VISI-GENIE MAGIC

Comms 'NIL' Option

Application Note

Revision 1.0

Contents

1. Description

This application note discusses the use of the 'nil' option for Comms selection. This option is useful if you do not want to use or if you need to customize the ViSi-Genie protocol for your application.

The Comms -Nil- option allows you to either turn off the ViSi-Genie command protocol or modify it according to your needs.

This application requires the PRO version of Workshop4 to access the ViSi-Genie Magic feature.

2. Prerequisites

It is recommended to have read and tried the following application notes including their own prerequisites prior to learning this application.

- ViSi-Genie: Getting Started with Picaso Displays
- ViSi-Genie: Getting Started with Diablo16 Displays
- ViSi-Genie: Getting Started with Pixxi Display Modules
- ViSi-Genie: onChanging and onChanged Events
- ViSi-Genie Magic: Code Insertion Points

3. Application Overview

To communicate with 4D Systems displays on ViSi-Genie environment using UART serial, there are 3 options that we can use.

1. COM0 which is the using TX0 and RX0. This is also used when programming the displays.
2. COM1 which can be used from the additional TX1 and RX1 pins available. For Diablo16 displays, RX1 and TX1 can be assigned (refer to Diablo16 display datasheet for supported pins).
3. NIL – which stops the comms.

Setting the comms to 'nil' will stop the display from sending the Genie protocol. In this case, you can use magic code to modify the Send Report function that the display will send to the host controller. For example, if you want to add a specific start character or other information from the host controller.

You can also make your own function to capture any information from the host controller.

4. 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 notes:

- ViSi-Genie: Getting Started with Picaso Displays
- ViSi-Genie: Getting Started with Diablo16 Displays
- ViSi-Genie: Getting Started with Pixxi Display Modules

There is an example from the Wokrshop4 IDE called `GenieCommsManual` that shows how to use the Nil option. To open, go to, *File > Samples > Picaso/Diablo/Pixxi ViSi-Genie > GenieCommsManual*.

This application note discusses the procedure on how to create a Comms -Nil- ViSi-Genie project similar to the WS4 project specified.

5. Creating a New Project

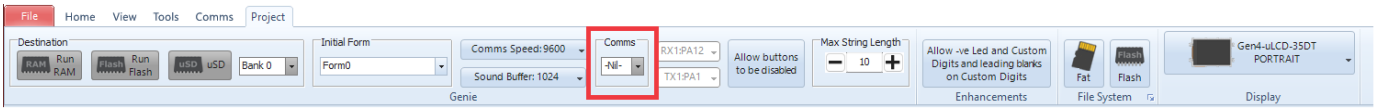
For instructions on how to create a new ViSi project, please refer to the section **Create a New Project** of the application notes:

- ViSi-Genie: Getting Started with Picaso Displays
- ViSi-Genie: Getting Started with Diablo16 Displays
- ViSi-Genie: Getting Started with Pixxi Display Modules

6. Design the Project

6.1. Setting Comms to NIL

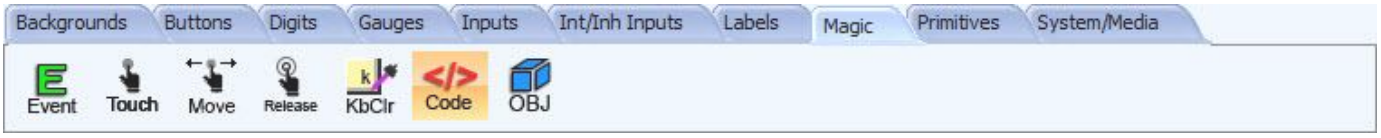
After creating a new ViSi-Genie project and designing your user interface, you can set the Comms option to -Nil- by going to *Project Menu* > *Comms* then from the dropdown select the **-Nil-** option.



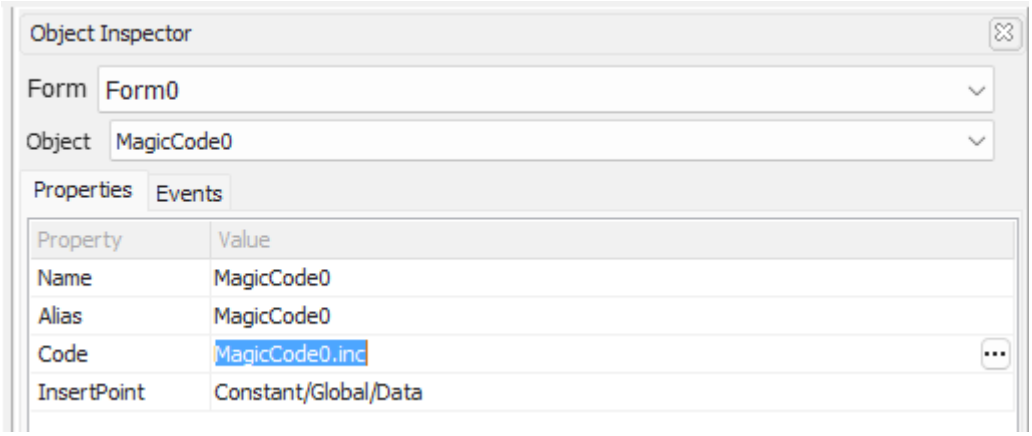
> **Note**
>
> This option requires that the `SendReport` function is added manually to a MagicCode. The MagicCode insert point should be set to Constants/Global/Data.

