



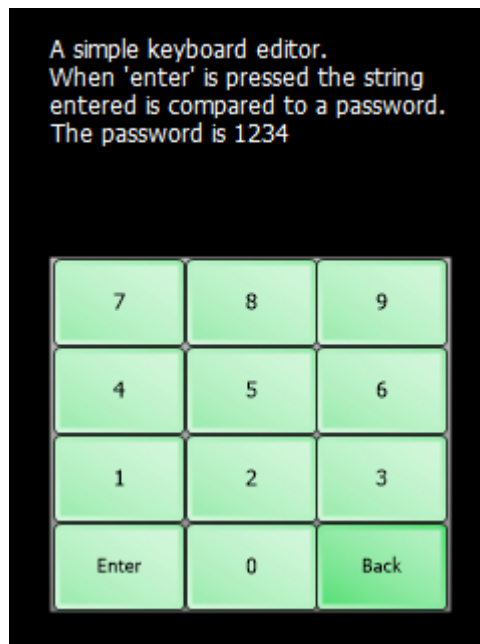
# ViSi-Genie Magic Keyboard Edit

DOCUMENT DATE: **21<sup>st</sup> May 2019**  
DOCUMENT REVISION: **1.1**



## Description

This application note primarily shows how to create a simple keyboard editor using magic Keyboard + ColorPicker Event Magic Object and other magic objects.



**Note 1:**The ViSi-Genie project for this application note is the demo “**KeyboardEdit**”, which is found in Workshop. Go to the File menu -> Samples ->ViSi Genie Magic (Picaso/Diablo16) ->**KeyboardEdit.4DGenie**.

**Note 2:**Workshop Pro is needed for this application.

Before getting started, the following are required:

- Any of the following Picaso display modules:

[uLCD-24PTU](#)  
[gen4-uLCD-24PT](#)

[uLCD-28PTU](#)  
[gen4-uLCD-28PT](#)

[uVGA-III](#)  
[gen4-uLCD-32PT](#)

and other superseded display modules which support the ViSi environment

- The target module can also be a Diablo16 display

[gen4-uLCD-24D](#)  
[Series](#)

[gen4-uLCD-35D](#)  
[Series](#)

[gen4-uLCD-70D](#)  
[Series](#)

[uLCD-35DT](#)

[gen4-uLCD-28D](#)  
[Series](#)

[gen4-uLCD-43D](#)  
[Series](#)

[uLCD-43D Series](#)

[gen4-uLCD-32D](#)  
[Series](#)

[gen4-uLCD-50D](#)  
[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.

- [4D Programming Cable /  \$\mu\$ USB-PA5/ \$\mu\$ USB-PA5-II](#) for non-gen4 displays (uLCD-xxx)
- [4D Programming Cable](#) & [gen4-IB](#) / [gen4-PA](#) / [4D-UPA](#), for gen-4 displays (gen4-uLCD-xxx)
- [micro-SD \( \$\mu\$ SD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)

- 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>4</b>
<b>Setup Procedure .....</b>	<b>4</b>
<b>Create a New Project .....</b>	<b>4</b>
<i>Create a New Project .....</i>	<i>4</i>
<b>Design the Project.....</b>	<b>4</b>
<i>Add a Static Text to Form 0.....</i>	<i>4</i>
<i>Add a Keyboard to Form 0 .....</i>	<i>5</i>
<i>Create a new Form .....</i>	<i>6</i>
<i>Add a Static Text to Form 1.....</i>	<i>6</i>
<i>Add a Fancy Button to Form 1.....</i>	<i>7</i>
<i>Add Magic Code.....</i>	<i>7</i>
MagicCode1 .....	7
MagicCode2 .....	8
<i>Add a Keyboard + ColorPicker Event Magic.....</i>	<i>8</i>
MagicKbClrEvent0 .....	8
<b>Build and Upload the Project.....</b>	<b>10</b>
<b>Proprietary Information .....</b>	<b>11</b>
<b>Disclaimer of Warranties &amp; Limitation of Liability .....</b>	<b>11</b>

## Application Overview

In the past it was not possible to create an application in ViSi-Genie that could print(in a string object) the characters pressed on a keyboard object. This was possible only if a host like an Arduino board receives the keyboard's value and then prints it to a string object. The Keyboard + ColorPicker Event Magic Object was used to make this possible. The application shown in this document will prompt the user to input a password. If the password is correct it will proceed to a different form else if the input password is incorrect then it will print 'Fail' on the display.

