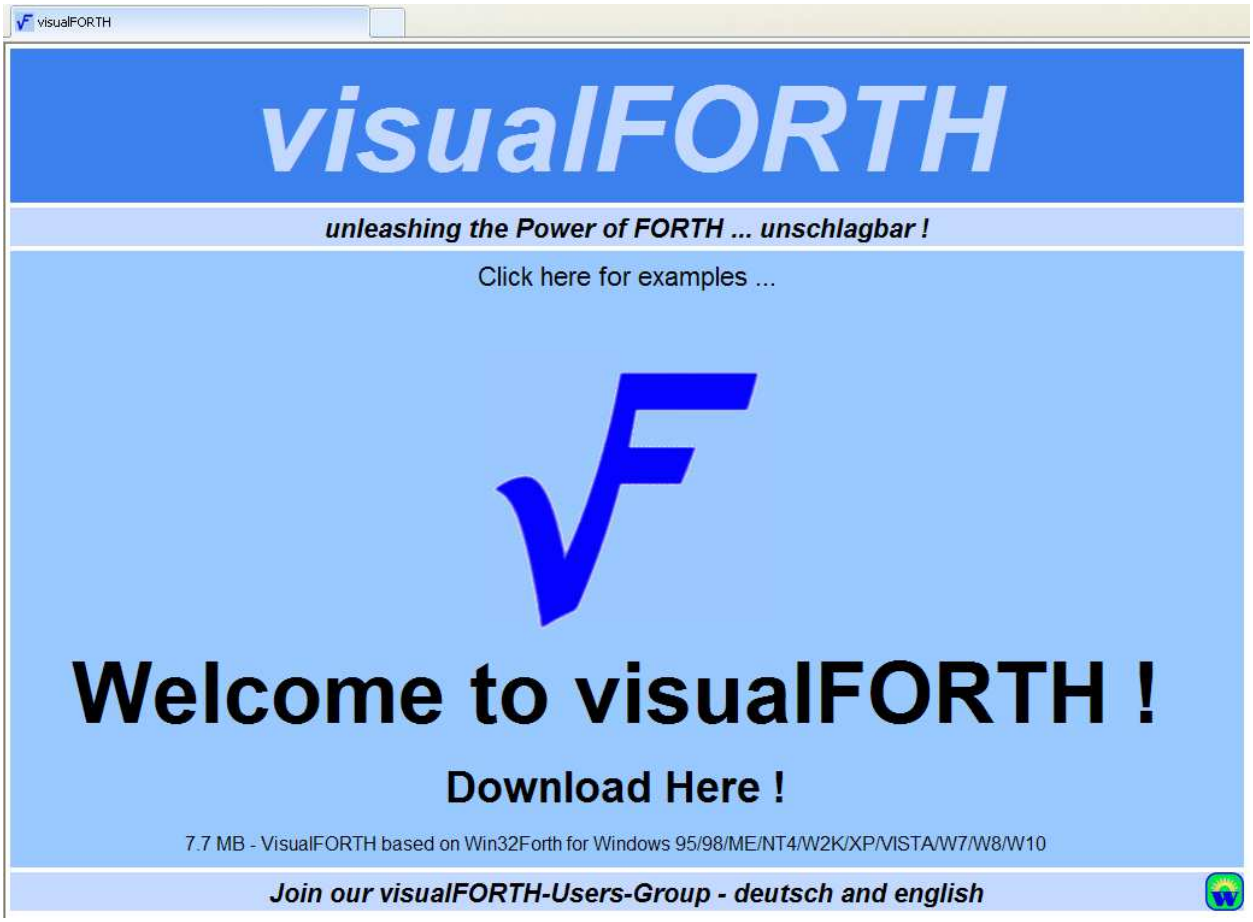
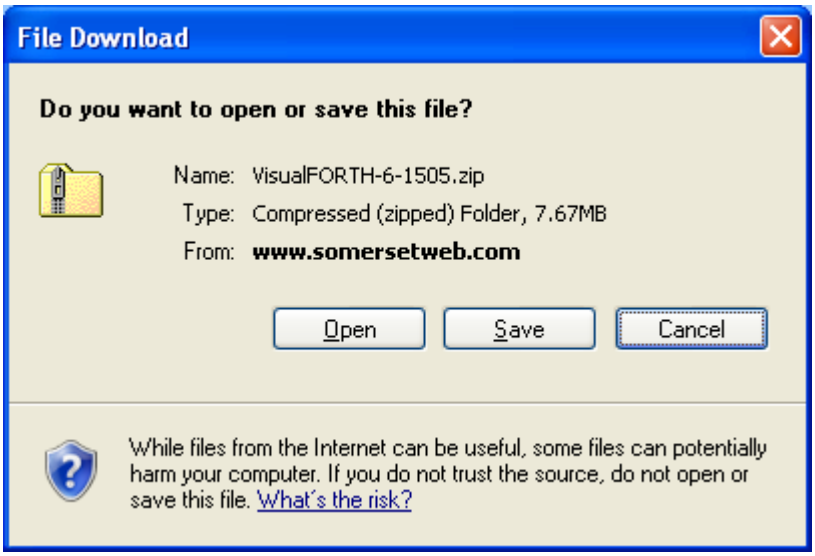
Starting VisualFORTH

1. Download

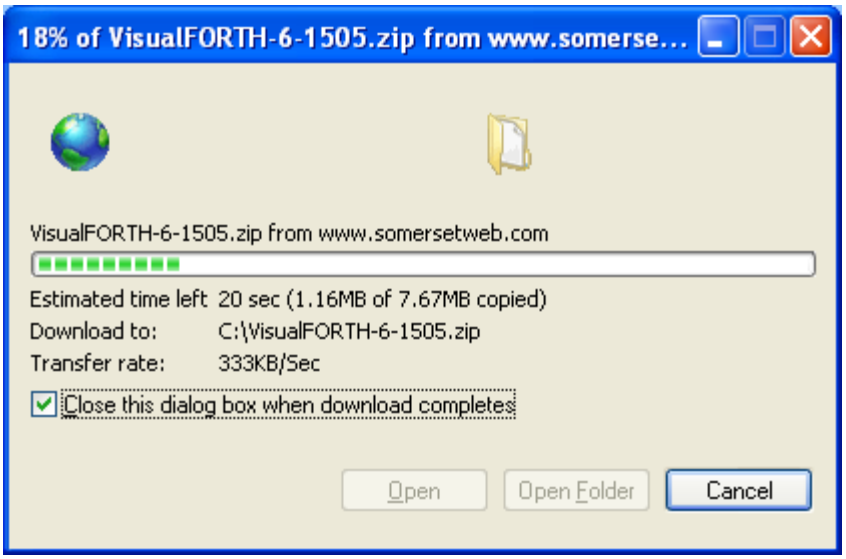


The above picture shows our http://www.visualforth.org/ website.
To download VisualFORTH please click on **Download Here !**

A File Download invitation will show up:



Click on Save, select Root-Folder C:, and use “Save” to save this zip-file
from http://www.somersetweb.com/visualFORTH/VisualFORTH-6-1505.zip onto drive C.

