Implementing schematics in Neapolis 4

Frédéric Hancart

Kαβαλα, June 1999

Supervisor (Greece) : George Kyranastasis
Supervisor (Belgium) : Luc Van Huffel

Home institution

KaHo St.-Lieven
Departement Gent Campus Rabot
Gebr. Desmetstraat 1
9000 Gent, Belgium

Host institution

Technological Educational Institution
(TEI) of Kavala, Macedonia, Greece
Department of Electrical Engineering
Laboratory of Power Electronics
Implementing schematics in Neapolis 4

Frédéric Hancart

Καβαλα, June 1999

Supervisor (Greece) : George Kyranastasis
Supervisor (Belgium) : Luc Van Huffel

Home institution

KaHo St.-Lieven Departement Gent Campus Rabot
Gebr. Desmetstraat 1
9000 Gent, Belgium

Host institution

Technological Educational Institution (TEI) of Kavala, Macedonia, Greece
Department of Electrical Engineering
Laboratory of Power Electronics
# Table of contents

**FOREWORD AND THANK YOU'S** ................................................................. 3

**INTRODUCTION** ......................................................................................... 4

- Neapolis, the name and history ................................................................. 4
- This report ................................................................................................. 4
- The subject ............................................................................................... 4

**NEAPOLIS** ............................................................................................... 5

- What is it? .................................................................................................. 5
- Overview .................................................................................................. 5
- How to use Neapolis .................................................................................. 6
- File structure ........................................................................................... 7
- Directory structure .................................................................................... 9

**STARTING TO WORK** .............................................................................. 10

- Visual Basic ............................................................................................ 10
- The parts I needed .................................................................................... 10

**VISUAL BASIC** ....................................................................................... 11

- Why ......................................................................................................... 11
- Creating forms on the fly .......................................................................... 11
- Creating textboxes ................................................................................... 12
- Already loaded? ....................................................................................... 13

**SCHEMATICS** ......................................................................................... 14

- Overview ................................................................................................ 14
- The main window ..................................................................................... 15
- Changes in Neapartı ................................................................................ 17
- New schematics program ......................................................................... 19

**IMPLEMENTING NEW SCHEMATICS** .................................................. 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGING THE DATA FILES</td>
<td>27</td>
</tr>
<tr>
<td>THE NEW FORM</td>
<td>28</td>
</tr>
<tr>
<td>THE SOURCE CODE</td>
<td>30</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>32</td>
</tr>
<tr>
<td>THE NEAPOLIS PROGRAM</td>
<td>32</td>
</tr>
<tr>
<td>Socrates</td>
<td>32</td>
</tr>
<tr>
<td>TABLE OF FIGURES</td>
<td>33</td>
</tr>
</tbody>
</table>
Foreword and Thank You’s

Since last year I knew that as a student of KaHo Sint-Lieven it was possible to do a final project work abroad.
As I never did such a thing, I applied for a final project in Kavala, Greece.
For me it was the ideal opportunity to learn a lot about another country with a different culture and different people. Greece is really a good example.

That’s why I want to thank first Mr. G. Kyranastasis for allowing me as a TEI-student for three and a half months.
And also Mr. D. Hanselaer and Mr. L. Vanhuffel of the KaHo Sint-Lieven, because they organised the whole thing.
And last but not least, my parents because without them none of this would be possible.

To all of them: Thank you!

Frédéric Hancart
Kavala - June, 1999
Introduction

Neapolis, the name and history

Neapolis is an educational program (by Prof. Dr. G. Kyranastasis) used for simulating power electronic circuits, like electrical motors, rectifiers, etc…
Neapolis was the ancient name of Καβαλα, where the TEI (Technological Educational Institution) is located, and the program is developed.

This report

In this report you won’t find any Visual Basic source code of the program I made. There are enough comments written through the code to fully understand what it’s doing.
Neither, I explain how Visual Basic works (data structures, loops, functions, etc…), everything is explained in the Books Online or can be found on the Internet.
Also there are no test-programs I made to get used to Visual Basic included or explained in this document.