## 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

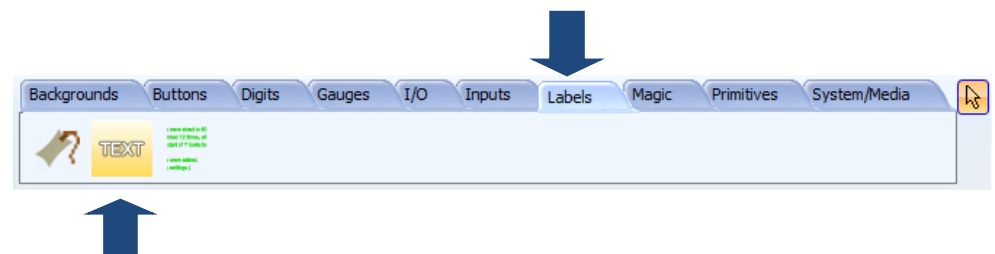
[ViSi Genie Getting Started – First Project for Diablo16 Displays](#) (for Diablo16).

## Design the Project

A uLCD-32PTU (portrait orientation) will be used for this application note.

### Add a Static Text to Form 0

The static text object under the Labels pane can be used to create labels.



A simple keyboard editor. When 'enter' is pressed the string entered is compared to a password. The password is 1234

7	8	9
4	5	6
1	2	3
Enter	0	Back

Property	Value
Name	Statictext0
Alias	Statictext0
AutoSize	Yes
Caption	A simple keyboard editor. \nWhen 'enter'
Color	BLACK
Font	(WHITE, [], Tahoma, 9, [])
Height	56
Left	20
Top	13
Transparent	Yes
Width	198
WordWrap	Yes

Click on the WYSIWYG screen to place the static text object and edit the properties according to the image shown above.

To know more about Static Texts, their properties, and how they are added to a projects, refer to the application note [ViSi-Genie Labels, Texts, and Strings](#).

### Add a Keyboard to Form 0

The keyboard object under the Inputs pane can be used for different applications that needs QWERTY, NUMERIC, CELLPHONE or CUSTOM input.

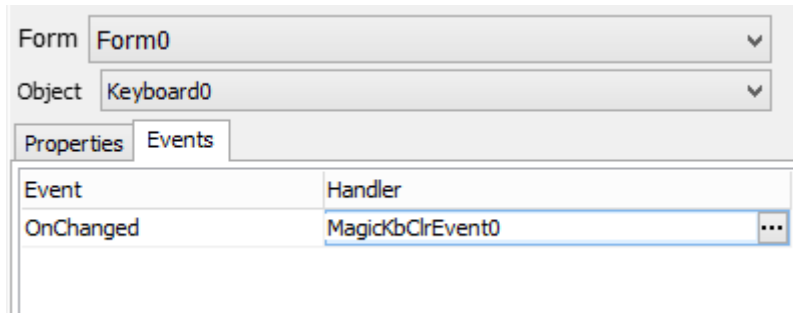
Backgrounds Buttons Digits Gauges I/O Inputs Labels Magic Primitives System/Media

Keyboard

A simple keyboard editor. When 'enter' is pressed the string entered is compared to a password. The password is 1234

7	8	9
4	5	6
1	2	3
Enter	0	Back

Property	Value
Name	Keyboard0
Alias	Keyboard0
AutoCapsDisplay	Yes
Fill	(Default)
Font	(BLACK, [], Tahoma, 8, [])
Height	178
HighlightCaps	BLACK
KeyboardType	KB1
KeyDistance	0
Left	20
SmallFont	(BLACK, [], Tahoma, 7, [])
Top	124
Width	201



The OnChanged event of the keyboard object is set to MagicKbClrEvent0 so that the pressed keys from the keyboard will be received by the MagicKbClrEvent0.

To know more about customizing a Keyboard Object, refer to the application note [ViSi-Genie Customised Keyboard](#).

### Create a new Form

In System/Media pane, click the Form object icon shown below.



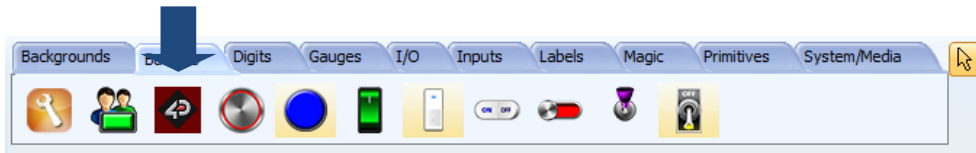
### Add a Static Text to Form 1

Property	Value
Name	Statictext1
Alias	Statictext1
AutoSize	Yes
Caption	Ta Dah!
Color	■ BLACK
Font	(WHITE, [], Tahoma, 28, [])
Height	90
Left	48
Top	76
Transparent	Yes
Width	171
WordWrap	Yes

Click on the WYSIWYG screen to place the static text object and edit the properties according to the image shown above.

