GNU SPICE GUI
(gSpiceUI)

User Manual

M.S.Waters

Version 1.1.03 (2021-08-29)

Contents

List of Figures

1 Introduction

If you are "of an impatient temper", you may stop reading now and go straight to section *2 Getting Started*, it's written with you in mind. For everyone else, read on.

The name *gSpiceUI* is an abbreviation of the project title *GNU SPICE GUI*, which is itself an acronym standing for **G**nu is **N**ot **U**nix, **S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis, **G**raphical **U**ser **I**nterface.

This document has become a central repository for all things gSpiceUI. Primarily it is still *user documentation* but it also includes material more appropriate to project design and management. Rather having many documents, some of which get forgotten, I'm hoping by using one document it will be easier for me to keep things more organised and up to date.

No GUI can encapsulate all the functionality of a non-trivial command-line application and electronic simulation engines are definitely not trivial. gSpiceUI doesn't attempt to plumb anything like the depths of the simulation engines it interfaces to. Rather, applying the Pareto principle (80/20 rule) to software, it's been said that 20% of the available functionality is used 80% of the time. The design of gSpiceUI attempts to encapsulate as much of that 20% of simulation engine functionality as possible. Thereby delivering as much high usage functionality as possible without making the development process too onerous.

gSpiceUI is intended to provide a GUI front-end for freely available electronic circuit simulation engines ie. *NG-SPICE* and *GNU-CAP*. The user opens a schematic (which is converted to a netlist) or netlist file, enters analysis information and gSpiceUI generates the necessary simulation engine instructions and appends them to a netlist file. The netlist file, now including the simulation instructions is then run by the simulation engine and a results file is generated.

Basically, the only function gSpiceUI performs itself is to generate simulation engine instructions. All other functionality is provided by other freely available applications and utilities written and supported by other authors. An EDA (**E**lectronic **D**esign **A**utomation) tool suite, usually *gEDA-gaf* or *Lepton-EDA* takes care of schematic capture and netlisting. Waveform data results can be displayed using *Gaw*, *Gwave* or *Kst*. PDF documentation is viewed using one of many utilities eg. evince, xpdf, mupdf, etc., depending on what is installed on your system.

gSpiceUI can also be used in conjunction with the *KiCAD* EDA tool suite. KiCAD doesn't expose it's netlister utility like Lepton-EDA or gEDA-gaf so this step must be performed by the user inside the schematic capture application *eeschema*. The import mechanism in gSpiceUI cannot be used in this case and eeschema can't be envoked inside gSpiceUI. Gaw, Gwave or Kst are still used to display simulation results.

The gSpiceUI GUI is designed to be as consistant as possible between analysis types and simulation engines. The intention is to abstract the technical particulars from the user so that the focus may be on the simulation rather than the simulation engine or even the analysis type.

Like all open source software gSpiceUI is a work in progress and probably always will be. It has been developed over a long period of time (since 2003) and has reached a relatively mature state. Experience in software development as well as using software in general has taught me that it's better to provide less functionality that works than more functionality that just promises a lot. Consequently, as a generally rule, bugs fixes and improvements

to the underlying architecture will take precedence over new functionality. I believe this approach pays off in the long term.

It is worth noting that NG-SPICE is derived from the SPICE code base, whereas GNU-CAP is not. GNU-CAP is an independent implementation of the principals used to analyses electronic circuits. Consequently analysis results can be generated using two independant mechanisms and then compared. This is one of the reasons for supporting more than one simulation engine; if the same or similar results can be achieved using two different mechanisms then it's highly likely that the results will bear some resemblence to what actually happens in the real world.

It is important to realise that a simulation engine *models* the behaviour of a system and therefore can never be 100% accurate. In addition the simulation engine itself is an implementation in software of a mathematical model and so is also subject to bugs. As the underlying mathematical model and software implementation improve, the simulation performance improves. One of the reasons for the choice of simulation engines to support was that they are both in constant development. Not to mention the inherent goodness and righteousness of open source software (so I won't mention it.)

The skills and judgement of the user have a profound impact on the usefulness of electronic simulation. It can be a technically demanding process requiring considerable thought, practice and persistence. The alternative is to go straight to building prototypes; in any case, ultimately prototypes must be created to finalize a design. However simulation is generally a rewarding step in the development process, even if it only provides a better understanding of the internal workings of a design.

2 Getting Started

At first glance, I'm told gSpiceUI can appear unintuitive. This section is aimed at trying to dispell some of the mystery. When first encountering gSpiceUI you might expect to see a electronic circuit schematic and to be able to click on any component and see simulation results displayed in a graph. This is what I would've liked before starting development of gSpiceUI but there just wasn't anything of the kind available for Linux.

What did exist, however, where simulation engine backends, schematic capture applications and waveform data viewer utilities. What was missing was something to simplify the interface to the simulation engine backend. That's the function I intended gSpiceUI to provide. However, over the years it has also evolved to become center point for tying together useful functionality.

Basically, all gSpiceUI does is generate simulation engine instructions. All other functionality is provided by other applications and utilities. An EDA tool suite, usually gEDA-gaf or Lepton-EDA takes care of schematic capture and netlisting. Waveform data results can be displayed using Gaw, Gwave or Kst. PDF documentation is viewed using one of many utilities eg. evince, xpdf, mupdf, etc., depending on what is installed on your system.

Like most applications gSpiceUI has the usual controls along with various application specific regions. Refer to *Figure 1 : gSpiceUI Main Window* for an image of gSpiceUI with the various regions of it's main window explicitly labeled.

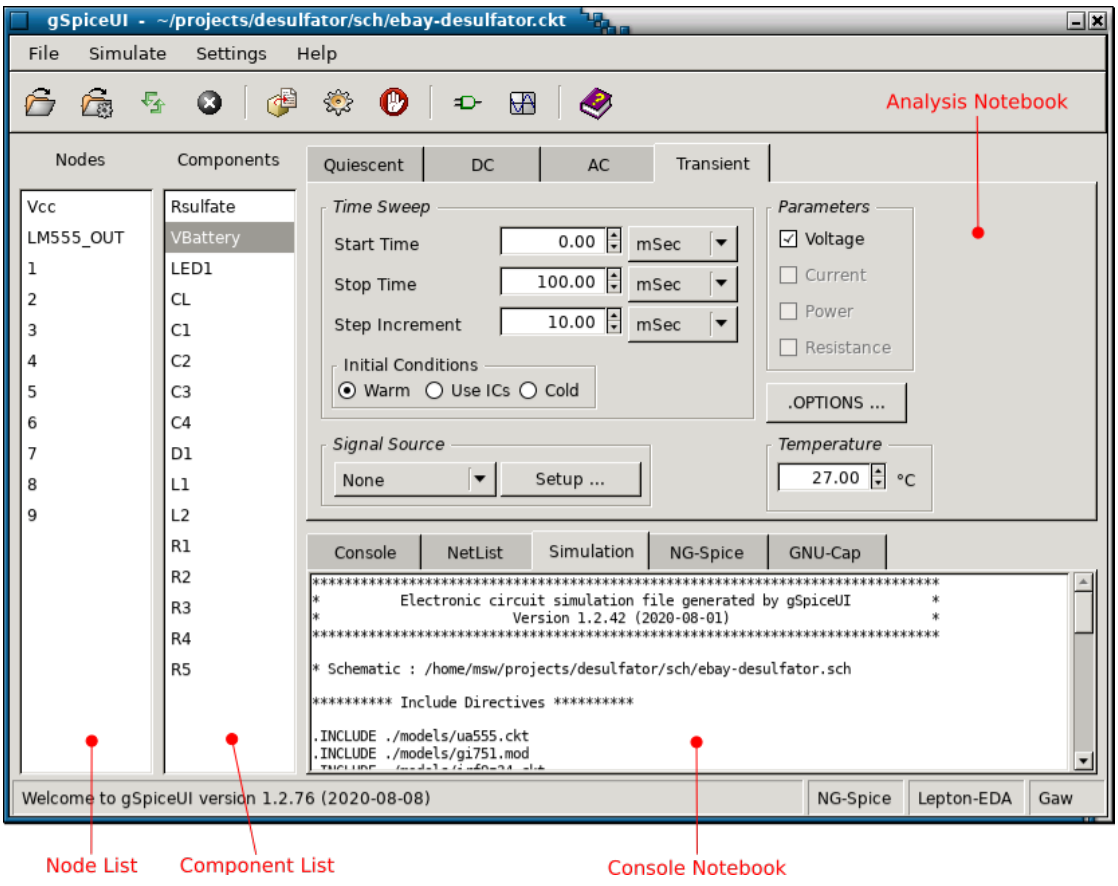

Figure 1: gSpiceUI Main Window.

The purpose for the different application specific regions is briefly describe below :

Node List : A list of node labels extracted from the netlist file. One or more nodes may be selected (or deselected) by clicking on them with the mouse. The probes for the selected nodes will be calculated and included in the simulation results.

Component List : A list of components extracted from the netlist file. One or more components may be selected (or deselected) by clicking on them with the mouse. The probes for the selected components will be calculated and included in the simulation results. At present only two terminal components are included in this list.

Analysis Notebook : The analysis notebook contains pages reflecting the various analysis types supported by the chosen simulation engine. Controls on each allow the entry of parameters such as sweep range, steps size and signal source component.

Console Notebook : The console notebook contains various text controls which display text data generated by the different operations initiated by gSpiceUI.

Now to try a simple example. As part of the gSpiceUI installation a number of demonstration schematics should have been included (usually in "/usr/share/gspiceui/sch/demos/"). We'll open one and do a simple simulation to illustrate the basic functionality of gSpiceUI. Follow the steps listed below :

1. Start gSpiceUI using the following command-line (to simplify these instruction use the NG-SPICE simulation engine and Gaw as the waveform data viewer) :

   ```
   gspiceui -s ngspice -w gaw
   ```

2. Import (create a netlist file) from the schematic file "bjt-amp-ce-1.sch" found amoung the demo. schematics. The *Import* operation can be initiated from the *File* menu or by clicking on the the tool bar icon with the tool tip "Import a schematic file". If all goes well the file should load without errors.

3. View the schematic by selecting the *Schematic* item from the *Simulate* menu or clicking on the tool bar icon with the tool tip "View / edit the schematic".

