inputted password to the saved password in the first form.

The Keyboard + ColorPicker Event Magic Object is used to perform gathering of input from Keyboard objects and then stores it in array. The input is also written on a string object. The Magic Object then compares two arrays and if correct it will print "Correct Password" in the string object, and if not it will print "Incorrect Password".

This application note shows how to use Keyboard + ColorPicker Event Magic Object to receive data from a key pressed in a keyboard object and then print it in a string object. The concept of deleting a printed character in a string object here uses a different approach, using a filled rectangle.

## OUTPUT:

SET PASSWORD			AUTHENTICATE		
Enter Your Desired Password(Maximum of 9 Characters)					
<input type="text"/>			<input type="text"/>		
7	8	9	7	8	9
4	5	6	4	5	6
1	2	3	1	2	3
Enter	0	Back	Enter	0	Back

## Setup Procedure

For instructions on how to launch Workshop 4, 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:

[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

## Create a New Project

### 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

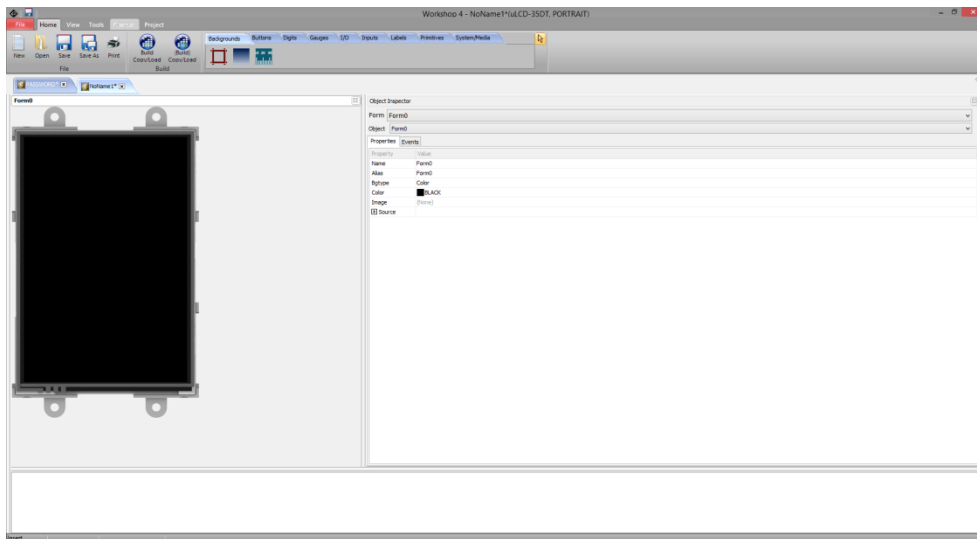
[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

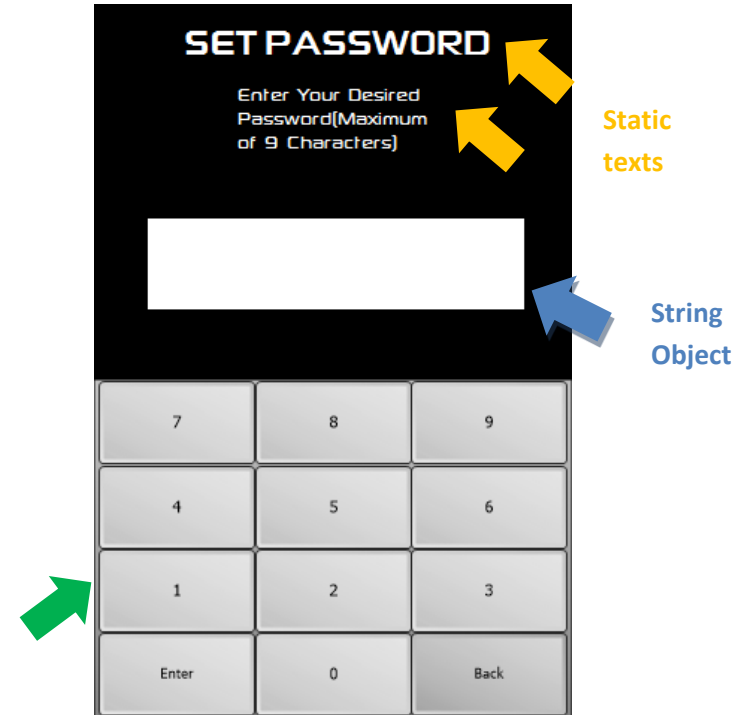
[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

## Design the Application

Everything is now ready to start designing the project. **Workshop 4** displays an empty screen, called **Form0**. A **form** is like a page on the screen. The form can contain **widgets** or **objects**, like sliders, displays or keyboards. Below is an empty form.