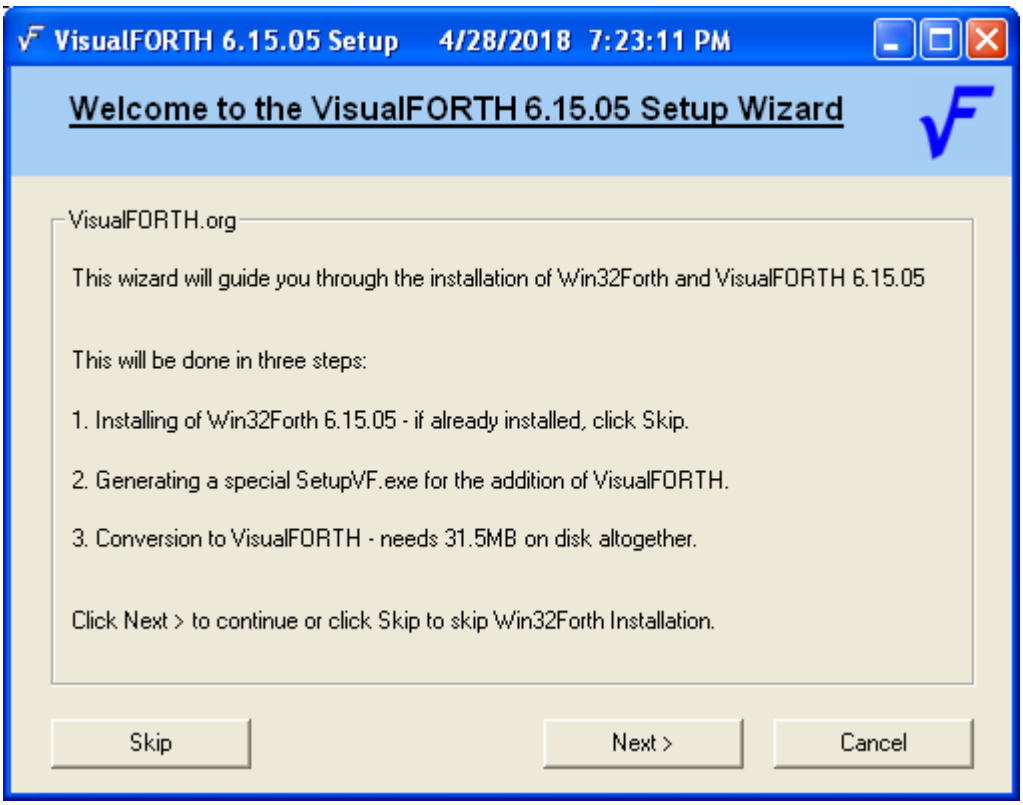


2. Installation

2.1 Unpacking the Zipfile and Installation

The VisualFORTH-Zipfile is now located inside Root-folder C.
Unpack folder VisualFORTH-6-1505.zip.
This folder includes a ReadMe file, the folder “visualFORTH-Batch”, holding folders with the VisualFORTH-Files which are needed to upgrade Win32Forth to VisualFORTH, and the self-unpacking Win32Forth-File w32f61505.exe and the VF-Wizard.exe.

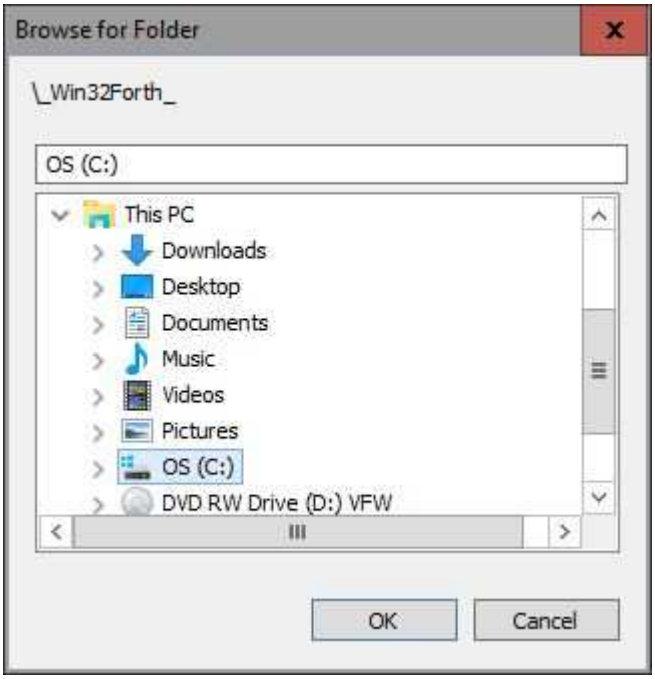
A click on VF-Wizard.exe starts our VisualFORTH-Wizard:



Now the installation is easy.

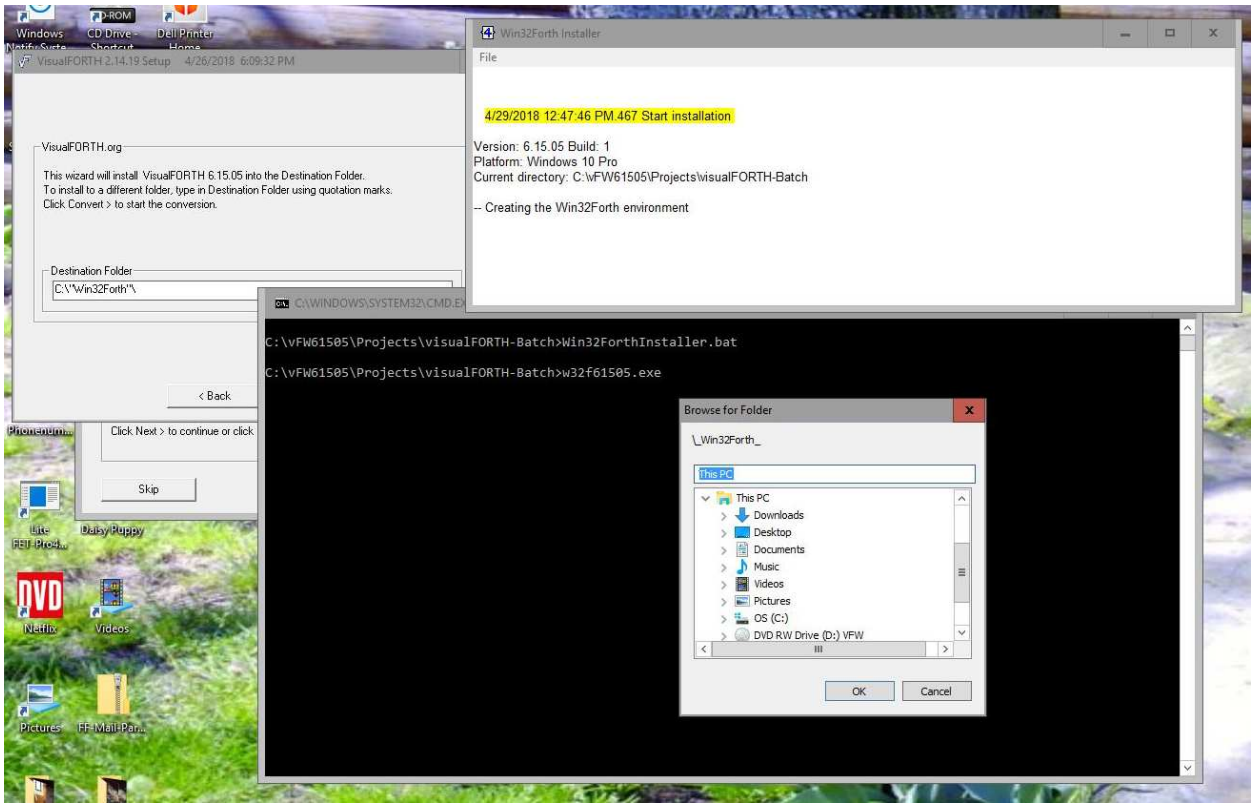
A click on “Next >” starts the installation of Win32Forth which may need a while.

A Browse for Folder will pop up:



Please click on “OS (C:)” to position Win32Forth in folder C.