4. Prepare a simulation by first selecting the *AC* analysis page in the Analysis Notebook and entering the sweep parameters : *Start Frequency* 100 Hz, *Stop Frequency* 500 kHz and 100 *Steps/Decade*. Select the *Signal Source* to be *Vin* and the output probe *Rout* from the *Components* list.

5. Now run the simulation by selecting the *Run* item from the *Simulate* menu or clicking on the tool bar icon having the tool tip "Run the simulation". If all goes well the simulation will run without error and the message "Simulation ran successfully" will appear in the left panel of the status bar.

6. To view the simulation results select the *Results* item from the *Simulate* menu or click on the tool bar icon having the tool tip "View simulation results".

7. Gaw uses a MDI (Multiple Document Interface) comprised of a smaller window containing a list parameters and a larger waveform display window. Select the parameter

VDB(Rout) and drag it to one of the waveform display regions. From the *Preferences* menu select *Show Grid* and *log x scale*. You should now see a bode plot of the amplifier transfer function and be able to confirm the specifications given in the schematic file. Refer to *Figure 2 : Gaw Main Window*.

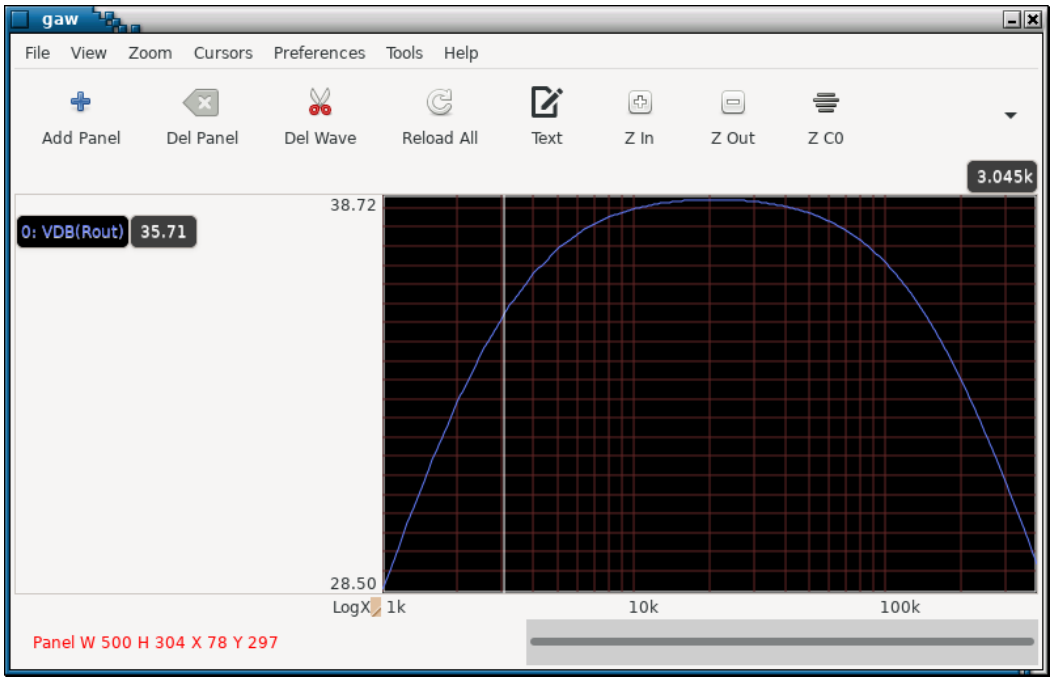

Figure 2: Gaw Main Window.

The gSpiceUI Help menu contains links to user documentation for the simulation engines (assuming it has been installed in the appropiate place on your system, refer to the Install file).

3 Installation

These instructions provide information required to compile and install gSpiceUI. Many operating systems have ready made packages for installing gSpiceUI, this is the preferred option.

3.1 Requirements

There are various requirements for building gSpiceUI and running it in a meaningful way. The different requirements are listed below :

- Compilation (Essential) : wxWidgets - C++ library
- Run time (Desirable) : GNU-CAP - electronic circuit simulation engine
- Run time (Desirable) : NG-SPICE - electronic circuit simulation engine
- Run time (Optional) : Gaw - analogue waveform data viewer
- Run time (Optional) : GWave - analogue waveform data viewer
- Run time (Optional) : Kst - analogue waveform data viewer
- EDA (Optional) : gEDA-gaf - EDA tool suite
- EDA (Optional) : Lepton-EDA - EDA tool suite
- Documentation (Optional) : NG-SPICE - user manual
- Documentation (Optional) : GNU-CAP - user manual

Notes :

- On Linux systems the NG-SPICE user manual is installed in :

  `/usr/share/doc/ngspice/manual.pdf`

- On Linux systems the GNU-CAP user manual is installed in :

  `/usr/share/doc/gnucap/gnucap-man.pdf`

3.2 wxWidgets Library

gSpiceUI is written in C++ and makes use of the wxWidgets library. The wxWidgets library provides all the functionality required by gSpiceUI in one library, thereby reducing external dependencies to one package. Is also possible to compile the same source code under Linux/UNIX, MS Windows, OSX and various other platforms. This library must be installed before gSpiceUI can be compiled.

Many systems now come with a version of wxWidgets pre-installed or provide it as a pre-built package which can be installed using the systems packaging system, this is the preferred option. There are circumstances however where building and installing the wxWidgets library from it's sources may be necessary. One such instance may be if the ABI (Application Binary Interface) of the pre-built wxWidget library package differs from the ABI used by the compiler used to build gSpiceUI. The ABI version using when building gSpiceUI can also be set within the gSpiceUI file src/Makefile.

The wxWidgets library home page is :

`http://www.wxWidgets.org/`

The recommended version of the wxWidgets library to use is the v3.0.x series. The archive file for a Linux based system is :

```
wxWidgets-3.0.x.tar.bz2
```

Untar the wxWidgets archive and change to the wxWidgets root directory using the following commands :

```
tar -jxvf wxWidgets-3.0.x.tar.bz2
cd wxWidgets-3.0.5
```

Installation of wxWidgets is based around autoconf and automake, so it should be straight forward. If necessary refer to the installation instructions provided with the wxWidgets sources (in the docs directory). Enter the following commands in the wxWidgets root directory :

```
mkdir my-build
cd my-build
../configure --without-subdirs --disable-compat28
make
su
Password:         <-- enter the root password
make install
```

Notes :

- To view all available options to the `configure` script enter :

  ```
  ../configure --help
  ```

- Some options to the `configure` script of particular interest are :

`--prefix=<DIR>`	(Installation directory, default /usr/local)
`--without-subdirs`	(Don't generate makefiles for "`samples/demos/...`")
`--disable-compat28`	(Disable wxWidgets 2.8 compatibility)
`--enable-debug`	(Build the library for debugging)
`--enable-debug_info`	(Create code with debugging information)
`--enable-debug_gdb`	(Create code with extra GDB debugging information)
`--disable-debug_flag`	(Disable debugging support, unset `__WXDEBUG__` flag)
`--with-gtk[=VERSION]`	(Use GTK+, `VERSION` can be 3, 2 (default), 1 or "any")

  At the time of writing there appears to be display problems when wxWidgets is compiled against GTK3, so building against GTK2 is recommended. Alternatively some distributions offer a compatibility layer, eg. for Fedora there is a package called "compat-wxGTK3-gtk2".

- The wxWidgets library can be uninstalled as follows :

  ```
  sudo make uninstall
  ```

3.3 Build System

After the wxWidgets library has been installed properly, go to the gSpiceUI root directory and enter the following :

```
make          (for Linux and Windows)
  or
gmake         (for FreeBSD)
  or
make maccfg   (for MAC OSX)
make
```

Note :

- Apple MAC OSX users are required to execute the extra target (ie. `make maccfg`) to create some directories and copies some files specific to their system, it is only required once per installation.

If all goes well the binary file "`gspiceui`" should have been generated in the directory `<ROOT>/bin`.

3.4 Installation

After building gSpiceUI, it may be installed into the default directory `/usr/local/` by entering the following command as root :

```
make install
  or
gmake install (for FreeBSD)
```

The application binary is intalled in `/usr/local/bin/`. The documentation and other supporting files are installed in `/usr/local/share/gspiceui/`.

The application may be uninstalled by entering the following command as root :

```
make uninstall
  or
gmake uninstall (for FreeBSD)
```

An alternative install path may be specified by manually changing the make variable `DESTDIR` in the main Makefile or by specifying it on the command-line as the following example illustrates :

```
make install DESTDIR=/alternative/install/directory
```

The same applies for the uninstall operation :

```
make uninstall DESTDIR=/alternative/install/directory
```

3.5 Microsoft Windows Support

This section provides a broad outline of how to build gSpiceUI on MS Windows. The methodology has been developed and tested on Windows 7 and may work on other releases. Detailed instructions are not provided since this information can change and so is best sourced from the Web sites providing the required packages which are outline below :

- *MinGW* (Minimalist GNU for Windows) is a minimalist development environment for native Microsoft Windows applications.
- *MSYS* (Minimal SYStem) is a Bourne Shell command line interpreter system. It's an alternative to Microsoft's `cmd.exe` providing a general purpose command line environment particularly suited to use with MinGW.
- *wxWidgets* is a C++ library that lets developers create applications for MS Windows, Apple Mac OSX, Linux and other platforms with a single code base.

An article written by Oliver Kohl *Using wxWidgets under Windows* was very helpful in formulating these instructions. The article can be found at :

`http://www.codeproject.com/Tips/630542/Using-wxWidgets-under-Windows`

Note : The MinGW environment can be accessed from Windows, outside of the MSYS shell. Paths containing forward slashes ('/') are from within MSYS and paths containing backslashes ('\') are from Windows. Eg. the path to my home directory within MSYS is `/home/msw`, whereas from MS Windows it's `C:\MinGW\msys\1.0\home\msw`.