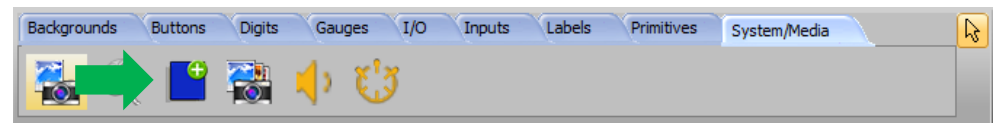


At the end of this section, the user will be able to create a form with four objects. The final form will look like as shown below, with the labels excluded.

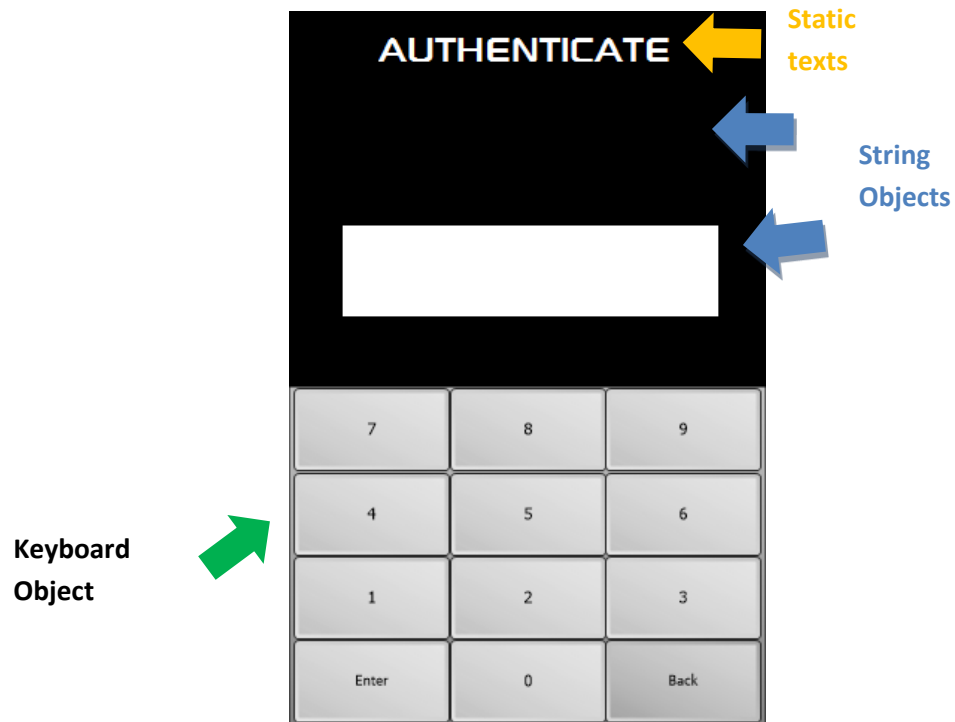


### Create a new Form

The **Form** object will create a new Form. To add a FORM object, go to the **System/Media** pane and select the third icon.



At the end of this section, the user will be able to create a form with four objects. The final form will look like as shown below, with the labels excluded.



## Object Properties

The images shown below are the properties of the objects that are used in the project included in this application note.

### StaticText0 and Statictext2

Form	Form0
Object	Statictext0
Properties	Events
Property	Value
Name	Statictext0
Alias	Statictext0
AutoSize	Yes
Caption	SET PASSWORD
Color	■ BLACK
Font	(WHITE, □, Sony Sketch EF, 24, [Bold])
Height	32
Left	60
Top	8
Transparent	Yes
Width	209
WordWrap	Yes