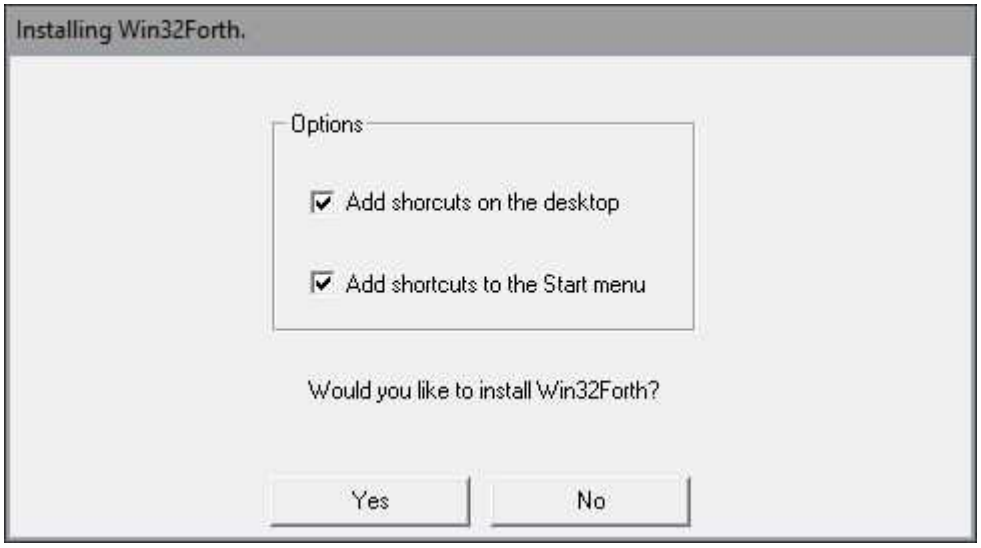
The Win32Forth Installation will proceed:



This line with yellow background tells us date and time of Win32Forth installation start.

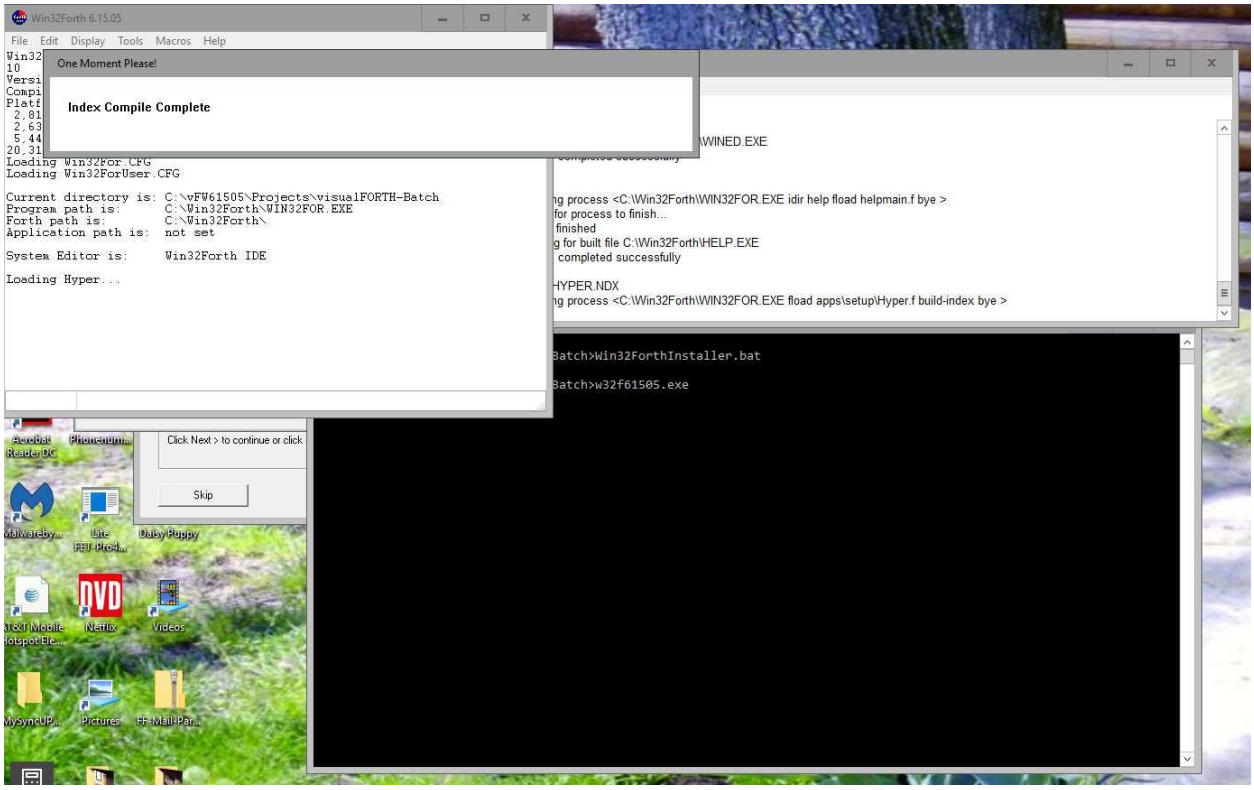
A Win32Forth Options pop up gives the opportunity to select shortcuts for desktop and Start

menu:

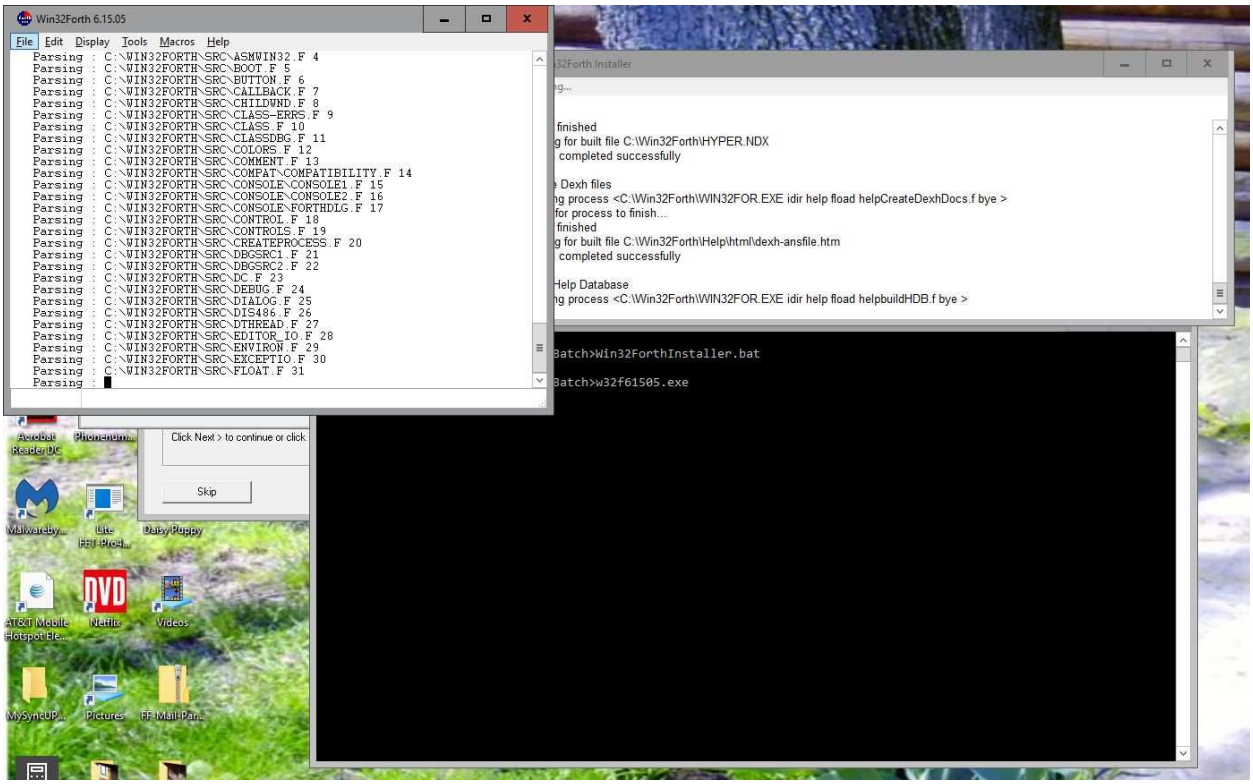


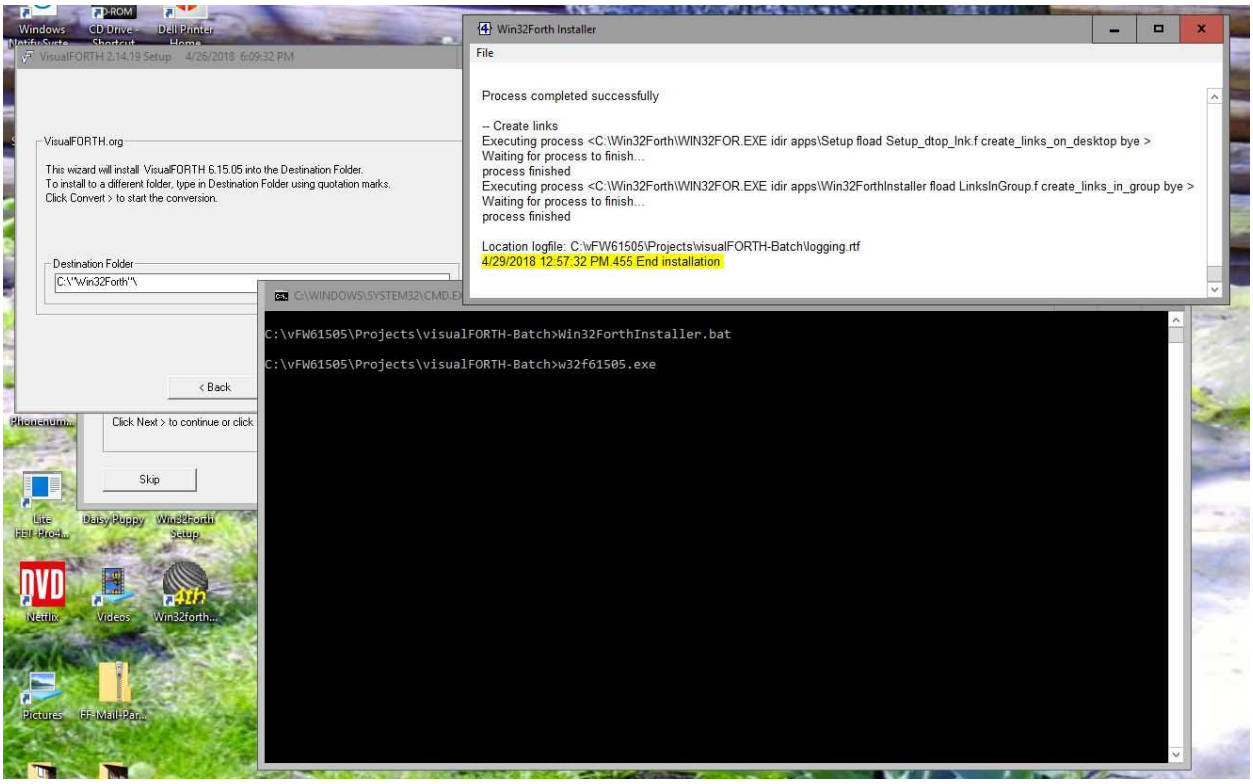
Click on “Yes” to proceed with this installation.

Several windows will show up with rapidly moving content, like this:

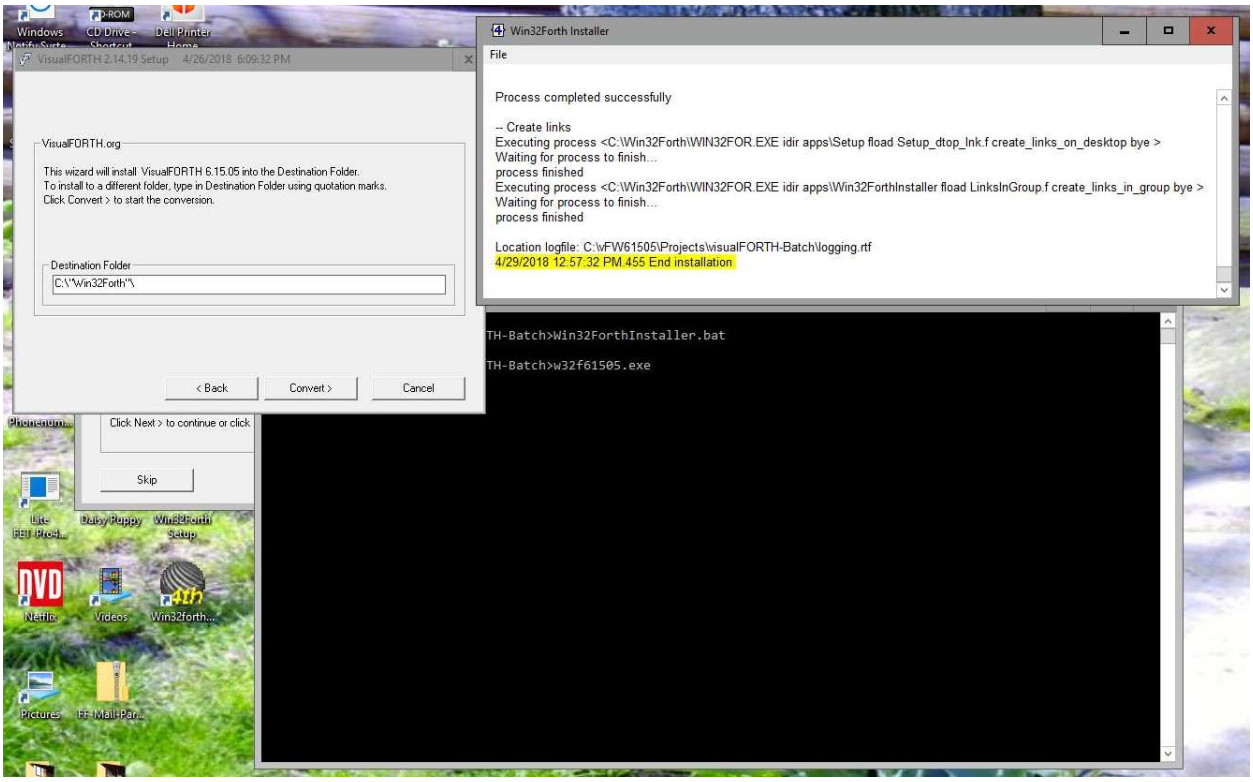


And like these on next page:



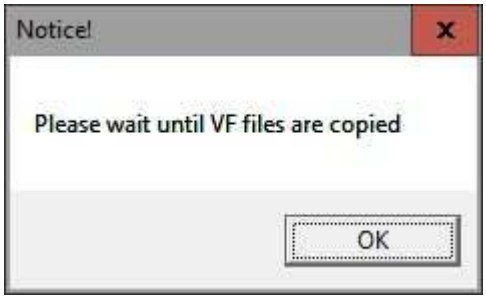


The end of the Win32Forth installation will be notified with another line with yellow bakground, see picture above.
When this occurs, click on the top page of our vF Setup-Wizard to bring this page on top:



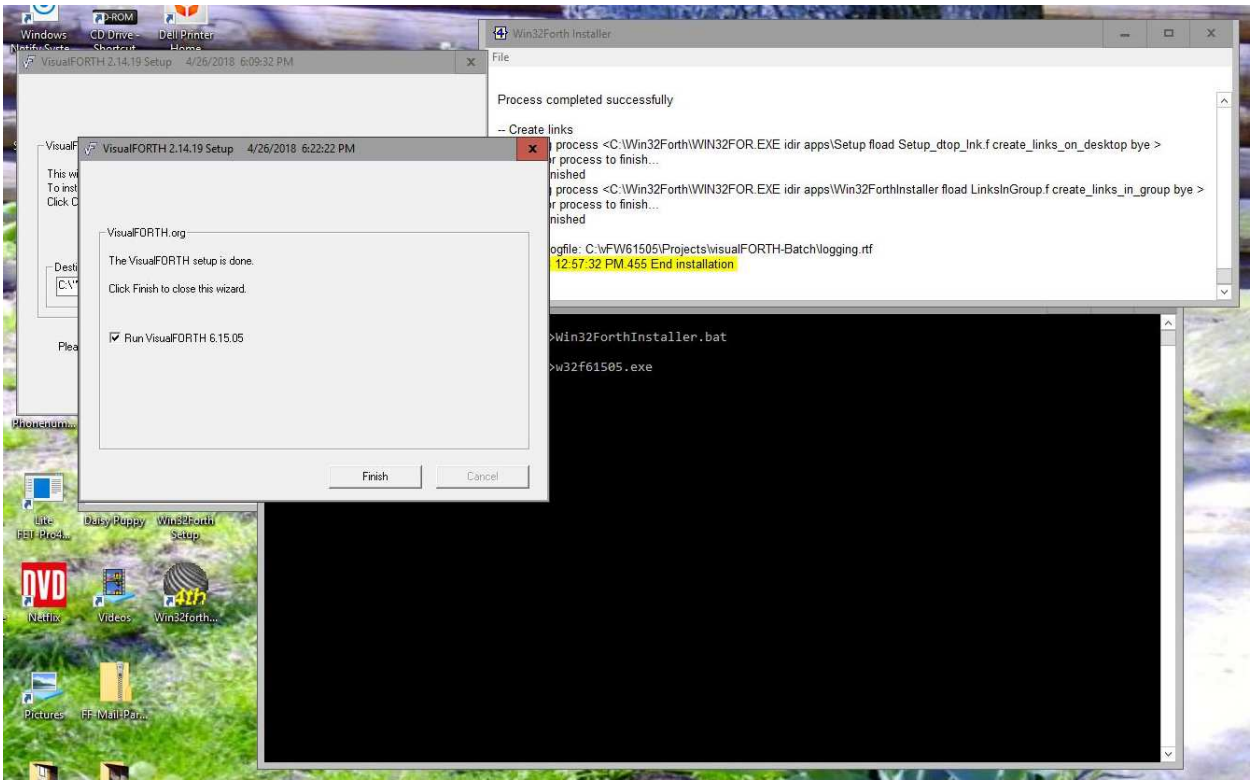
Next Step:
A click on “Convert >” adds VisualFORTH to the just finished Win32Forth installation.

Some windows will show up for these conversions, and two times a pop up asks to click on “OK” to proceed:

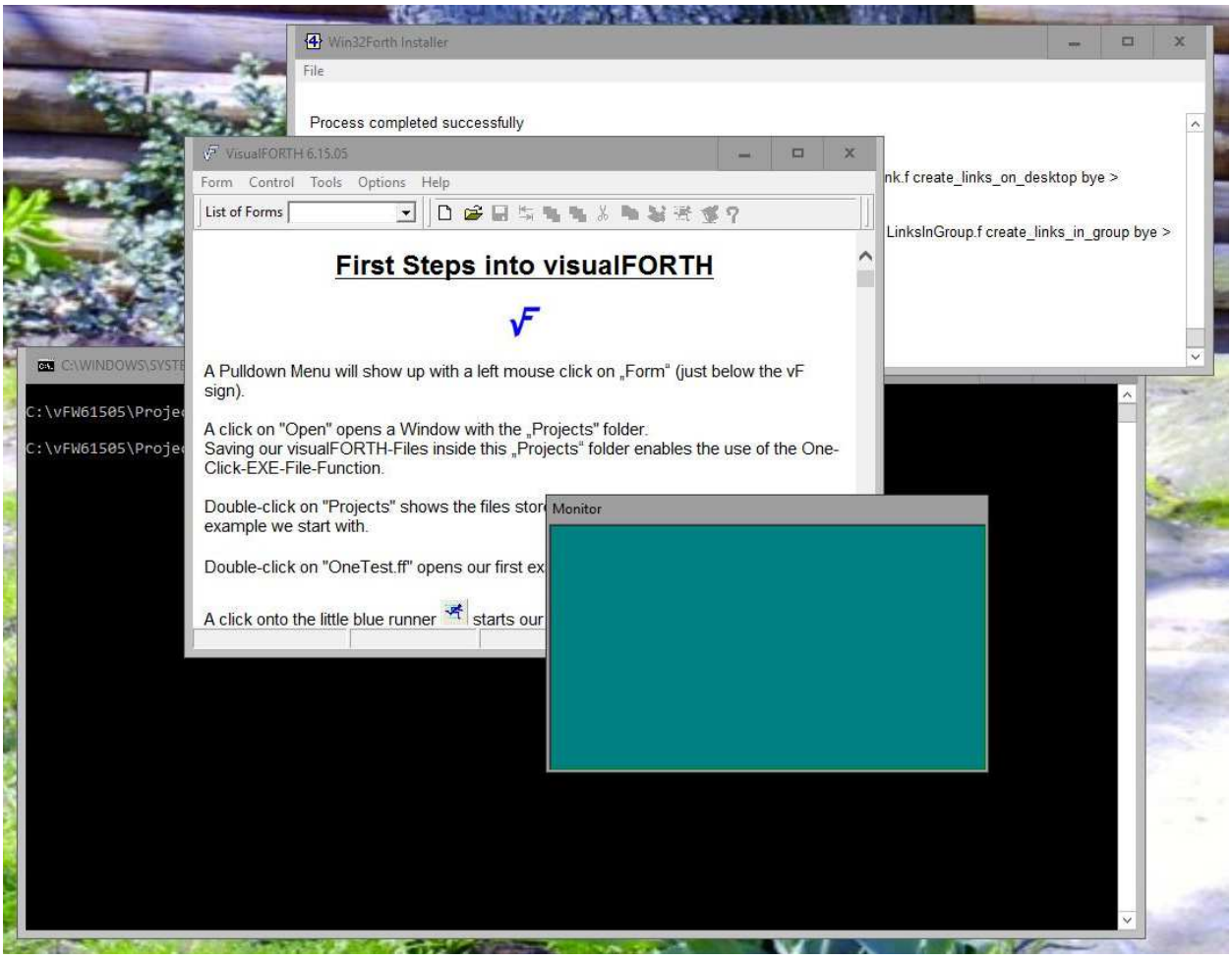


The first one shows up while copying the vF-files (see above), and the second shows up while finishing the setup file SetupVF.exe to finally setup VisualFORTH (below).