The steps to building gSpiceUI for Windows are as follows :

1. Install MinGW and MSYS :
   - Go to `http://www.mingw.org/` and retrieve the MinGW installer :
     `mingw-get-setup.exe`
   - Run the installer and select at least the following packages : C & C++ compilers and MSYS. If MinGW is installed in the root of your system drive (ie. `C:\MinGW`) you'll have no problems with navigation and paths.
   - The installer will retrieve the selected packages from the Web and install them.
   - Ensure that the file `C:\MinGW\msys\1.0\etc\fstab` exists and contains at least the following line, followed by an empty line :
     `C:\MinGW  /mingw`
   - Now navigate to the directory `C:\mingw\msys\1.0` and run msys.bat, this creates your home folder.
   - It is worth creating a start menu item and/or a desktop icon for msys.bat.
2. Install the wxWidgets sources :
   - Go to the web site `http://www.wxwidgets.org/` and retrieve the wxWidgets sources eg. `wxWidgets-3.0.5.zip`.
   - Create a folder (eg. `C:\wxWidgets`) and unpack the wxWidgets sources archive into it.
3. Build wxWidgets using MinGW :

- Open MSYS and go to the wxWidgets sources root directory eg. :
  ```
  cd /c/wxWidgets/wxWidgets-3.0.5
  ```
- Create the build folder and navigate into it eg. :
  ```
  mkdir build-msw
  cd build-msw
  ```
- Run the wxWidgets configure script which creates the make files and sets system dependent variables :
  ```
  ../configure --build=x86-winnt-mingw32 --disable-shared --disable-threads
  ```

  Notes :
  - My AVG virus scanner breaks the configuration process when the TIFF library is processed. There are two solutions : disable the TIFF library by adding the option `"-without-libtiff"` to the above command line or temporarily disable the AVG Resident Shield component.
  - The build process can be sped up if the wxWidgets samples and demos are skipped. Add the option `"-without-subdirs"` to the above command line.
- Build wxWidgets :
  ```
  make MONOLITHIC=1 SHARED=0 UNICODE=1 BUILD=release DEBUG_FLAG=0
  ```
- Install the wxWidgets libraries :
  ```
  make install
  ```

4. Build gSpiceUI using MinGW :
   - Retrieve the gSpiceUI sources and store them in eg. your MinGW home directory.
   - Open MSYS and unpack the gSpiceUI sources eg. :
     ```
     tar -zxvf gspiceui-v1.2.00.tar.gz
     ```
   - Enter to the wxWidgets sources root directory eg. :
     ```
     cd ~/gspiceui-v1.2.00
     ```
   - Build the gSpiceUI sources (the same as for Linux) :
     ```
     make GSPICEUI_MSWIN=1
     ```
   - If all goes well you can now run gSpiceUI under Windows :
     ```
     bin/gspiceui.exe
     ```

Note : There will be things that don't work in gSpiceUI under Windows eg. opening the online manual. The Windows setup seems to be a bit different from Linux but it's a start (for someone else to build on).

3.6 Apple Mac OSX Support

There is good and bad news for Mac users. The good news is that gSpiceUI can be installed and run under OSX (based on user reports). The bad news is that I don't have any instructions on how to do it, since I don't have access to an OSX system. I've added patches to the code base provided by Mac user who are running gSpiceUI on OSX. I've tried to run a OSX client under Qemu but without success so far. If anyone wants to provide some instructions I'm happy to insert them here.

4 Command-Line

Generally, envoking gSpiceUI simple requires entering the binary name (eg. gspiceui) at the command prompt. Various options and/or arguments may also be passed to gSpiceUI via the command-line. Configuration settings are recorded in a configuration file so preference entered in previous invocations are carried over to future sessions. Options entered via the command-line over-ride configuration file settings. What follows if the output generated by entering the "-h" option :

```
$ gspiceui -h

Analyse a electronic circuit using a GUI front-end to numerical simulation
engines

USAGE   : gspiceui [-OPTION [ARG]] [FILE/S]

OPTIONS : -h          : Print usage (this message) and quit
          -v          : Print version information and quit
          -c          : Rebuild/clean the configuration file and quit
          -d          : Enable debug mode (generate console spew on stderr)
          -r <RCFILE> : Specify a configuration file
                        <RCFILE> = ~/.gspiceui.conf (default)
          -s <SIMENG> : Specify the simulation engine to be used
                        <SIMENG> = ngspice or gnucap
          -e <EDA>    : Specify the EDA tool suite to be used
                        <EDA>    = lepton (Lepton-EDA) or geda (gEDA-gaf)
          -w <WAVWR>  : Specify the waveform data viewer to be used
                        <WAVWR>  = gaw, gwave or kst
          -g [<PROC>] : Import schematic file/s and (optionally) specify the
                        Guile procedure to use
                        <PROC>   = spice-sdb (default), protelii, verilog, etc.
          -a <PAGE>   : Specify the analysis page to be displayed
                       <PAGE>   = op (default), dc, ac, tr

FILE/S  : Load a circuit description (netlist) file or import schematic file/s
```

The available command-line options are explained in further detail below :

-h Print the usage message (as shown above) to the console and quit.

-v Print the application version information to the console and quit.

-c Clear disused configuration items from the configuration file and quit. As gSpiceUI is developed the format of the configuration file (ie. /.gspiceui.conf) may change. Over time it can become cluttered with superseded variable and/or group names. Although it is not essential, rebuilding the configuration file occasionally is a good idea, eg. whenever gSpiceUI is updated to a newer version.

-d Enable debug mode. As the application runs debug messages will be printed to the console. This is mainly used as development tool, it will generally not be of interest to the normal user.

-r Specify a alternative configuration file other than the default.

-s Specify the simulation engine to be used. This option can only work if the chosen simulation engine is installed on the system.

-e Specify the EDA tool suite to be used. This option can only work if the chosen EDA tool suite is installed on the system.

-w Specify the waveform data viewer application to be used. This option can only work if the chosen application is installed on the system.

-g Import schematic file/s and optionally specify the Guile procedure to be used. Importing schematic file/s means converting them to a single netlist file. The netlist conversion utility uses a Guile procedure to do the actual conversion. Different schematic file formats (eg. Protel or gEDA) require different processes to generate the netlist file. (If no Guile procedure is specified a default is used.)

-a Specify the analysis page to be displayed at startup.

There are two types of file which may be passed to gSpiceUI via the command-line (these file types cannot be mixed) :

Netlist : A single netlist file may be specified by placing it at the end of the command-line.

Schematic : One or more *related* schematic files may be specified at the end of the command-line. In addition the **-g** must be given to indicate that files are schematic. The file/s are converted to a single netlist file as part of gSpiceUI startup process.

Notes :

- The default configuration file is called ".gspiceui.conf" and is stored in the user's home directory. The configuration file is intended to be human readable although under normal circumstances it shouldn't be necessary to do so.
- Schematic file/s cannot be imported with the KiCAD EDA tool suite since it has no command-line netlister utility. A netlist file must be manually generate from within the KiCAD schematic capture application eeschema and then loaded / reloaded into gSpiceUI.

5 Graphical User Interface (GUI)

The main window is made up of various display objects which are described in detail in the following sub-sections. The GUI design attempts to facilitate the electronic circuit analysis work flow. There are two list boxes on the left containing possible test points, either electronic circuit nodes or single port components. To the right are analysis panels which each define a type of analysis which may be performed and allowing the user to enter parameters defining the analysis itself. The lower section of the GUI contains the text input and output of the various stages of the simulation. From the netlist created from the circuit schematic to the results generated by the simulation engine.

The main window title bar always contains at least the application name. If a file is open, whether it be a netlist or schematic file/s, the name of a netlist file will be appended to the normal window title banner. Keep in mind that this netlist file is the main file that gSpiceUI operates on.

5.1 File Menu

The file menu contains options associated with handling netlist and schematic files.

5.1.1 File → Open Netlist

This option opens a file dialogue allowing the user to choose a netlist (circuit description) file to open. The components and circuit nodes are loaded into the associated list controls and the netlist file contents are also loaded into the NetList text control in the Console notebook. If the netlist file contains simulation instructions gSpiceUI attempts to parse them and the settings are displayed in the appropriate page of the Analysis notebook.

Notes :

- Be aware, gSpiceUI is very likely to modify the netlist file. It will change file the formatting and remove things it doesn't understand. If this isn't OK, create a copy of the netlist file and open the copy with gSpiceUI.
- When using the gEDA tool set the netlist file extension is ".ckt". For the KiCAD tool suite the netlist extension is "cir". Be sure to specify the appropriate file extension for the EDA tool suite you are using.
- There is an upper limit to the number of lines which may be displayed in any text control (although the underlying file isn't truncated). This setting maybe adjusted by the user, refer to *5.3.4 Preferences*.

5.1.2 File → Import Schematic/s

This option opens a file dialogue allowing the user to choose one or more *related* schematic files to import. A utility provided by the EDA tool suite (eg. for gEDA-gaf it's gnetlist) is used to convert schematic file/s to a single netlist file. The netlist file name is automatically derived from the first schematic file name in the list with the file extension changed to ".ckt". Eg. if the schematic files *schem1.sch*, *schem2.sch*, *schem3.sch* were specified to be imported the netlist file name would be *schem1.ckt*.

Notes :

- If you are using the KiCAD tool suite the *Import* option won't work. The netlister utility associated with KiCAD is built into the schematic capture application *eeshema*

so gSpiceUI can't envoke it. The netlist file must be created or updated from within *eeshema* by the user and the netlist file must be updated in gSpiceUI using the *Reload* operation.

5.1.3 File → Reload

This option reloads the netlist file. If the netlist was imported from one or more schematic files, the import operation is repeated and then the resulting netlist file is reloaded. As far as possible the simulation settings shown in the currently displayed analysis notebook page are retained.

When gSpiceUI is used to create a netlist file from schematic file/s it inserts a reference to the schematic file/s near the beginning of the netlist file. If a netlist file is opened and has this schematic reference, the reload operation performs an import operation first before reloading the netlist file, otherwise the netlist is simply reloaded. In either case the simulation settings shown in the currently displayed analysis notebook page are retained as far as possible.

Notes :

- There is a preference which enables automatic re-generation of the netlist file from the schematic file/s if the modification timestamp of the netlist file is older than that of a schematic file/s. Refer to Section *5.3.4 Preferences*.

5.1.4 File → Close

Close any/all open schematic and/or netlist file and resets all GUI parameters to default values.

5.1.5 File → Quit

Close all files and child processes (eg. Gaw), delete temporary files if required and exit the application.

5.2 Simulate Menu

The simulate menu contains options associated with circuit simulation operations and provides access to schematic capture and waveform data viewing applications.