## StaticText1

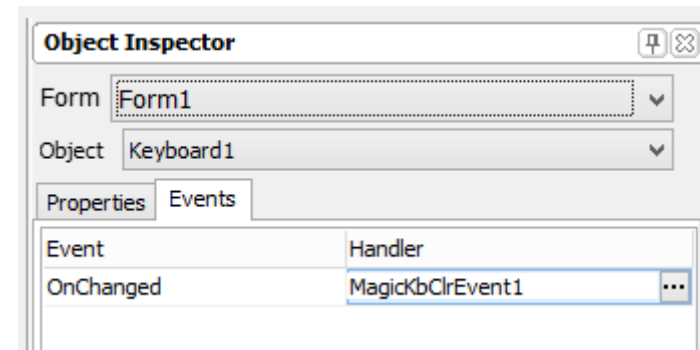
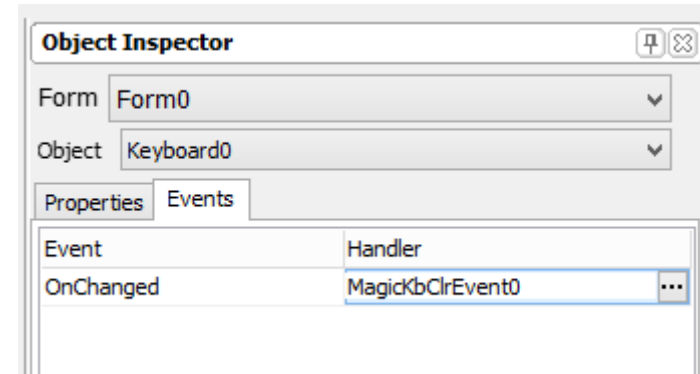
Form	Form0
Object	Statictext1
Properties	Events
Property	Value
Name	Statictext1
Alias	Statictext1
AutoSize	Yes
Caption	Enter Your Desired Password(Maximum of 9 Characters)
Color	■ BLACK
Font	(WHITE, [], Sony Sketch EF, 12, [])
Height	48
Left	96
Top	52
Transparent	Yes
Width	127
WordWrap	Yes

## Strings

Form	Form0
Object	Strings0
Properties	Events
Property	Value
Name	Strings0
Alias	Strings0
Alignment	Left
BGcolor	<input type="checkbox"/> WHITE
Height	61
FGcolor	■ BLACK
Font	
Bold	No
CharSet	ANSI
Italic	No
Name	Sony Sketch EF
Opaque	Yes
Size	34
Strikethrough	No
Underline	No
Left	36
Strings	\n
StringsStyle	Message
Top	144
Width	252

## Keyboard

Property	Value
Name	Keyboard0
Alias	Keyboard0
AutoCapsDisplay	Yes
Fill	(Default)
Font	(BLACK, [], Tahoma, 8, [])
Color	BLACK
Effects	[]
Name	Tahoma
Size	8
Style	[]
Height	228
HighlightCaps	BLACK
KeyboardType	KB1
KeyDistance	0
Left	0
SmallFont	(BLACK, [], Tahoma, 7, [])
Color	BLACK
Effects	[]
Name	Tahoma
Size	7
Style	[]
Top	252
Width	320



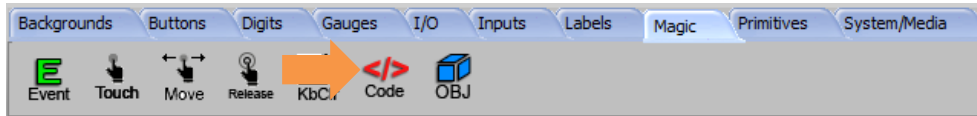
The OnChanged Event of Keyboard0 and Keyboard1 are set to MagicKbClrEvent0 and MagicKbClrEvent1 respectively. This is for MagicKbClrEvent Object to receive key pressed on the keyboard and can be processed later on.

For in depth details on customizing a Keyboard object : [ViSi-Genie Customised Keyboard](#)



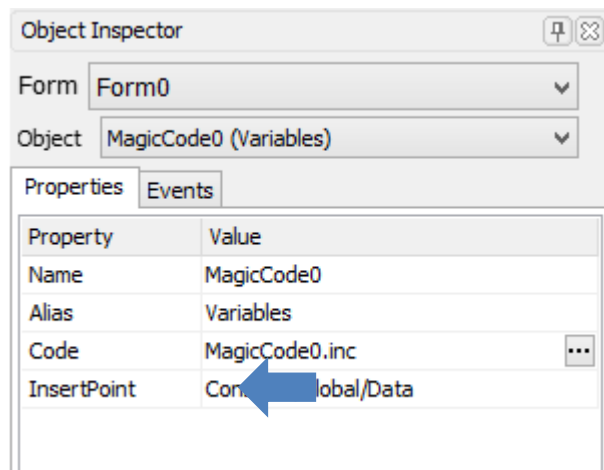
## Adding MagicCode

To add a **MagicCode** object:



Note: To include Magic Objects, Workshop Pro must be activated.

## MagicCode0



The purpose of Magic Code in this application is for the declaration of global variables. That is why 'InsertPoint' property of the MagicCode0 is set to 'Constant/Global/Data'.

```

4 // MagicCode1
5 //
6 var SetPass[9];
7 var Pass[9];
8 var counter, counter2;
9 var fontHeight, fontWidth;
10 var fontHeight2, fontWidth2;
11 var PassFlag;
12
13 #constant Str0x 36
14 #constant Str0y 144
15
16 var firstRunFlag := 1;
17
18 fontHeight := 0;
19 fontWidth := 0;
  
```

The content of MagicCode0 is shown in the image above. The array 'SetPass' is used to store the password set by the user. The array 'Pass' is used to store the password that will be compared to the set password during authentication. Both arrays have a size of 9 since the maximum characters that can be inputted is 9. The variable 'counter', and 'counter' to are used as an indexed for the arrays. The constant variables 'Str0x' and 'Str0y' are the left and top property of the string objects respectively. The variable fontHeight, fontWidth, fontHeight2, fontWidth2 are for coordinates when writing to a character on string objects.