6.2. Constants/Global/Data

Add MagicCode by selecting the MagicCode as shown:



Open the code MagicCode0.inc by clicking **...** button.



The MagicCode0 file will open and show an empty 4DGL code.

When using Comms -Nil- option, it is required that the `SendReport` function is added to Global code.

```
//
// SendReport is always needed, if you never send reports, this routine should be 'null'
//
func SendReport(var id, var objt, var objn, var val)
    // By making this empty (null), Report Event in widget event options does nothing
endfunc
```

The SendReport function should take four (4) arguments:

id

This is the command ID which is either REPORT_EVENT or REPORT_OBJ. In most cases, REPORT_EVENT is used as it is triggered by touch inputs from widgets.

objt

This is the type of widget the triggered the report.

objn

This is the index number of the widget.

val

This is the new value of the widget.

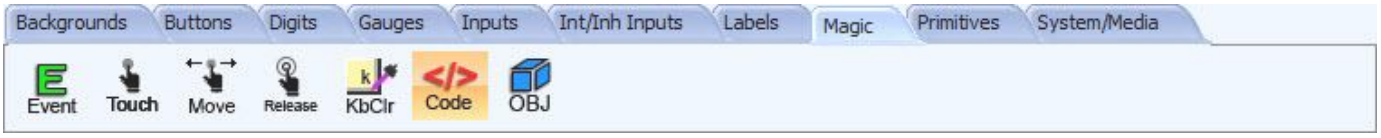
The SendReport function can be modified by adding information to the packet that will be sent to the host or make it 'null' if you do not use send report.

For this application note, the ViSi-Genie protocol is added manually to demonstrate how to customize the reporting after disabling.

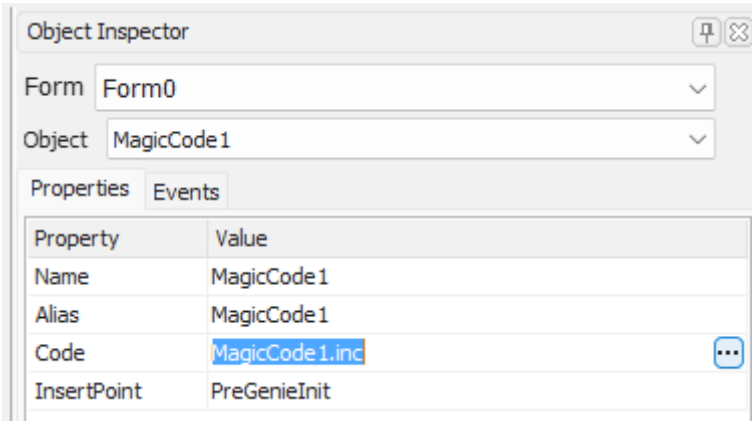
```
#CONST
    CMDLenMAX       80  // 80 is used for 'default' max string length of 75
    CommsBuffLen    40  // 40 is used for 'default' max string length of 75
                        //    (i.e. half of CMDLenMAX)
#END

var comRX[CommsBuffLen], cmd[CMDLenMAX] ;
var InputCS, OutputCS ;

func seroutCS(var op)
    serout(op) ;
    OutputCS ^= op ;
endfunc

func nak0()
    serout(NAK) ;
    InputCS := 0 ;
endfunc

func seroutOcs()
    serout(OutputCS) ;
    OutputCS := 0 ;
endfunc

//
// SendReport is always needed, if you never send reports, this routine should be 'null'
//
func SendReport(var id, var objt, var objn, var val)
    seroutCS(id) ;
    seroutCS(objt) ;
    seroutCS(objn) ;
    seroutCS(val >> 8) ; // first 8 bits
    seroutCS(val) ;
    seroutOcs() ;
endfunc
```

6.3. PreGenieInit

Since the project aims to manually add the ViSi-Genie Serial protocol, the Serial comms needs to initialized manually as well. To do this, add another MagicCode to the project and set its insert point to **PreGenieInit**.



Open the code MagicCode1.inc by clicking **...** button.