5.2.1 Simulate → Create Simulation

The *Create Simulation* operation tranforms the netlist file (displayed in the *Netlist* page of the Console notebook) into a simulation file (displayed in the *Simulation* page of the Console notebook).

The *Create Simulation* operation has some subtleties which are worth noting. The simplist way to use it is to enter your preferences into the GUI, *Create Simulation* then parses the netlist file, creates the appropriate simulation engine commands and then generates the simulation file. The simulation file can then be passed directly to the simulation engine to be executed.

Prior to running the simulation, there is a extra step possible that often proves useful. The simulation file, once created, can be edited by the user, eg. to temporarily change a component value. It is important to note however that the simulation file should only be

edited *after* the *Create Simulation* operation or the edits will be lost when the simulation file is recreated.

If a problem is encountered during the *Create Simulation* operation a message box is displayed giving a brief explanation as to why the operation could not be completed. The raw console spew generated by the process is sent to the Console page of the Console notebook; a careful perusal of this output generally helps to reveal the cause of the problem/s.

5.2.2 Simulate → Run Simulation

The *Run Simulation* operation creates a new simulation engine process and passes the simulation file (displayed in the *Simulation* page of the Console notebook) to the new process for execution.

The *Run Simulation* operation has some subtleties which are worth noting. If the analysis parameters entering in the GUI have changed since the simulation file was last created a *Create Simulation* operation is initiated automatically before creating the simulation process. Refer to the section *5.2.1 Create Simulation*.

If the simulation runs successfully then the results are formatted and written to a data file (refer to section *6 Temporary Files*) and are displayed in the appropriate simulation results page of the Console notebook.

If a problem is encountered during the *Run Simulation* operation a message box is displayed giving a brief explanation as to why the operation could not be completed. The raw console spew generated by the process is sent to the Console page of the Console notebook; a careful perusal of this output generally helps to reveal the cause of the problem/s.

5.2.3 Simulate → Stop Simulation

The *Stop Simulation* operation stops a running simulation immediately. This option only works if a simulation is actually running. This is a useful option if a simulation is running too long or appears not to be converging.

5.2.4 Simulate → Schematic Capture

The *Schematic Capture* operation starts a schematic capture application where the schematic can be viewed and/or edited. If there is a schematic file associated with an open netlist file it is opened using the EDA tool suite schematic capture application (eg. gschem or lepton-schem). It is very useful (almost essential) to view a schematic file whilst simulating a circuit. It provides a context for the various components and circuit nodes. Values in the circuit can be adjusted as the simulation process reveals new knowledge and understanding. Such changes are not, however, automatically reflected in gSpiceUI.

Remember that the schematic capture program and gSpiceUI are separate applications. Communication between the two happens via file/s on a storage device. When a schematic is changed, the associated file has to be updated and gSpiceUI told to re-read the file and updates it's GUI. This is a very simple but important process to keep in mind. Save your schematic to file and then reload it in gSpiceUI. Refer to section *5.1.3 File Reload*.

It's important to be aware that gSpiceUI cannot just run a simulation based on "any old" schematic file. In fact, the simulation will probably fail unless the schematic is trivial or you're just lucky. Certain prerequisites must be met before a schematic can be simulated. It must be possible to translated it into a form which can be interpreted by the simulation engine, by the EDA tool suite's netlist conversion utility (eg. gnetlist or lepton-netlist).

The extra information that must be explicitly defined in the schematic over that sufficient for human readability is :

- All power supply sources must be explicitly defined.
- All signal sources must be explicitly defined.
- Components not already defined in the simulation environment must be defined using models.
- Component labels must adhere to the rules specified for the simulation engine eg. resistor names always start with the letter 'R'. In particular be aware that often a model file contains a sub-circuit which must have a name starting with the character 'X'.
- Net labels must adhere to the rules specified for the simulation engine eg. they can't start with a digit. Refer to the specific simulation engine documentation.

Notes :

- In gschem and lepton-schem, models may be included by setting the component's file attribute or by use of a SPICE Model object. Either way the component value is set to the model or sub-circuit label. Check the example schematics that come with gSpiceUI to see how it's done eg. the LM555 or operational amplifier examples.
- gSpiceUI recognizes a component type by the first letter of the component name eg. an inductor name is expected to start with the letter 'L'. The following is list of the first letter of various component types (this format is derived from NG-SPICE) :

B - Non-Linear Dependent Source

C - Capacitor

D - Diode

E - Voltage Controlled Voltage Source

F - Current Controlled Current Source

G - Voltage Controlled Current Source

H - Current Controlled Voltage Source

I - Independent Current Source

J - JFET (Junction Field-Effect Transistor)

K - Coupled (Mutual) Inductors

L - Inductor

M - MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor)

O - Lossy Transmission Line (LTRA)

P - Coupled Multi-conductor Transmission Line (CPL)

Q - BJT (Bipolar Junction Transistor)

R - Resistor

S - Voltage Controlled Switch

T - Lossless Transmission Line

U - Uniform Distributed RC Transmission Line (URC)

V - Independent Voltage Source

W - Current Controlled Switch

X - Sub-circuit

Y - Single Lossy Transmission Line (TXL)

Z - MESFET (Metal-Semiconductor Field Effect Transistor)

- If the KiCAD EDA tool suite is used the *View Schematic* function will not work. The KiCAD schematic capture application *eeschema* must be manually started by the user and the netlist file updated by the user from within eeschema whenever the schematic is changed. Perform a Reload operation and then proceed as normal.

5.2.5 Simulate → View Results

The *View Results* operation starts a waveform data viewing application (ie. Gaw, Gwave or Kst) where the simulation results can be viewed and analysed. In the majority of cases, the first operation following a simulation run is to view the waveform data results. The run simulation tool bar button is pressed followed immediately by the view simulation results button. In these instances everything happens seemlessly, the results are generated and then viewed. However, gSpiceUI can generate a number of different data files any of which you may wish to view. Refer to section *6 Temporary Files*. You can load the desired results file manually into the waveform data viewer or configure gSpiceUI so that the desired data will be displayed when the waveform data viewer is started by gSpiceUI.

For this reason there is a mechanism for deciding which data file is opened by the waveform data viewer. The choice is made according to the following criteria in descending priority :

1. The currently selected page in the Analysis Notebook.

   Eg. if the *Quiescent* analysis page is selected in the Analysis Notebook then gSpiceUI will only attempt to display operating point data, if none has been generated an error message is displayed.

2. The currently selected results page in the Console Notebook.

   Eg. if the *AC* analysis page is selected in the Analysis Notebook and data for NGSPICE has been generated but the GNU-CAP result page is selected in the Console Notebook an error message will be displayed re no GNU-CAP result existing.

3. The currently selected Simulation Engine.

   Eg. if the *Transient* analysis page is selected in the Analysis Notebook and data has been generated by both simulation engines and the Netlist page in the Console Notebook is selected then the data for the currently select simulation engine will be displayed.

5.2.6 Simulate → Calculator

The *Calculator* operation starts a calculator application (eg. Xcalc). This calculator application is not part of gSpiceUI but is one of many freely available applications which can be downloaded and installed from the Web. A calculator is frequently a useful tool when simulating electronic circuits. The particular application to be launched can be configured in *5.3.4 Preferences*.

5.3 Settings Menu

The settings menu contains application configuration options. This menu begins with three blocks of radio buttons which allow easy access to options that are used frequently. Less often accessed options are contained in a preferences dialogue.

5.3.1 Settings → Simulation Engine Selection

The first pair of radio buttons allow the selection of the electronic circuit simulation engine to be used (assuming it has been installed). The analysis notebook is updated based on the chosen simulation engine, the selected simulation engine name is displayed in the second status bar pane and, when simulation results are generated, they are sent to the appropriate console tab.

When the simulation engine is changed gSpiceUI attempts to transfer simulation information to the new simulator environment. The information is tranferred via the simulation file so if information entered on the GUI is not in the simulation file it will be lost. If this is a problem create a fresh simulation file prior to changing the simulation engine.

5.3.2 Settings → EDA Tool Suite Selection

The second pair of radio buttons allow the selection of the EDA (Electronic Design Automation) tool suite to be used (assuming it has been installed). This specifies the schematic capture / edit application and the netlister utility to be used. The currently selected EDA tool suite name is displayed in second last pane of the status bar.

5.3.3 Settings → Waveform Data Viewer Selection

The third block of radio buttons allow the selection of the waveform data viewer application to be used (assuming it has been installed). The currently selected waveform data viewer application name is displayed in last pane of the status bar.

5.3.4 Settings → Preferences

The preferences dialog allows the user to configure gSpiceUI and contains the following settings :

- Simulation engine :

  A choice box containing the available simulation engines (assuming they have been installed). The currently selected simulation engine name is displayed in second pane of the status bar.

- EDA tool suite :

  A choice box containing the available EDA (Electronic Design Automation) tool suites (assuming they have been installed). The currently selected EDA tool suite name is displayed in second last pane of the status bar.

- Waveform data viewer :

  A choice box containing the available waveform data viewer applications (only applications which are currently installed will be listed). The currently selected waveform data viewer application name is displayed in last pane of the status bar.

- Calculator :

  A choice box containing the available calculator applications currently installed on your system.

- PDF viewer :

  A choice box containing the available PDF viewer applications currently installed on your system. The PDF viewer is used to display PDF documentation, eg. this gSpiceUI User Manual.

- Temporary files :

  A choice box containing the available temporary file management strategies, ie. automatically delete temporary files, prompt the user for permission or keep all temporary files. Most of the files in question contain simulation results data and may be useful outside gSpiceUI. Refer to section *6 Temporary Files* for more details.

- Main frame layout :

  A choice box containing the available main frame layout schemes. There are two schemes to choose from : longer probes (Nodes and Component list) controls or a wider console notebook control.

- Phase / angle units :

  A choice box containing the available phase / angle units, ie. degrees or radians.

- Results precision :

  A choice box allowing the precision of the simulation results to be specified. The data are rounded to the chosen precision, not truncated.

- Netlister Guile backend :

  A choice box containing the available netlist conversion utility Guile backend procedures. This list of Guile backends is derived by querying the netlister utility directly so whatever procedures are listed are available.