## Adding Keyboard + ColorPicker Event Magic Object

To add a **Keyboard + ColorPicker Event Magic** object:



### MagicKbClrEvent0

```
func rMagicKbClrEvent0(var reportID, var objType, var objHash, var value)
  txt_FontID(hFonts[0]) ;
  txt_FGcolour(BLACK);
  txt_BGcolour(WHITE);
  if((value>=48) && (value<=57) && counter <9) //check of characters a
    SetPass[counter] := value;
    if(firstRunFlag)
      firstRunFlag := 0;
    endif
    if(counter>0)
      fontWidth := fontWidth + charwidth(SetPass[counter - 1]);
    else if(counter == 0)
      fontWidth:=0;
    endif
    putchXY(Str0x + fontWidth, Str0y,SetPass[counter]);
    counter++;
```

This part of the code checks if the key pressed are from 0 to 9. In ASCII Code, the decimal value of '0' is 48 and '9' is 57. The array SetPass will then store the key pressed 'value'. The variable value is the report message of keyboard object to the MagicKbClrEvent. After the value is stored it will then print using putchXY. The variable counter increments every time a key is pressed and is used for the index of the SetPass array. The next key pressed after the first, the program will then compute the total font width of the strings

printed. This will be used for deleting a character using filled rectangle that will be discussed later on.

```
else if(value == 8)
  if(counter)
    gfx_RectangleFilled(Str0x + fontWidth,Str0y,Str0x + fontWidth + charwidth(SetPass[counter - 1])
    ,Str0y + charheight(SetPass[counter - 1]) , WHITE);
    fontWidth := fontWidth - charwidth(SetPass[counter - 2]);
    counter--;
    SetPass[counter] := 0;
  endif
```

The this part of the program checks if the keypressed is 'Back'. In ASCII Code the decimal value of 'backspace' is 8. A rectangle filled will be placed on the latest printed character on the string object. The dimensions of the rectangle is based on the latest printed characters font width and height.

```
else if(value == 13)
  fontWidth := 0;
  counter := 0;
  WriteObject(tForm,1,0);
endif
```

This part of the program checks if the key pressed is 'Enter'. In ASCII Code the decimal value of 'backspace' is 8. If Enter is pressed it will proceed to the next form, the AUTHENTICATION part.

## MagicKbClrEvent1

For the authentication part, the same process with the MagicKbClrEvent0 will be done if a number is pressed or Back. The difference is when 'Enter' is pressed.

```

else if(value == 13)
    txt_FontID(hFonts[2]) ; //set to font of Strings2
    gfx_MoveTo(64, 76) ; //set cursor to position of Strings2
    txt_FGcolour(WHITE);
    txt_BGcolour(BLACK);
    gfx_RectangleFilled(64,76,64+fontWidth2,76+fontHeight2,BLACK);

    for(counter2 := 0; counter2 <= (counter-1) ; counter2++)
        if(Pass[counter2] == SetPass[counter2])
            PassFlag := 1;
        else
            PassFlag := 0;
        endif

        if(PassFlag == 0)
            break;
        endif
    next

```

This part of the program compares the contents of SetPass array with Pass array. The SetPass array holds the value of the user's input from Form0 where the user will decide his/her desired password. The Pass array holds user password input.

```

if(PassFlag == 1)
    print("CORRECT PASSWORD");
    fontWidth2 := strwidth("CORRECT PASSWORD");
    fontHeight2 := strheight();
    pause(2000);
    gfx_RectangleFilled(64,76,64+fontWidth2,76+fontHeight2,BLACK);
else
    print("INCORRECT PASSWORD");
    fontWidth2 := strwidth("INCORRECT PASSWORD");
    fontHeight2 := strheight();
    pause(2000);
    gfx_RectangleFilled(64,76,64+fontWidth2,76+fontHeight2,BLACK);
endif

```

This part of the program prints to the String2. If the password inputted is the same as the password that was set, then it will print "CORRECT PassWord" else it will print "INCORRECT PASSWORD".

## Build and Upload the Project

For instructions on how to build and upload a ViSi-Genie project to the target display, please refer to the section “**Build and Upload the Project**” of the application note

[ViSi Genie Getting Started – First Project for Picaso Displays](#) (for Picaso)

or

[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

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

## 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.