The subject

Mr Kyranastasis wanted to add the schematics feature of Neapolis 2.3 into Neapolis 4 in such a way that the user can follow a selected device’s voltage or current while the program is simulating the circuit.
The basic idea is to make it easier for the user to understand what curve he’s looking at. He has just to click on that part of the circuit he wants to see the curve of.
Neapolis

What is it?

Neapolis is an educational program born from the mind of Prof. Dr. G. Kyranastasis in 1989. At that time Neapolis was being developed in a DOS operating system. Nowadays, Neapolis is a 32 bit MS Windows™ program written in Visual Basic 5.

Neapolis is developed in the lab of Power Electronics for the students of this department. Neapolis is a simulation package for power electronics circuits, like motors, converters and motor drives.

Simulating a circuit has many advantages. Students are using a computer instead of the real thing, which usually operates at a high (dangerous) voltage.

It also possible to change different parameters quickly, analysing the behaviour of a circuit step by step, by means of numbers and graphs and it’s possible to process the results obtained by the simulation part of Neapolis.

Overview

Basically, Neapolis consists of a ‘shell-program’ and menu system, a Simulation part and a Result Processing part.

In Neapart1, the user can choose a system from the menus and configure it in the Define System part. Once configured, you can start the Simulation, and later also the Result Processing part of Neapolis.

The figure below illustrates this.
How to use Neapolis

The first thing you will have to do if you start Neapolis is chose a system you want to simulate from the menus, for example 1-phase rectifiers. You will see your selection will appear on the main form.

Figure 2 Define System : 1-phase rectifier

Afterwards the Define System button (and menu option) gets enabled. To change the different parameters, like the kind of load, the kind of supply and the typical parameters of the chosen circuit, select this option.
After configuring your chosen system, you can just exit (the Define System will write the necessary files) or you can save your settings into a file (so you're able to load your settings again, later). You can also start the simulation directly from the Define System screen.

Once back in the main window, you can simulate it by clicking on the Simulate button (or selecting the simulation in the Action menu). This will load the Simulation program. This one plots the most important waveforms of the circuit on the screen.

Figure 3 Simulating the 1-phase rectifier

After the simulation, you're able to process the output of this with the result processing part of Neapolis. It analyses the data from the simulation part to calculate harmonics, RMS values etc...

File structure

The most important files of Neapolis are:

- Used or created by the first part of Neapolis (Define System and the Menu)
**progname.dat**: this file contains the start drive, the storage drive, the short for the selected language and the short for the chosen circuit.

**FD-files**: are used by the Define System program and contain the default settings for the systems. The files are used to create the `model.dat`, `supply.dat`, `load.dat` and `graphics.dat` files, which are used by the menu system to display the default values for a circuit.

In the \Nea4\Texts directory you can find all the text files use by the 3 systems to support multiple languages.

**• Used or created by the Simulate System part of Neapolis**

**SY-files**: These files are created by the Define system program and contain the current settings or a chosen system.

**EX-files and OD-files**: the Simulate System program puts the different values obtained while simulating in this file.

**• Read by the Result Processing part of Neapolis**

Except the `progname.dat`, the SY-file, it read the EX-file and the OD-file written by the Simulate System program.
Directory structure

While developing a program you need a place to store the source code and forms for your program. (Of course the end-user never gets these)

Therefore there are a few directories created:

- **Convert** with the forms for the converters
- **Motors** with the forms for the motors
- **Drives** with the forms for the drives
- **Fileopen** contains forms for opening and using files
- **Forms** stores the forms for the 3 main parts of Neapolis
- **Def-Sys** stores the forms for the Define System program
- **Modules** contains the different modules used by Neapolis

For more information, see the other final project reports available in the lab of power electronics or the library of the TEI.
Starting to work

Visual Basic

As I had never programmed in Visual Basic before I came here, so I had to study the program language first.