- Max. text control size :

  A value control which sets the maximum number of lines which may be displayed in any of the Console Notebook pages (text controls). Simulation results files can, potentially, be very big; this value places an upper limit on how many lines are loaded into any text control. The files themselves are not changed in any way, ie. truncated.

- Spin control period :

  A value control which sets the period (in msec) between successive spin button updates.

- Tool tip delay :

  A value control which sets the delay (in msec) before a tool tip is displayed.

- Show tool tips :

  A check box which enables or disables tool tips.

- Auto. config. file clean :

  A check box which enables or disables automatic configuration file clean / rebuild. As gSpiceUI is developed the configuration file format may change (ie. /.gspiceui.conf). Over time it can become cluttered with superseded variable and/or group names. Although it is not essential, rebuilding the configuration file occasionally is a good idea, if nothing else it helps with human readability. If this option is enabled and the application is updated to a newer version, on first startup the configuration file is automatically rebuilt.

- Sync. sweep sources :

  A check box which enables or disables synchronization of the sweep sources. If a sweep source name is selected in one analysis page of the Analysis Notebook, where possible the same source name will appear in other analysis pages. Note : the sweep source *value* is not carried over between pages.

- Sync. temperatures :

  A check box which enables or disables synchronization of the ambient temperature values between analysis pages of the Analysis Notebook and the OPTIONS dialogue.

- Keep the netlist file :

  A check box which controls whether netlist files are included in the list of temporary files and therefore the temporary file management strategy.

- Netlister verbose mode :

  A check box which enables or disables the netlist conversion utility (eg. gnetlist) verbose mode. With this enabled the netlister generates lots of console spew which appears in the Console tab of the Console Notebook. It's useful for debugging when the netlister is having problems importing schematic file/s.

- Include model defs. :

A check box which controls whether .INCLUDE directives are used for model files or the model file contents are inserted in the netlist file by the netlister (eg. gnetlist). (This option controls the the SPICE Guile backend option "include_mode".)

- Embed include files :

  A check box which controls how the netlister (eg. gnetlist) behaves when it encounters a .INCLUDE directive. With this option checked the netlister embeds (inserts) the file model contents into the netlist file. (This option controls the netlister SPICE backend option "embedd_mode".)

- Fix component prefixes :

  A check box which controls whether component label prefixes are automatically tested and modified if the incorrect prefix is used. SPICE simulation engines recognizes different components based on the first character of the component label (eg. R1 would be identified as a resistor). When this option is checked, if an incorrect prefix is detected the correct prefix is prepended to the component label. (This option controls the netlister (eg. gnetlist) SPICE backend option "nomunge_mode".)

- Auto. regenerate netlist :

  A check box to control whether the netlist is automatically regenerationed from the associated schematic file/s. If this option is checked, before each simulation run the schematic file modification timestamp/s are compared to that of the netlist file. If a schematic file is newer than the netlist a reload operation is automatically executed (ie. the netlist file is regenerated from the schematic file/s).

5.4 Help Menu

The Help Menu contains the options to display various user manuals (assuming they are installed properly) as well as gSpiceUI "About" information. These options are described in the following sections.

5.4.1 Help → gSpiceUI Manual

Open the gSpiceUI HTML documentation (ie. this document) in a HTML viewer window. The HTML viewer is part of the wxWidgets library and only provides basic browser functionality however it is quite adequate for the purpose. This document can, of course, also be opened in any web browser outside of gSpiceUI. This options is included for convience.

5.4.2 Help → NG-SPICE Manual

Open a PDF viewer application to read the NG-SPICE user manual. The NG-SPICE user manual must be installed on the system. Refer to the gSpiceUI install instructions for the location gSpiceUI expects to find the PDF file.

5.4.3 Help → GNU-CAP Manual

Open a PDF viewer application to read the GNU-CAP user manual. The GNU-CAP user manual must be installed on the system. Refer to the gSpiceUI install instructions for the location gSpiceUI expects to find the PDF file.

5.4.4 Help → About

Open a dialogue containing application "About" information, including gSpiceUI version details, the version of wxWidgets library gSpiceUI is built against, supported external applications and warranty / copyright information.

5.5 Tool Bar

The tool bar contains buttons which give easy access to the most used functions. The buttons are presented in groups associated with the different types of activities. The following table shows the button icon with a brief explanation of it purpose and a link to the section dealing with it's function in detail :

 → Open a netlist file (refer to section *5.1.1 File Open*)

 → Import schematic file/s (refer to section *5.1.2 File Import*)

 → Reload netlist/schematic file/s (refer to section *5.1.3 File Reload*)

 → Close netlist/schematic file/s (refer to section *5.1.4 File Close*)

 → Create simulation file (refer to section *5.2.1 Create Simulation*)

 → Run simulation (refer to section *5.2.2 Run Simulation*)

 → Stop simulation (refer to section *5.2.3 Stop Simulation*)

 → Schematic capture (refer to section *5.2.4 Schematic Capture*)

 → View simulation results (refer to section *5.2.5 View Results*)

 → Calculator (refer to section *5.2.6 Calculator*)

 → Preferences (refer to section *5.3.4 Preferences*)

 → View gSpiceUI user manual (refer to section *5.4.1 gSpiceUI Manual*)

5.6 Node List

The *Node List* contains node labels extracted from the netlist file. These labels can be used as probes into the circuit being simulated. A probe is a point where electrical parameters (eg. voltage) are calculated by the simulation engine. One or more nodes may be selected (or deselected) by clicking on them with the mouse. The probes for the selected nodes will be calculated and included in the simulation results.

A single node may be selected from the list by left clicking on it with the mouse. To select multiple nodes, press the control key and left click on the desired nodes with the mouse. To select a range of nodes left click on the desired range with the mouse while holding down the shift key.

5.7 Component List

The *Component List* contains component labels extracted from the netlist file. These labels can be used as probes into the circuit being simulated. A probe is a point where electrical

parameters (eg. voltage) are calculated by the simulation engine. One or more components may be selected (or deselected) by clicking on them with the mouse. The probes for the selected components will be calculated and included in the simulation results. At present only two terminal components are included in this list.

A single component may be selected from the list by left clicking on it with the mouse. To select multiple components press the control key and left click on the desired components with the mouse. To select a range of components left click on the desired range with the mouse while holding down the shift key.

5.8 Analysis Notebook

The *Analysis Notebook* contains analyis pages, each page reflects the a different analysis type supported by the associated simulation engine. There are two slightly different analysis notebooks associated with the two different simulation engines supported by gSpiceUI ie. GNU-CAP or NG-SPICE. The following sections *briefly* outline the analysis types supported by gSpiceUI. For more information refer to the user documentation for the appropriate simulation engine (much of what follows is taken from this documentation). Refer to section *5.4 Help Menu*.

5.8.1 Operating Point Analysis

NG-SPICE : This analysis type is supported using the DC command in NG-SPICE. The DC analysis determines the DC operating point of the circuit with inductors shorted and capacitors opened. In this instance the DC analysis is used to generate DC transfer curves for temperature stepped over a user-specified range and the DC output variables are stored for each sequential temperature value.

GNU-CAP : Performs a nonlinear DC steady state analysis. If a temperature range is given, it sweeps the temperature. If there are numeric arguments, they represent a temperature sweep. They are the start and stop temperatures in degrees Celsius, and the step size, in order. This command also sets up the quiescent point for subsequent AC analysis. It is necessary to do this for nonlinear circuits. The last step in the sweep determines the quiescent point for the AC analysis.

5.8.2 DC Analysis

NG-SPICE : The DC analysis determines the DC operating point of the circuit with inductors shorted and capacitors opened. A DC analysis is automatically performed prior to a Transient analysis to determine the transient initial conditions, and prior to an AC small-signal analysis to determine the linearized, small-signal models for nonlinear devices. The DC analysis can also be used to generate DC transfer curves : a specified independent voltage, current source, resistor or temperature is stepped over a user-specified range and the DC output variables are stored for each sequential source value.

GNU-CAP : Performs a nonlinear DC steady state analysis, and sweeps the signal input, or a component value. If there are numeric arguments, without a part label, they represent a ramp from the generator function. They are the start value, stop value and step size, in order. In some cases, you will get one more step outside the specified range of inputs due to internal rounding errors. The last input may be beyond the end

point. The program will sweep any simple component, including resistors, capacitors, and controlled sources.

5.8.3 AC Analysis

NG-SPICE : The AC small-signal analysis determines the output variables as a function of frequency. The DC operating point of the circuit is first determined and used to generate linearized, small-signal models for all of the nonlinear devices in the circuit. The resultant linearized version of the circuit is then analyzed over a user-specified range of frequencies. The desired output of an AC small-signal analysis is usually a transfer function (trans-impedance, voltage gain, etc.). If the circuit has only one AC input, it is convenient to set that input to unity and zero phase, so that output variables have the same value as the transfer function of the output variable with respect to the input.

GNU-CAP : Performs a small signal, steady state, AC analysis. Sweeps frequency. The AC command does a linear analysis about an operating point. It is absolutely necessary to do an OP analysis first on any nonlinear circuit. Not doing this is the equivalent of testing it with the power off. If the start frequency is zero, the program will still do an AC analysis. The actual frequency can be considered to be the limit as the frequency approaches zero. It is, therefore, still possible to have a non-zero phase angle, but delays are not shown because they may be infinite.

5.8.4 Transient Analysis

NG-SPICE : The Transient analysis determines the transient output variables as a function of time over a user-specified time interval. The initial conditions are automatically determined by a DC analysis. All sources which are not time dependent (for example, power supplies) are set to their DC value.

GNU-CAP : Performs a nonlinear time domain (transient) analysis. Do not use a step size too large as this will result in errors in the results. If you suspect that the results are not accurate, try a larger argument to skip. This will force a smaller internal step size. If the results are close to the same, they can be trusted. If not, try a still larger skip argument until they appear to match close enough. The most obvious error of this type is aliasing. You must select sample frequency at least twice the highest signal frequency that exists anywhere in the circuit. This frequency can be very high, when you use the default step function as input. The signal generator does not have any filtering.

5.9 Console Notebook

As a normal part of gSpiceUI operations, alot of text output is generated (eg. by command-line utilities like gnetlist) and collected by gSpiceUI. This text may be in the form of file contents or output sent to standard out. (Like many UNIX applications gSpiceUI mostly communicates with the outside world via files.) It collects the console output of the processes it runs for debugging purposes but mostly gSpiceUI depends on files. These files are human readable and can be of use to the user. The Console Notebook is the repository for this text output. It is divided into various pages (described below) according to category of the text output.