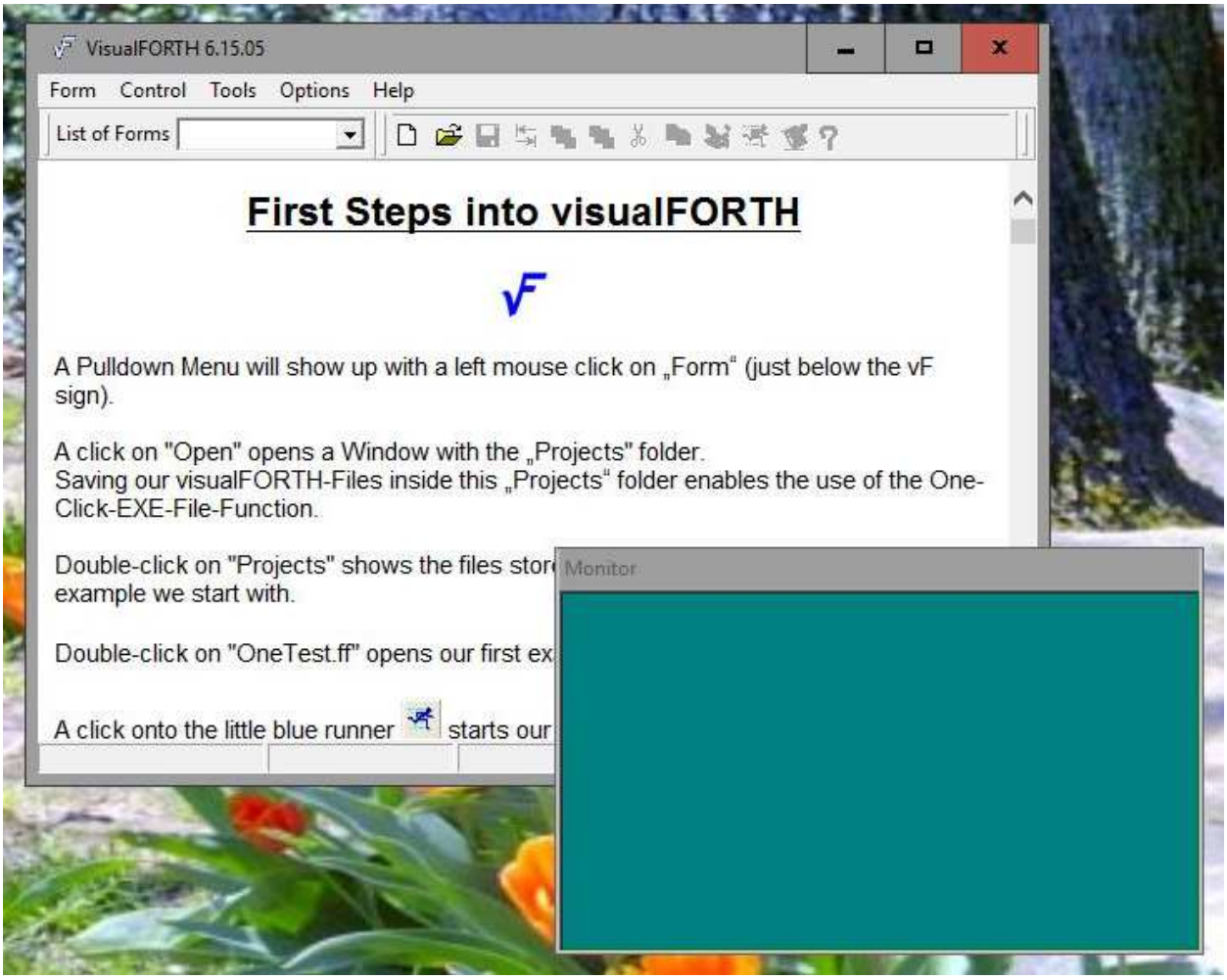
Last but not least our vF-Wizard waits for it's last step:



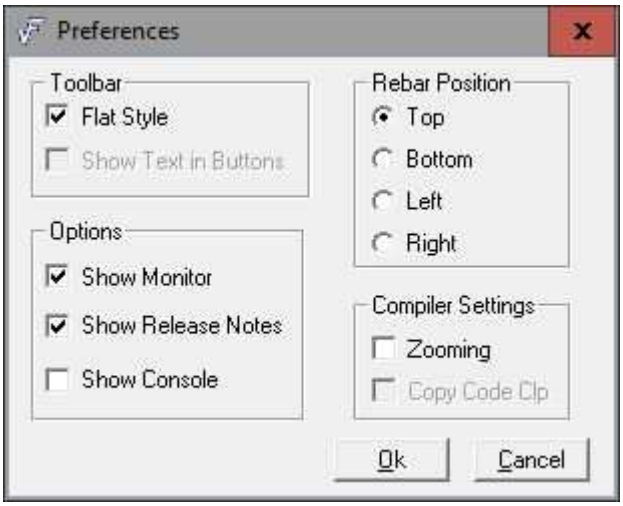
With a click on “Finish” VisualFORTH will start:



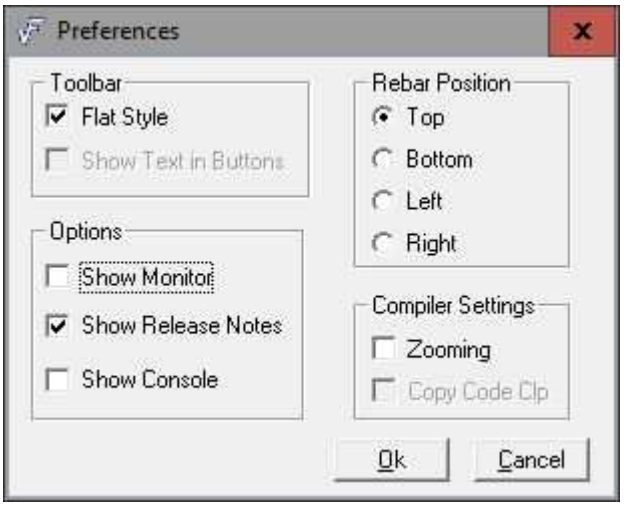
We only need this:



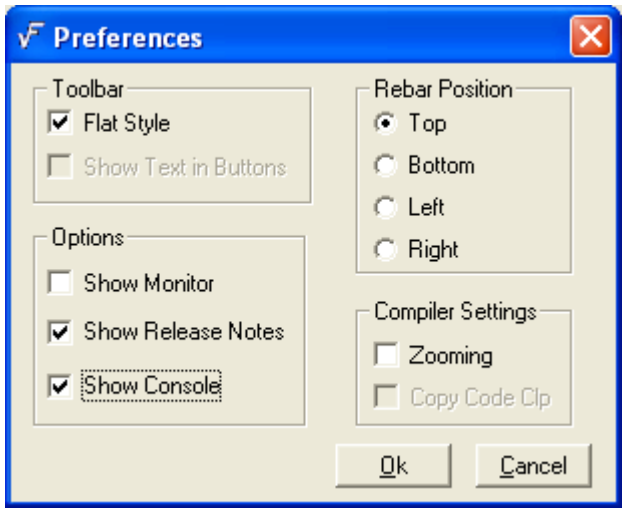
The green window is a monitor window which shows where our to be developed GUI is located. We later can use it to locate our GUI on a desktop place we wish to have.
But for now we can omit this monitor. A click on “Options” and “Set Preferences” gives us this pop up:



We remove this hook at “Show Monitor”, and with a click on “Ok” the monitor is gone:



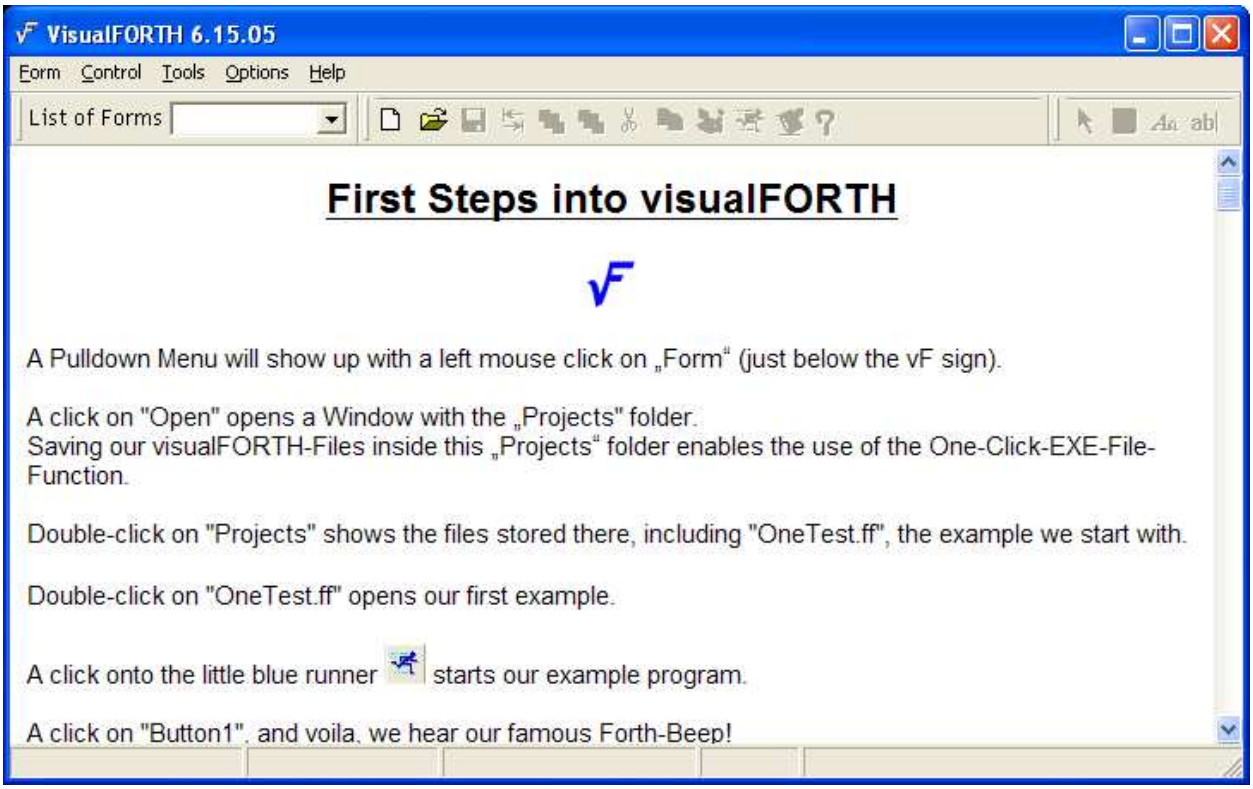
For testing purposes it would be great to have the Win32Forth console:



Another way to get Forth Console is with click on “Tools”, “Forth Console”, with certain error messages, and to show with click on “Control”, “List Controls”, “Copy to Console” the Controls used by the active Form, and with click on “Control”, “List Functions”, “Copy to Console”, the functions, respectively. The last two only work with an open Form.

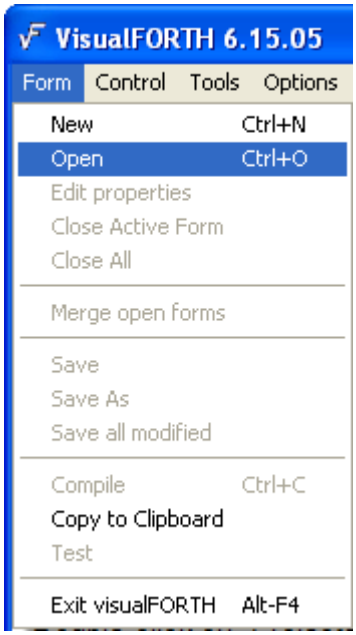
This Win32Forth Console is specially prepared for Forth Form / VisualFORTH, but if you type in Forth <Enter> all of Win32Forth is accessible.

3. First Steps in VisualFORTH

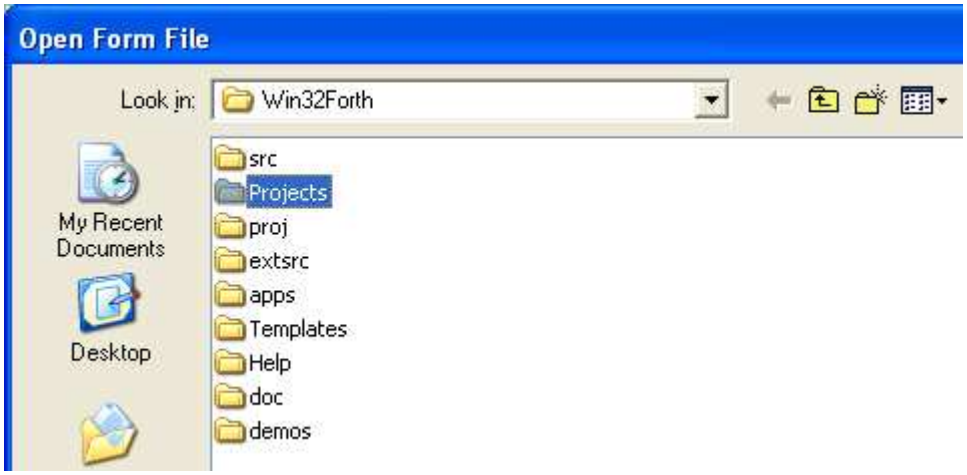


3.1 A First Example

Let's look into a first example!
A click on "Form", and a Pulldown-Menu shows up:

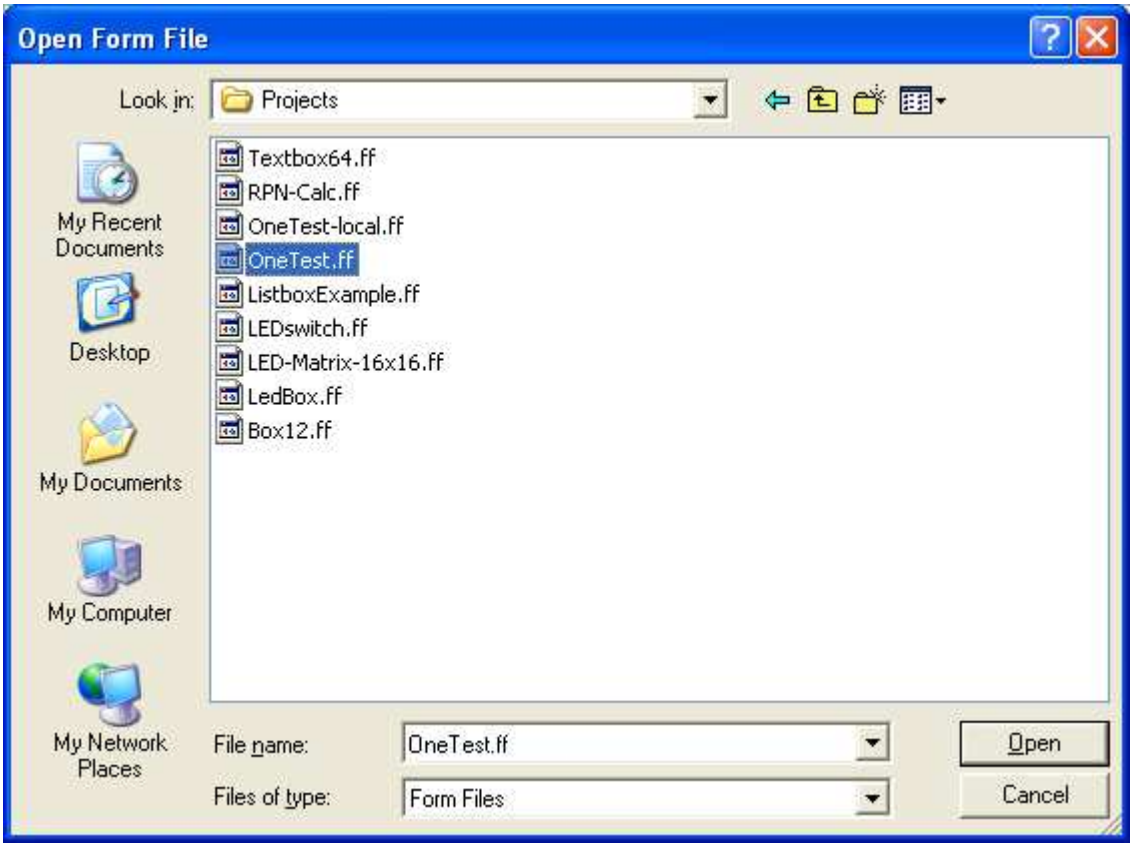


A click on "Open" opens a window with the folder "Projects":