One good thing about Visual Basic is the online book about the programming language that comes with the compiler. It explains the whole programming language briefly, even stating some examples.

Nevertheless, I needed information that I did not find in the online book, but on some Internet sites. You can find a lot of very interesting links, all about programming VB, at http://www.vbhelp.net/pages/links.html/#general.

The parts I needed

Afterwards I tried to understand how Neapolis works, what the general structure of Neapolis is.

The 2 parts I needed for this project are the Define System and Menu program and the Simulation System program.

In the Neapolis main project file (Neapolis.vbp, this is the Define System program) I had to find out how the menu system of Neapolis works and how to start another program from there.

From the Simulate System program (Neapart2) I tried to learn how to print the curves of a simulation on a form, because I had to do something similar.
**Visual Basic**

**Why**

For some topics I had some troubles finding the right statements in the books online. That's why I put this chapter in my report. It's meant to be a little help for the future Neapolis programmer that has to work with the Schematics program.

**Creating forms on the fly**

This example illustrates how to create a form at run-time. The new form is created when the Command1 button is clicked on.

The set command assigns an object reference to a variable or property.

**Syntax**

```vbnet
Set objectvar = { [New] objectexpression | Nothing }
```

- **The name of the variable or property**
- **Free the allocated memory after using association.**

The New option: this option enables you to create new objects at run-time. Combined with the set statement, it creates a new instance of the class.

Objectexpression: This is the name of an object type that has to be created.

```vbnet
Private Sub Command1_Click()
    Dim frm As Form1

    Set frm = New Form1
    frm.Caption = "Hello"
    frm.Show
End Sub
```

The source code in the above frame, creates a new form with "Hello" as caption. Create a button on a form, double click on it and copy/paste this code.
To create series of new objects of a specified kind, you can set the index property. The index of the “mother-object” is set to 0. The newly created objects can be used like a kind of array.

```
Private Sub Command1_Click()
    Dim frm(l to 10) As Form
    For I = 1 to 10
        Set frm(I) = New Form1
        frm(I).Caption = "Hello"
        frm(I).Show
    Next I
End Sub
```

**Creating textboxes**

Like the forms, the textboxes on the schematic form have to be created dynamically too. But they can’t be created in the same way as the forms.

**Syntax**

```
Load object
```

The only thing that has to be done, is to put a “mother”-object (just like with the forms) somewhere on the form and change the index property from nothing to 0. Also put the `Visible` property to `False` because at run-time the mother-object is not used.

Now you can create new object like the mother-object. Of course you’ll have to move the object to the right place on the form at run-time.

```
For I = 1 to 10
    Load Textl(i)
    Textl(i).Visible = True
Next I
```

The newly created textboxes (in this example) are invisible (just like the mother-object), so you have to change this also.

For changing the position you can use the `Text1.Move` method. This is faster then changing the `Text1.left` and the `Text1.top` properties.
Already loaded?

Another useful feature is checking if an application is already loaded because it's not necessary to start the same application twice. Instead of starting another instance, it's better to just give the existing one the focus. Only a few lines of Visual Basic code are enough.

```vba
If App.PrevInstance Then
    AppActivate "application-title"
End If
```

The `App.PrevInstance` returns `True` if the program is already loaded. Then, the only thing to do is give the focus to this program and exit the one that is being loaded.

Application-title is the text in the title bar of the window (is of course the same as the caption of the form of the second instance that is being loaded).
Schematics

Overview

As I had to write a new program and implement it in Neapolis, I decide to do it in the same way as the other programs are implemented. The figure shown below illustrates the new structure of Neapolis.

The Menu & Define System program had to be changed so it was able to load the Schematics program, in the same way as the Simulation program or the Results Processing part of Neapolis.

Take a look at the figure below.