5.9.1 Console

Display the console input/output for the last process executed by gSpiceUI. This includes the command-line used by gSpiceUI followed by all text output sent to stdout and stderr. If an error occurs this is a good place to look for indications of what has gone wrong.

5.9.2 Netlist

Display the raw contents of a netlist file. The variables and values contained in the netlist file are displayed in the GUI objects. In particular the Node and Component Lists are derived from the netlist file.

5.9.3 Simulation

Display the contents of the simulation file. This is the file sented to the simulation engine when a run operation is intiated. It is the netlist file (as displayed in the Netlist tab of the Console Notebook) plus simulator commands generated by gSpiceUI based on user input. The simulation file can be manually edited by the user before executing a simulation. NOTE : if *Create Simulation* operation is initiated, the contents of this tab will be over-written, destroying any user input.

5.9.4 NG-SPICE Results

Display the results from the last run of the NG-SPICE simulation engine. The raw results data is reformatted to aid readability; non-data records are removed and the precision is set according to the value set in the Preferences dialog. If required the raw output from the simulation engine may also be viewed in the Console tab.

5.9.5 GNU-CAP Results

Display the results from the last run of the GNU-CAP simulation engine. The raw results data is reformatted to aid readability; non-data records are removed and the precision is set according to the value set in the Preferences dialog. If required the raw output from the simulation engine may also be viewed in the Console tab.

5.10 Status Bar

The status bar is located at the bottom of the gSpiceUI main frame. It consists of four panes which contain text messages showing the current state of gSpiceUI. From left to right the panes contain :

1. The last application status or error message (abbreviated to fit if necessary).
2. The currently selected simulation engine.
3. The currently selected EDA tool suite.
4. The currently selected waveform viewer application.

6 Temporary Files

As part of normal operations gSpiceUI can generate various temporary files in the directory occupied by the schematic and netlist files. To illustrate, consider the situation where the schematic file "test-circuit.sch" is imported and every possible analysis type is run, using both NG-SPICE and GNU-CAP. Along with the schematic file (also listed below) the following temporary files which would be created :

<path>/sch/test-circuit.sch	(The schematic file to be analysed)
<path>/sch/test-circuit.ckt	(Netlist file with simulation commands)
<path>/sch/test-circuit.sch~	(Backup file to the schematic file)
<path>/sch/gspiceui.log	(Temporary file for raw process output)
<path>/sch/test-circuit.ngspice.op	(Results file - operating point analysis)
<path>/sch/test-circuit.ngspice.dc	(Results file - DC analysis)
<path>/sch/test-circuit.ngspice.ac	(Results file - AC analysis)
<path>/sch/test-circuit.ngspice.tr	(Results file - transient response analysis)
<path>/sch/test-circuit.gnucap.op	(Results file - operating point analysis)
<path>/sch/test-circuit.gnucap.dc	(Results file - DC analysis)
<path>/sch/test-circuit.gnucap.ac	(Results file - AC analysis)
<path>/sch/test-circuit.gnucap.tr	(Results file - transient response analysis)

All file names begin with the schematic file name less the file extension, in this example "test-circuit". File name extensions are then generated depending on the file type. In particular, files containing simulation results have two extension, the first based on the the simulation engine used to generate the data, the second is an abbreviation indicating the analysis type.

These temporary files can quickly proliferate and clutter your file system. As a consequence gSpiceUI offers several ways to manager these temporary files. The simplest methods is to allow gSpiceUI to delete them when they no longer required. If a schematic file is closed or gSpiceUI is exited then any temporary files are deleted. The second option is for gSpiceUI to prompt the user prior to deleting temporary files asking for permission. The last option is to just do nothing and not delete anything. There are occasions where you may wish to retain results data to be view later perhaps.

Notes :

- For details on how to set the temporary file for management scheme refer to section *5.3.4 Preferences*.
- If temporary files are deleted this also includes the schematic backup file.

7 Demonstration Schematics

This section is intended to serve the dual purpose of providing user documentation for the demonstration schematics that come with gSpiceUI, and to help me (the developer) keep track of them (ie. try to make ensure that they work). The directory <install-dir>/sch/demos/ contains the demostration schematic files. They are intended to illustrate the simulation of specific circuit elements eg. a diode.

In addition, the directory <install-dir>/sch also contains various schematic files which may be used to experiment with gSpiceUI. These examples show how to prepare a schematic for simulation, which is not always a trivial task. These files are a collection of things that have taken my interest over the years, some work and some don't. It's worth noting that although these schematics are definitely circuit designs, they may not be good designs, so use them in the real world at your own risk.

The directory <install-dir>/lib/symbols/ contains various circuit element symbol that I've developed over the years. They may also be of use.

Below is a list of the available demonstration schematics and a brief description of what each is intended to accomplish :

amp-bjt-ce-1.sch : BJT Common Emitter Amplifier

Demo. circuit showing how to simulate a NPN BJT in it's linear mode using AC analysis.

NG-SPICE (v32.1) ➛ (2020-08-22) OK

GNU-CAP (v20171003) ➛ (2020-08-22) Fault

amp-bjt-ce-2.sch : BJT Common Emitter Amplifier

Demo. circuit showing how to simulate a NPN BJT in it's linear mode using AC analysis (an alternative implementation of the previous example).

NG-SPICE (v32.1) ➛ (2020-08-22) OK

GNU-CAP (v20171003) ➛ (2020-08-22) Fault

amp-bjt-diff.sch : Audio Amplifier Differential Input Stage

Demo. circuit showing a common audio amplifier input stage topology using NPN BJTs.

NG-SPICE (v32.1) ➛ (2020-08-23) OK

GNU-CAP (v20171003) ➛ (2020-08-23) Fault

amp-jfet-cs-1.sch : N-Channel JFET Common Source Amplifier

Demo. circuit showing how to simulate a N-Channel JFET using either DC or AC analysis.

NG-SPICE (v32.1) ➛ (2020-08-23) OK

GNU-CAP (v20171003) ➛ (2020-08-23) Fault

amp-mosfet-cs-1.sch : N-Channel MOSFET Common Source Amplifier

Demo. circuit showing how to simulate a N-Channel MOSFET using either DC or AC analysis.

NG-SPICE (v32.1) ➛ (2020-09-06) OK

GNU-CAP (v20171003) ➛ (2020-09-06) Fault

astable-bjt-npn.sch : NPN BJT Astable Multivibrator

Demo. circuit showing how to simulate a NPN BJT when operated as a switch using Transient analysis.

NG-SPICE (v32.1) ➛ (2020-08-22) OK

GNU-CAP (v20171003) ➛ (2020-08-22) OK

astable-bjt-pnp.sch : PNP BJT Astable Multivibrator

Demo. circuit showing how to simulate a PNP BJT when operated as a switch using Transient analysis.

NG-SPICE (v32.1) ➛ (2020-08-22) OK

GNU-CAP (v20171003) ➛ (2020-08-22) Fault

diode-led-1.sch : Light Emitting Diode

Demo. circuit showing how to simulate a LED using DC analysis.

NG-SPICE (v32.1) ➛ (2020-09-08) OK

GNU-CAP (v20200806dev) ➛ (2020-09-08) OK

diode-signal-1.sch : Signal Diode

Demo. circuit showing how to simulate a signal diode using DC analysis.

NG-SPICE (v32.1) ➛ (2020-09-08) OK

GNU-CAP (v20200806dev) ➛ (2020-09-08) OK

diode-zener-1.sch : Zener Diode

Demo. circuit showing how to simulate a zener diode using DC analysis.

NG-SPICE (v32.1) ➛ (2020-09-08) OK

GNU-CAP (v20200806dev) ➛ (2020-09-18) Fault

filter-bp-1.sch : Series Resonant Band Pass Filter

Demo. circuit showing how to simulate a simple series resonant band pass filter using AC analysis.

NG-SPICE (v32.1) ➛ (2020-09-20) OK

GNU-CAP (v20200806dev) ➛ (2020-09-20) OK

filter-bp-2.sch : T-Section Resonant Band Pass Filter

Demo. circuit showing how to simulate a T-section resonant band pass filter using AC analsis.

NG-SPICE (v32.1) ➛ (2020-09-20) OK

GNU-CAP (v20200806dev) ➛ (2020-09-20) OK

filter-lp-1.sch : Single Pole Low Pass Filter

Demo. circuit showing how to simulate a simple single pole low pass filter using AC

analsis.

NG-SPICE (v32.1) ➛ (2020-09-20) OK
GNU-CAP (v20200806dev) ➛ (2020-09-20) OK

lm555-timer.sch : LM555 Timer

Demo. circuit showing how to simulate using a model based on a sub-circuit.

NG-SPICE (v32.1) ➛ (2020-09-23) OK
GNU-CAP (v20200806dev) ➛ (2020-09-23) OK

lvl-shftr-bjt.sch : ???

lvl-shftr-opamp.sch : ???

subckt-1.sch : MOSFET Switch Circuit

Demo. circuit showing how to simulate using a model based on a sub-circuit.

NG-SPICE (v32.1) ➛ (2020-09-23) OK
GNU-CAP (v20200806dev) ➛ (2020-09-23) Fault

tline-1.sch : Loss-Less Transmission Line

Demo. circuit showing how to simulate a loss-less transmission line.

NG-SPICE (v32.1) ➛ (2020-09-20) Fault
GNU-CAP (v20200806dev) ➛ (2020-09-20) OK

8 Design

This section provides information related to the design of gSpiceUI. In reality, this application was designed in my head as it went along. A fair amount of it has been re-written as I found better ways of doing things. The design methodology would be best described as prototyping.

The following object models where developed to help provide a better picture of how things went together (as the project got to the point where I couldn't remember how it all worked). The object models also depict how wxWidgets binds into gSpiceUI.

Figure 3 : gSpiceUI Application Object Model depicts the top level application architecture. The class App_gSpiceUI inherits from the wxWidgets library wxApp class which is the entry point to the application (ie. it contains main( )).

Figure 4 : Main Frame Class Object Model depicts the top level of the GUI architecture. The FrmMain class inherits from the wxWidgets library wxFrame class which contains all display objects within this class hierarchy.