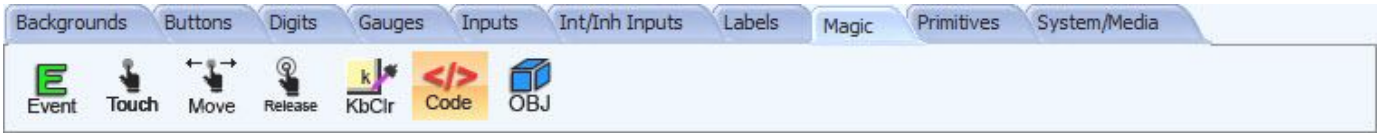
The MagicCode1 file will open and show an empty 4DGL code.

You can put additional variables and functions needed by the application. In this case, two variables used by default comms are included. Initialize the comms, set baudrate and TXBuffer.

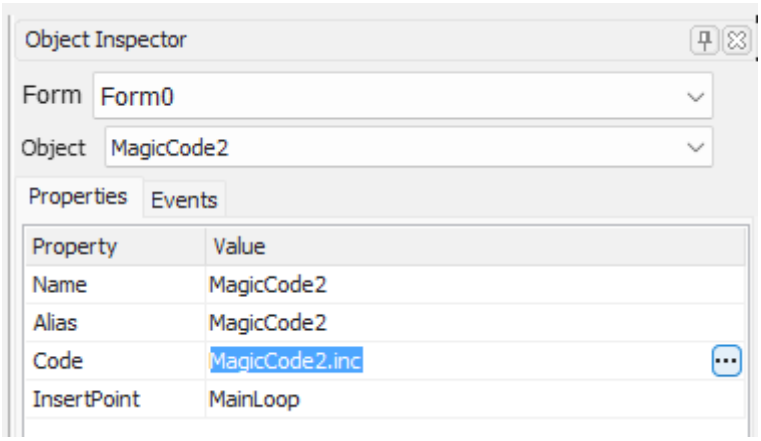
```
//
// Two variables used by the default comms. You might not want/need them,
// or you might want them as global variables.
//
var comTX[50], cmdi ;

com_Init(comRX,CMDLenMAX,0);
com_SetBaud(COM0,960);
com_TXbuffer(comTX, 100, 0);
```

6.4. Mainloop

Finally, to complete the manually added the project aims to manually add the ViSi-Genie Serial protocol, the commands coming from the host controller needs to be handled manually as well. To do this, add another MagicCode to the project and set its insert point to **MainLoop**.



Open the code MagicCode2.inc by clicking **...** button.



The MagicCode2 file will open and show an empty 4DGL code.

In the main loop, we can add handling of commands coming from the host controller as shown:

```
//
// Standard/simplified comms input,
// does not handle Magic Objets, Unicode Strings, strings, labels, inherent widgets,
// and internal buttons.... Just a simple example!
//

// check comms for command, how to NAK invalid command
if (com_Count() != 0)
    i := serin() ;
    InputCS ^= i ;               // update checksum
    cmd[cmdi++] := i ;
        if ((cmd[0] == READ_OBJ) && (cmdi == 4))
        if (InputCS)
            nak0() ;
        else
            ReadObject(cmd[1], cmd[2]) ;
        endif
        cmdi := 0 ;
    else if ((cmd[0] == WRITE_OBJ) && (cmdi == 6)) // 6 byte write command (gen option)
        if (InputCS)
            nak0() ;
        else
            WriteObject(cmd[1], cmd[2], cmd[3] << 8 + cmd[4]) ;
            serout(ACK) ;
        endif
        cmdi := 0 ;
    else if ((cmd[0] == WRITE_CONTRAST) && (cmdi == 3))
        if (InputCS)
            nak0() ;
        else
            gfx_Contrast(cmd[1]) ;
            serout(ACK) ;
        endif
        cmdi := 0 ;
    else if (cmdi == 6)    // we have 6 bytes and we've gotten here -> something wrong
        nak0() ;
        cmdi := 0 ;
    endif
endif   // a character is available
```

7. Build and Upload the Project

To test this project, add some input widgets (buttons, slider or knobs) and set the events to Report Event as you would in a simple ViSi-Genie project. You can refer to the application note ViSi-Genie: onChanging and onChanged Events

for more information.

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 with Picaso Displays
- ViSi-Genie: Getting Started with Diablo16 Displays
- ViSi-Genie: Getting Started with Pixxi Displays

The uLCD-32PTU and/or the uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other ViSi-Genie compatible displays. For this application note, gen4-uLCD-35P4T was used.

8. Legal Notice

8.1. 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. 4D Systems reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

All trademarks belong to their respective owners and are recognised and acknowledged.

8.2. Disclaimer of Warranties & Limitations of Liabilities

4D Systems makes no warranty, either expressed 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.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Systems range of products, however the quality may vary.
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.