![Diagram of Neapolis structure]

**Figure 4** New structure of Neapolis
The main window

The figure below shows the Schematics main form; it's the first form the user gets to see. It’s pretty easy to understand.

![Figure 5 The schematic main form](image)

It features:

- A Start/Stop button: to start and stop the simulation (obvious)
- A Pause button: if you want to have a break
- A Slider: to change the speed of the simulation

Click on the start button to start the simulation.

If you click on one of the text boxes a new form appears with the waveform of simulating circuit.
Figure 6 The Schematics program in action
Changes in Neapart1

Button

The first thing I did was adding this Schematics button to the main menu of Neapoli.

Figure 7 The new buttons on the main form

The Schematics button had to react in the same way as the Simulate system button. They are two more or less similar programs, with the same input files. So I started to look at the code behind the Simulate System button and adapt it to the Schematics button.
Menus

Of course I had to add the Schematics option to the Action menu too and adapt the language files.

Figure 8 The new menu option

In the same way as the button, I changed the menu system of Neapolis so that the user is also able to choose the Schematics program from the menus.

Text files

Neapolis uses text files for storing the different text string of supported languages it puts on the menu and the buttons. Those had to be altered too.

You can find the text files in the \Nes4\Texts\TextsDef directory.
New schematics program

The name

This program is located in the Schematics subdirectory of the Neapolis home directory. The only reason I chose for "Schematics.exe" in stead of "NeapartX" because it's more understandable.

Basic structure

![Diagram of basic structure]

**Figure 9 Basic structure of the Schematics program**

As you can see the schematics program uses several input files (the parallelograms above the rectangle). As output, it has only interaction with the...
user (user screen). The System Definition Program of Neapolis creates some of the input files, some other files especially created for being used by the schematics part of Neapolis.

Created by the System Definition Program:

- *progrname.dat*: this text file contains the startdrive, the storage drive, the language and the kind of system that was chosen with the Define System Program

- *program.dat*: stores the number of periods that has to be simulated and the amount of calculations per period

Schematics Files:

- *?_schema.TXT* can be: where the “?” can be E (English), G (Greek), N (Dutch) or D (German)
- *.*.WMF or *.BMP*: Windows MetaFile or BitMap of the schematic
- *.*.COO*: co-ordinates file
- *RelCon.ALL*: links the type, controltype and kind of load to the appropriate schematic graphic
- *SYRelCon.DAT*: System variables of the chosen circuit
The SelectSchematic process is an invisible one. The form it runs on is not shown to the user. It loads the necessary variables from the text files to be able to decide which from has to be loaded next. Re1Con1 is an example of this.

The Re1Con1 process performs the simulation of the selected circuit.
Starting the program

One of the disadvantages of using a 'shell'-program that starts the other applications (and using MS Windows™) is that the user can swap back to the shell-program and run another instance of that particular program. Therefore, every time the schematics program is started, it checks if it is already loaded into the memory. In the case it is, it exists immediately.

Metafile or bitmap

The major difference between those to file formats is that WMF (Windows MetaFile) is also vector oriented and BMP’s are only Bitmap files. The most important
consequence is that WMF's are (if you use only vectors) small and perfectly scalable, and so the ideal file format for this job.

Another reason I used WMF is the fact that I encountered some problems while scaling rather big files with Visual Basic.

### Multilingual

Neapolis is designed to be able to work with 4 different languages. Therefore all the text you see on the screen in menus, labels or on buttons etc. are stored in text files.

I designed the Schematics program in a similar way.

During the initialisation, the program reads the "?_schema.txt" file, which contains the captions and text lines for one specified language.

Like in every text file created for the Schematics program comment lines can be added just by putting a ";" as first character of a line.