Figure 5 : Analysis Class Object Model depicts the class structure of the objects which fill the analysis notebook (the notebook with the analysis tabs). It inherits from the wxWidgets library wxNotebook class.

Figure 6 : Process Class Object Model depicts the class structure of the objects which maybe executed as tasks by the application. It inherits from the wxWidgets library wxProcess class.

Figure 7 : PnlValue Class Object Model depicts the class structure of the display control used to contain numeric values. It combines wxWidgets library text, spin and choice controls so that numeric values (either integer or floating point) can be set using the mouse or keyboard.

Notes :

- The GNU SPICE GUI project is comprised of many C++ classes, more than are depicted in the object models shown here. However, all the major classes have been depicted giving a good idea of the overall application architecture.
- The object models depicted here largely show the actual application architecture. However, the software is under development and is often in a state of flux, it will not always *exactly* match the information shown in the object models.

The source code is heavily commented and has been designed and written so that it may be maintained (or reused) by myself or anyone else for that matter. The project has well and truely got to the stage where it can no longer be maintained simply from memory.

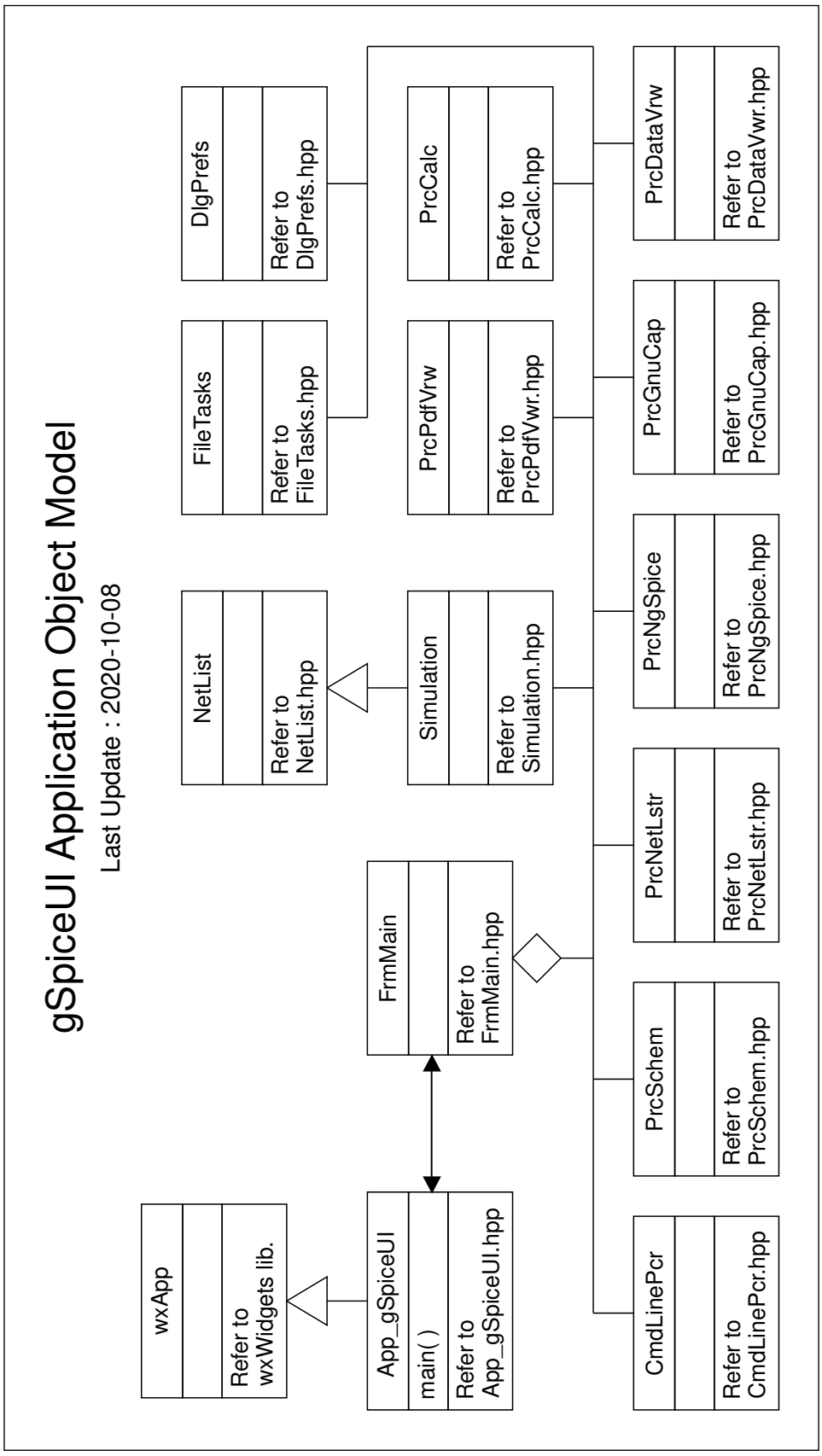

Figure 3: Overall gSpiceUI application object model.

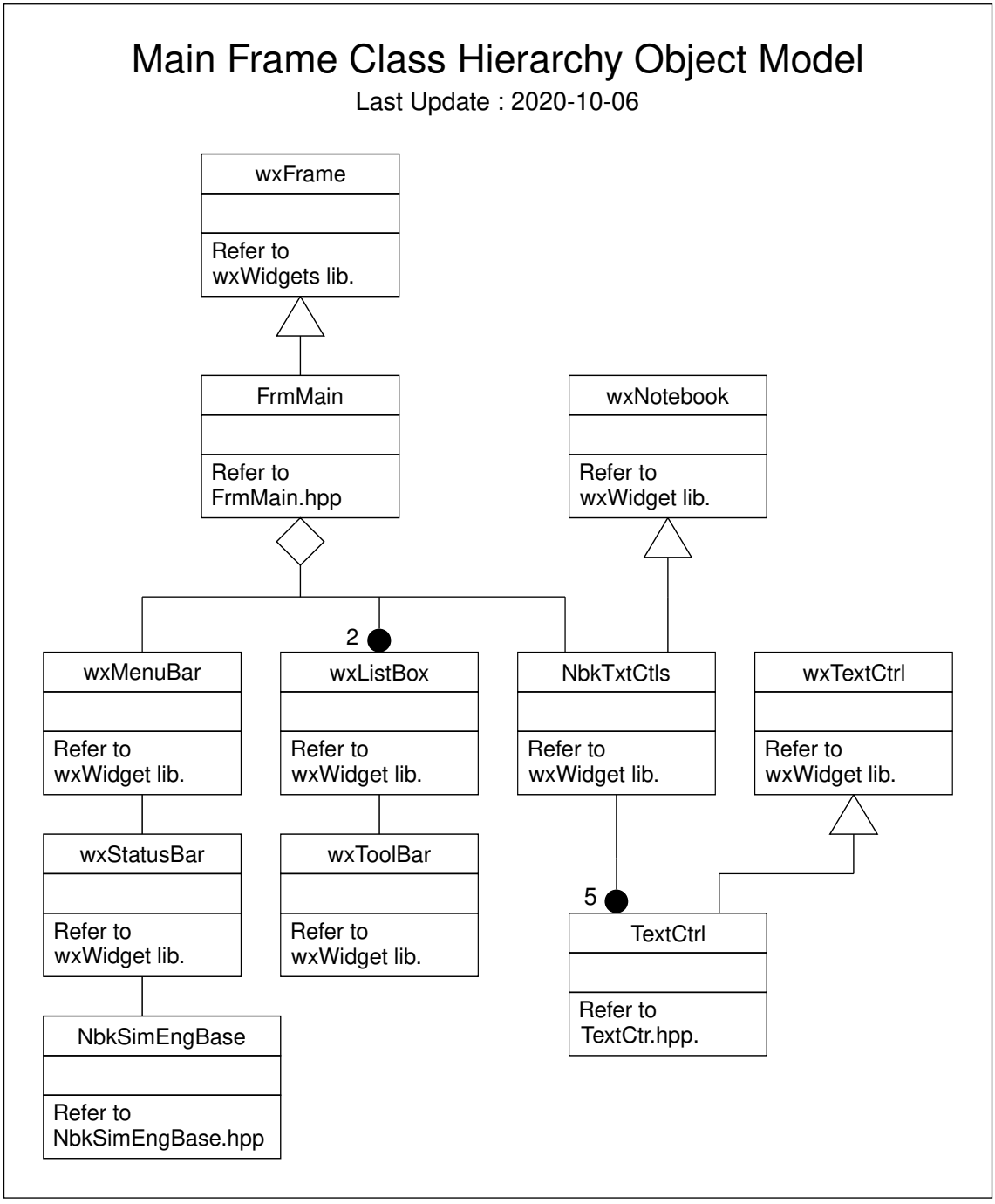

Figure 4: Main application frame object model.