## Add a Fancy Button to Form 1

In Buttons pane, click the Fancy Button icon shown below.



Form: Form1  
Object: Winbutton0

Property	Value
Name	Winbutton0
Alias	Winbutton0
Appearance	
AutoSizeToPictureNo	
Bevel	Yes
BevelColor	<input type="checkbox"/> dWhite
Caption	Lock
Color	dGray
Font	(dWindowText, [], Tahoma, 8, [])
Height	35
Left	64
Matrix	-1
Momentary	Yes
Picture	(None)
StatusWhenOf	
StatusWhenOn	
Top	256
Width	120

The OnChanged event of Winbutton0 is set to 'Form0Activate' to be able to return to Form0 if the button is pressed.

To know more about Fancy buttons, refer to the application note: [ViSi-Genie Customised Keyboard](#).

## Add Magic Code

To know more about adding Magic Objects, refer to the application note [ViSi-Genie How to Add Magic Objects](#).

## MagicCode1

```

1 //
2 // Added 3/11/2014 2:46:14 PM
3 //
4 // MagicCode1
5 //
6
7 func resetchars ()
8     txt_MoveCursor(6, lft);
9     print("      ");
10    txt_MoveCursor(6, lft);
11 endfunc
12
13 #constant maxresp 4
14 #constant lft 10
15 //
16 var respchp, chars ;
17 var Response[3] ;

```

The image shown is the 4DGL code inside of the Magic Code object. MagicCode1 is the declaration of variables, constant and resetchars() function.

The 'maxresp' constant is the maximum number of input for the password. The 'lft' constant is the start position for echo of \*s. And the resetchars function is a timer routine used to blank 'Failed' from the screen.

### MagicCode2

```
// init variables used on form0
if (CurrentForm == 0)
    respchp := str_Ptr(Response) ;
    chars := 0 ;
endif
```

The image above shows the content of MagicCode2. This magic code's insert point is set to 'PostActivateForm'. This checks if current form is Form0 and then set keyboard response to start.

### Add a Keyboard + ColorPicker Event Magic

To know more about adding Keyboard + ColorPicker Event Magic, refer to the application note [ViSi-Genie How to Add Magic Objects](#).

### MagicKbClrEvent0

This magic object receives the value inputted on the keyboard object. The value is stored in 'var value'. If there are multiple keyboard objects in a project, then their 'OnChanged' event should be set to the 'Keyboard + ColorPicker Event Magic' that will handle the input value.

```
func rMagicKbClrEvent0(var reportID, var objType, var objHash, var value)
var i ;
if (value == 8) // 'back'
    if (chars)
        respchp-- ;
        chars-- ;
        txt_MoveCursor(6,lft+chars);
        putch(' ');
        txt_MoveCursor(6,lft+chars);
    endif
```

The image shown above is a part of the content of MagicKbClrEvent0. This part of the program checks if the input value from Keyboard0 is 'Back'. The character 'Back' of keyboard has a decimal value '8'. This simply checks if a character is already printed and then replaces it with space ' ' after moving the text cursor to the character that will be deleted.



```

else if (value == '*') // 'enter'
    str_PutByte(respchp++, 0) ;
    i := str_Ptr(Response) ;
    txt_MoveCursor(6, lft) ;
    if (str_Match(&i, "1234"))
        print("Match") ;
        ActivateForm(1) ;
    else
        txt_FGcolour(RED) ;           // 0 t
        print("Fail") ;
        txt_FGcolour(LIME) ;         // 0
        sys_SetTimerEvent(TIMER0, resetchars) ;
        sys_SetTimer(TIMER0, 500) ;
    endif
    respchp := str_Ptr(Response) ;
    chars := 0 ;

```

Next, the image shown above is the part of MagicKbClrEvent0.inc that will execute if 'Enter' is pressed on the keyboard. This simply uses str\_Match to compare the content of the array 'Response' to "1234". If it does match then it will proceed in activating FORM1. Else if the content of 'Response' array does not match "1234" then it will print "Fail" then perform the resetchars() timer routine.

```

else if (chars < maxresp)
    str_PutByte(respchp++, value) ;
    txt_MoveCursor(6, lft+chars) ;
    chars++ ;
    putchar(value) ;
    putchar('*') ;
endif

```

Lastly the image shown above is part of MagicKbClrEvent0.inc that will execute if numbers 0-9 is pressed on the keyboard object. This increases the column of the cursor and then prints asterisk '\*'. if you want to print the value then uncomment "putch(value)" then comment out "putch('\*)".

To know about 4DGL programming please refer to [4DGL Programmer's Reference Manual](#) and [GOLDELOX, PICASO, DIABLO Internal Functions Manual](#).

## 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.