### Dynamic creation of textboxes

In order to make it easier to add more schematics in the future without writing the same code over and over again, the Schematics programs had to be as flexible as possible. That's why all the textboxes used by the simulation procedure are created on the fly. The program reads the necessary data (the x and y co-ordinates in percentage of the form, the length of the textbox and the caption of the matching form) from a text file. The format of the filename is "?anyname.coo" (coordinates), where "?" is the language abbreviation and "anyname" the filename from the .ALL file.
Dynamic creation of forms

Like the textboxes, the forms on which the curves have to be drawn have to be created in the same way, at run time. The format of the filename is "name.ALL" where name is the short for the kind of circuit (e.g. RelCon, Re2Con...). When the Schematics program reads this file, it compares the first 3 words of each line with the data provided by the Define System. When they match is found the following word of the text line has to be the filename of the schematic graphic and the last word the co-ordinates filename.

Plotting onto the forms

The two major statements to draw on a form are PSet and Line.
PSet puts dot on an object. The advantage of PSet is that it only used a x and a y co-ordinate. But the killer drawback of PSet is that if the y co-ordinate (amplitude) rises fast the dots gets separated which means there is no actual line drawn.
Therefore I had to use the line statement and find a solution for its disadvantage of having to keep in mind at what co-ordinate the previous line ended.

Scaling

The drawing procedures of the program are designed in such a way that they scale the curve automatically. The amplitude is recalculated so that the height of the curve is close to the next hundred. For example: in the figure below, the amplitude of the supply voltage is 170 V, so the maximum on the scale is 200 V.

Radarline

In order to make things a bit easier I implemented a "radarline". It's a line that appears on every form of the simulation at the same place. In that way, it's easy to see what is happening at a specific time in the simulation on all of the curves. To enable the radarline, just click on one of the forms with the left mouse button. You can move the radarline by moving the mouse to the left or to the right on one
of the forms while holding down the left mousebutton. To disable it again, use the right mousebutton.

![Supply Voltage](image.png)

**Figure 13 The radarline**

**Closing the program**

All the different curves that were selected by the user are plotted on newly created forms. In stead of closing all those curve-forms one by one, I implemented them so that they ‘follow’ the main form’s state. This means that if the main form closes, minimises or maximises they all close, minimise or maximise.
Implementing new schematics

In this chapter I’m trying to explain how to implement the other and new circuits to the Schematics program. It should be very easy. Just follow the step described under here.

Changing the data files

The Circuit

Draw the circuit with some program that can export drawings into Windows Meta File (.WMF) or Bitmap (.BMP) format. I used Corel Draw™ (by Corel Corporation) because most electronic drafting programs can not export to a vector oriented file format.

The text files

Comment lines can be added at any place in all the text files used by the Schematics program just by placing a “;” at the beginning of the line.

- Open the RelCon.ALL file with a text editor (eg. Notepad)
  The first 3 words of the line are the type (1 phase, 3 phase…), how the circuit is controlled (no control, semi or fully controlled) and the kind of load (resistive or reactive). The shorts for these parameters must be exactly the same as the one from the Define System program. The fourth word of the line is the filename of the circuit file. The fifth word is the name of the file where the co-ordinates of the textboxes are stored. Make sure that the first letter (language) is not written in that line because it is added automatically in the program. For example: RelCon1.coo is written in the .ALL file and real filenames are ERelCon1.coo, NRelCon1.coo,…

; Listfile of all possible 1 phase rectifiers
; --------------------------------------------------
; last changed 991308
Implementing new schematics

; empty lines must begin with a ';'  
; 
; RectType TectControlType LoadType SchematicFilename  
; CoordinatesFilename  
; IP- No Resis 1phRec1.wmf RelCon1.coo  
; IP- No React 1phRec2.wmf RelCon2.coo  
; End Of File

- Open the ERelConX.COO file with a text editor
  
  In this text file the line above the co-ordinates is the caption of the form where the curves are drawn on. The first letter of these lines must be a "#". For this file there are of course 4 versions, one in every language.