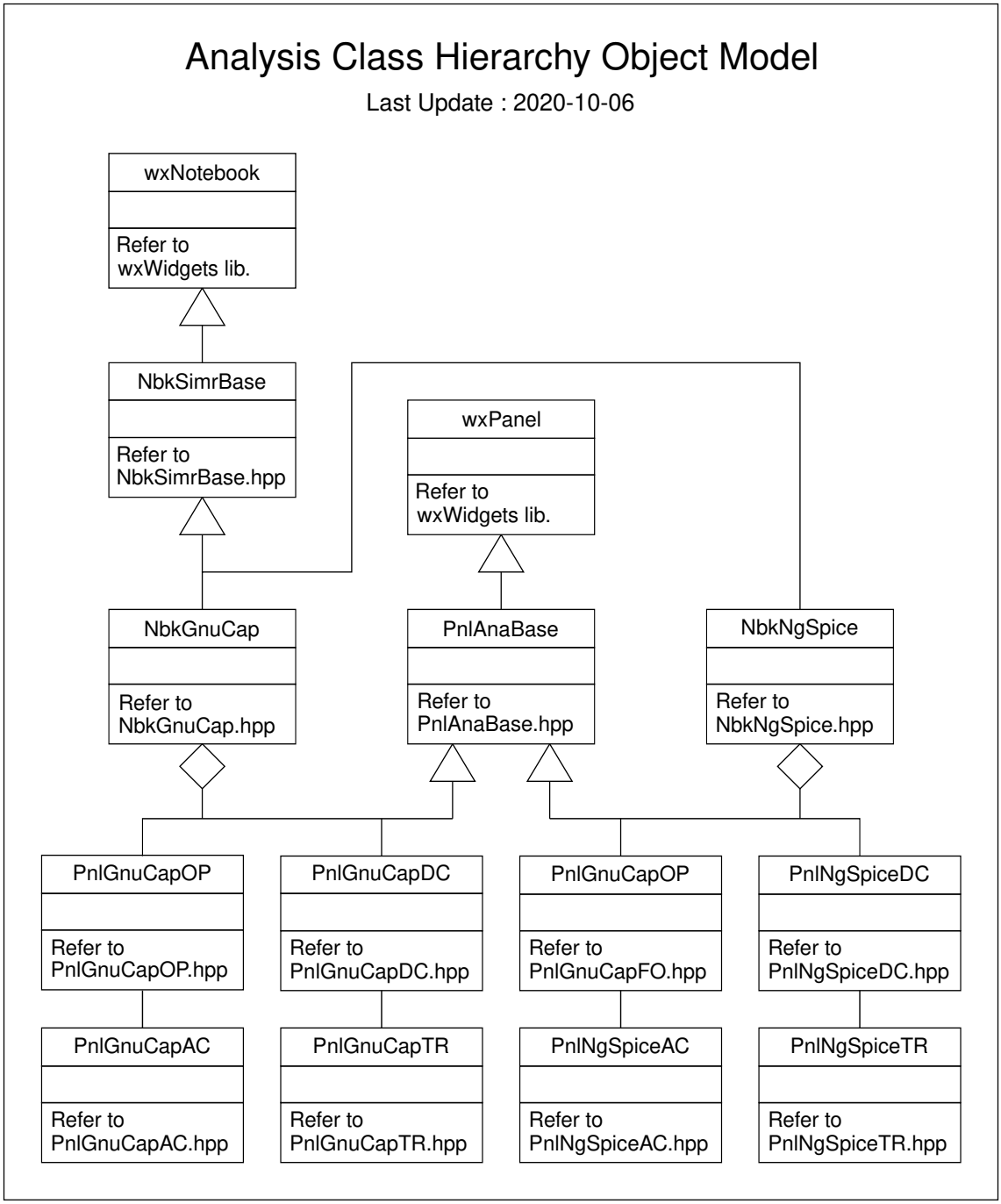

Figure 5: Analysis notebook object model.

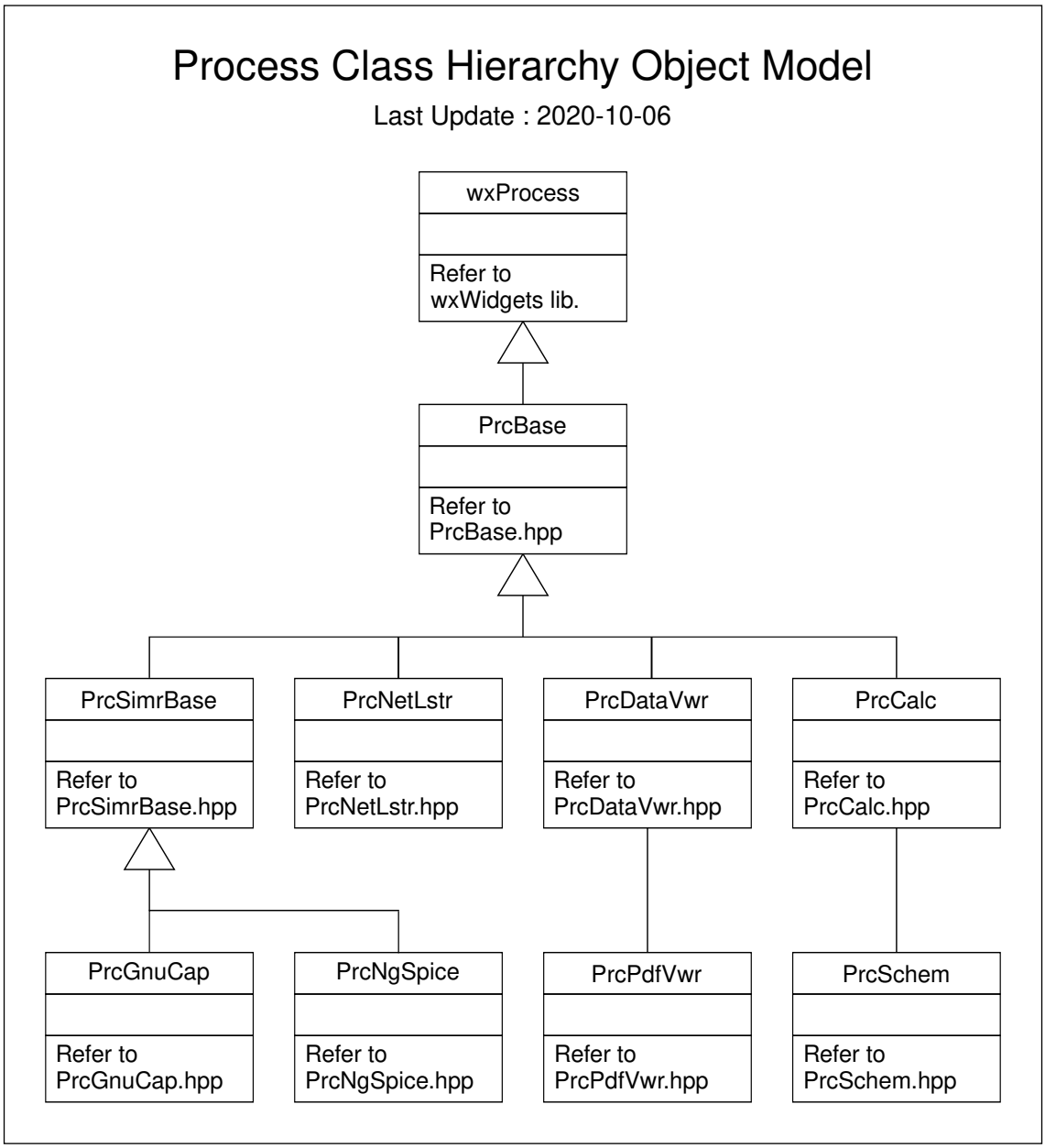

Figure 6: Process object model (tasks gSpiceUI may envoke).

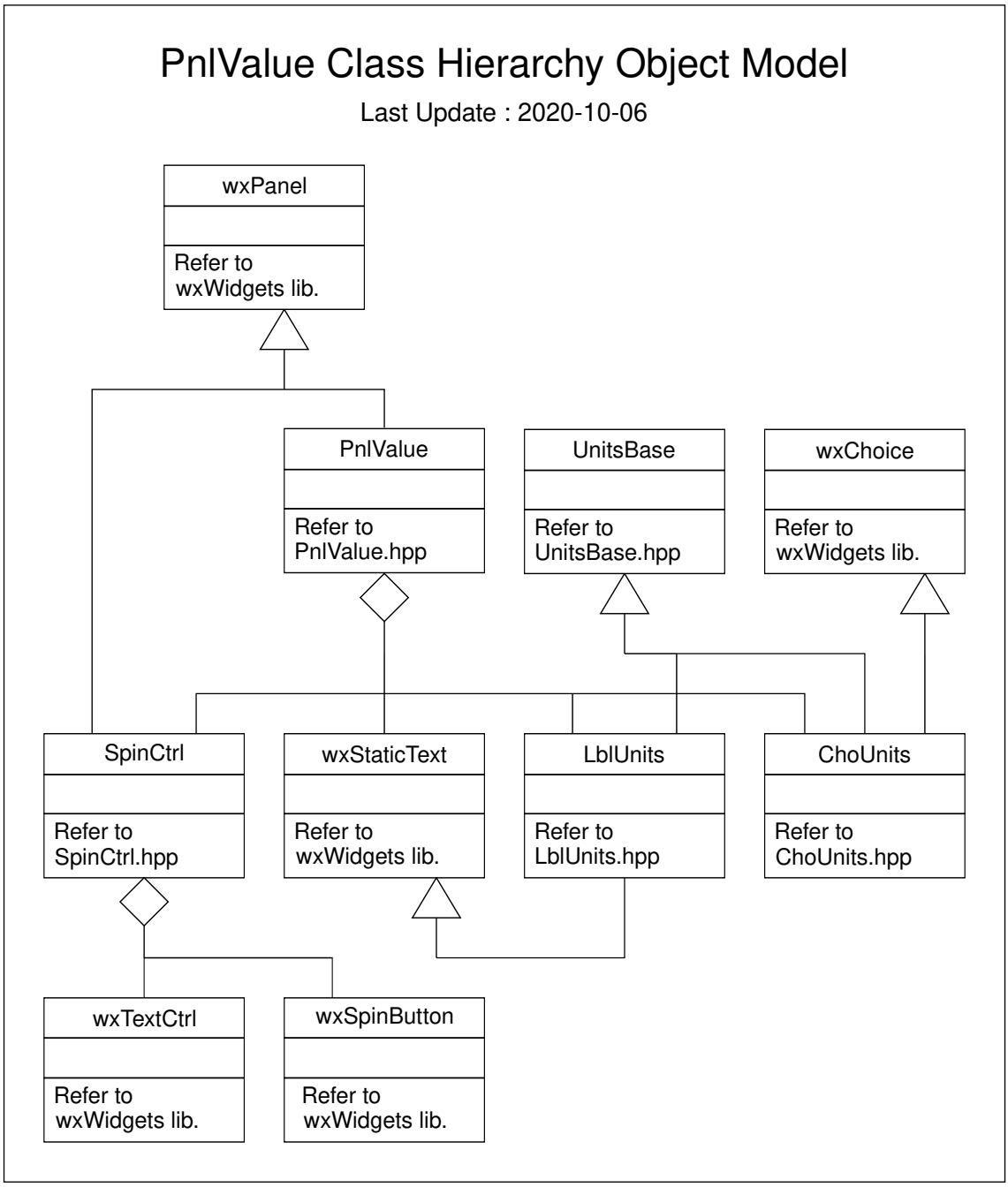

Figure 7: Value panel object model (displays data variables).