; Coordinates of textboxes for 1 phase rectifier  
; English caption lines  
; Format : Xco Yco Width  
; (in percentages)  
; Textline starting with "#"  
;  
; #Supply Voltage  
022 047 15  
; #Diode Voltage  
045 023 15  
; #Load Voltage  
070 049 15  
; #Load Current  
020 011 15  
; End Of File

**The new form**

Open the Schematics project with Visual Basic
The first thing to do is add a new form to the project

Put the needed objects on the form and change their properties
The textbox control

 TextBox control

 Text1.Index = 0
 Text1.Visible = False

 The Image control

 Image1.Stretch = True

 The Button controls

 PauzeBtn.Caption = "Pauze"
 StartBtn.Caption = "Start"

 The Slider control

 Slider1.Max = 500
 Slider1.Min = 1
 Slider1.Selstart = 1
 Slider1.TickStyle = 3 - sldNoTicks
 Slider1.Value = 1

 The Label controls

 Fastlabel.Caption = "Fast"
 Slowlabel.Caption = "Slow"
The source code

The Forms

- **SelectSchematic Frm**: this is the main form of the Schematics program. It loads the necessary files *(proname.dat, program.dat, the language file _schema.txt,...)* to be able to select the appropriate form for the simulation.

- **Form1**: this is the dummy form to plot the curves on. Dummy form means that it is not actually used (it has the index 0) but the other forms are create from this one dynamically. This form has an index, so I was able to create more forms like this one from the data of the text file. This form is used as a kind of a template for the new ones.

- **RelCon1 Frm**: this is a typical form for the 1-phase rectifier. This form can be used as an example to implement the other circuits. This one is called from the `invisible` **SelectSchematic Frm** form and is the main form the user sees.

The Modules

- **Global Variables**: Like the name says, this module is used to store the global variables. Global variables are variables that have to be able to be accessed from anywhere in the project.
• **Utils**: This module contains procedures of functions that are not typical for a specified form, but can be used generally.

**The procedures**

The most important procedure in the code is the `FillTextBoxes` procedure. This one calculates the values that have to appear in the textboxes and runs the procedures that are supposed to draw the curve(s) on the form(s) depending on which forms are opened.

But, as I said before, there are enough comments in the source code to understand what it's doing.
Conclusion

The Neapolis program

About the Neapolis program, I fully agree with the conclusion of Jo Van Den Berghe (Belgian Socrates student 1998).

The first versions of Neapolis ran under the DOS (Disk Operation System). Nowadays it’s an MS Windows™ program, but the major structure of the DOS version does still exists, making the program slower and more complex than necessary.

Another drawback of Neapolis is that it is expanding every day and everybody which is new to the Neapolis source code has to spend a lot of time studying the code of the program because it’s getting really big and complex. A decent analysis of the program would be a big help to the future student-Neapolisprogrammer.

Anyway, after the few misunderstandings that occurred, it feels great to know that people are really using the program that I also worked on.

And, now, there is one more program language I know the basics of...

Socrates

About the exchange project, I can only be positive. I really learned a lot about our host country (and the people) and of course about the other foreign students, because I spent most of the time with them.

It’s funny when you realise how much you can say without knowing the language very well.

Even though I missed my friends, my family and my girlfriend very much, I never regretted I took this once in a lifetime opportunity.

Thanks guys and girl for the time we spent together!

And see you later!
# Table of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Neapolis program structure</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Define System: 1-phase rectifier</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Simulating the 1-phase rectifier</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>New structure of Neapolis</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>The schematic main form</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>The Schematics program in action</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>The new buttons on the main form</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>The new menu option</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>Basic structure of the Schematics program</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>Schematics structure detailed</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>The SelectSchematic process detailed</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>The simulation process detailed</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>The radarline</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>Implementing a new main form</td>
<td>29</td>
</tr>
<tr>
<td>15</td>
<td>The forms and modules</td>
<td>30</td>
</tr>
</tbody>
</table>