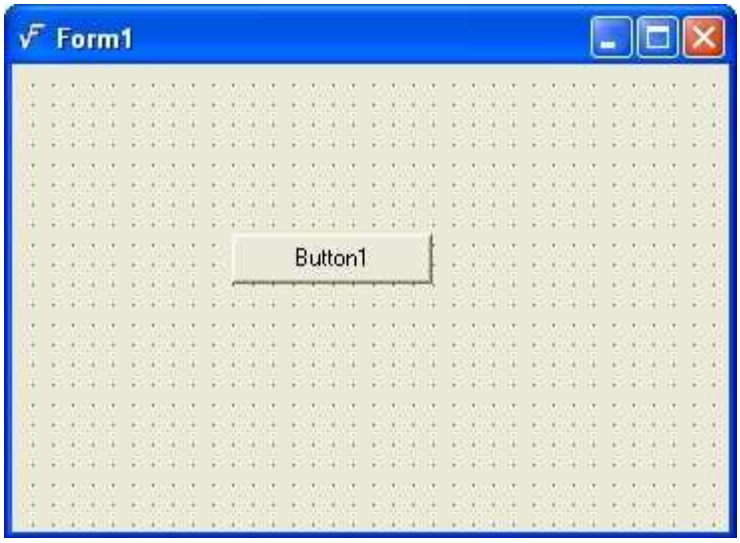


To use the One-Click-EXE-File-Generation our VisualFORTH-Files have to be saved into this folder “Projects”!

A doubleclick on "Projects" shows our example "OneTest.ff", which we use now :

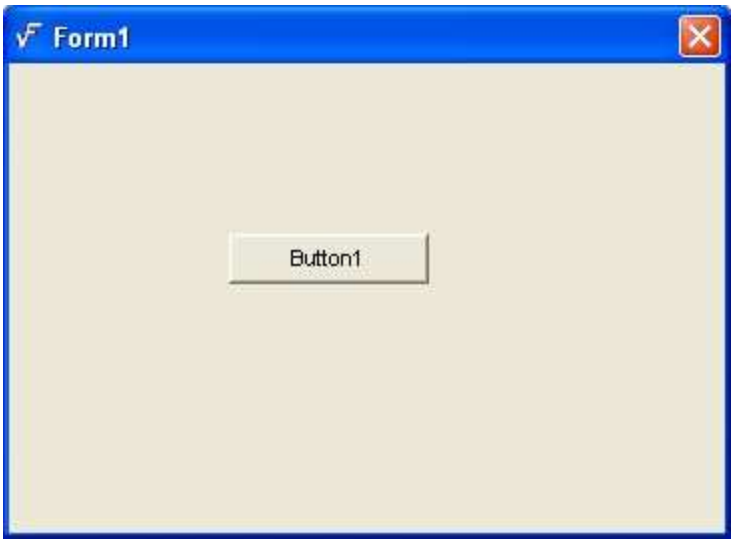


A doubleclick on "OneTest.ff" opens our example:



A click onto the little blue runner in the Menu line

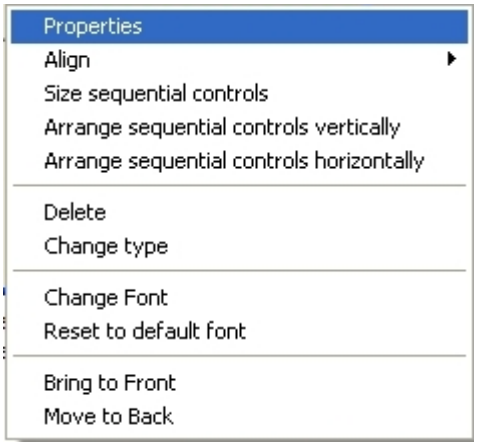
starts the example program:



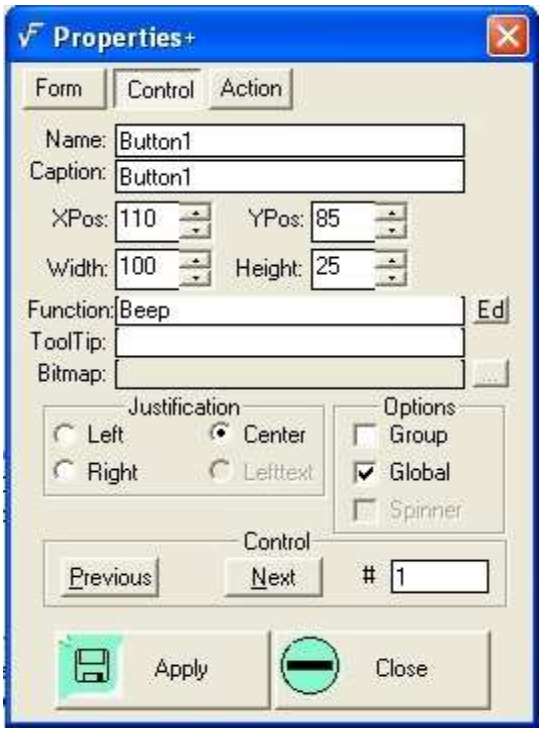
A click on "Button1", and we hear our famous Forth-Beep.
It sounds a little bit different with Windows 10, more like a “blop” than a beep.
If not, check the audio-options of your computer!

No we like to see how this worked. End Form1 with a click on the upper right **x** or move the top example-window, so we can see this Form with the grid again.

A right mouse click on "Button1" produces a Popup-Menu:



And a click on "Properties" pops up the "Properties+" window:

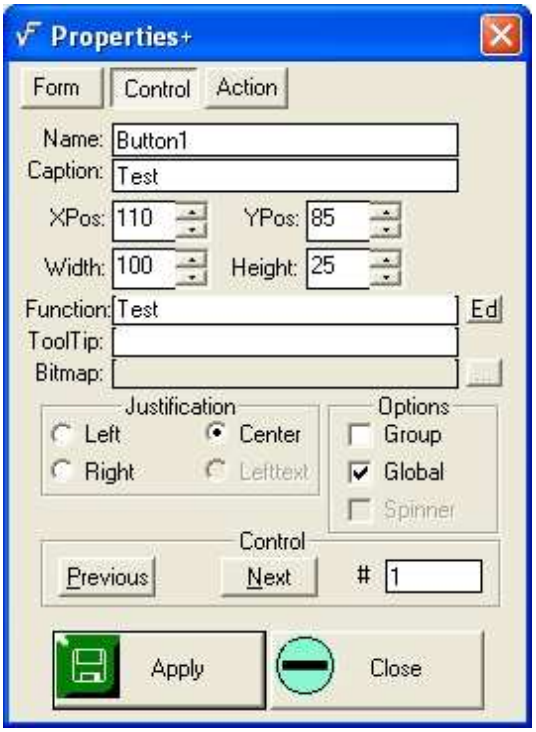


There we see the Word "Beep" at the right side of "Function" -
This is the secret of our function!
That's how we get it to work!

To tell the Button what to do when clicking on it, we type the predefined function word (which may be defined by another file) into this field at the right side of “Function”, and click on “Apply” to make it work.

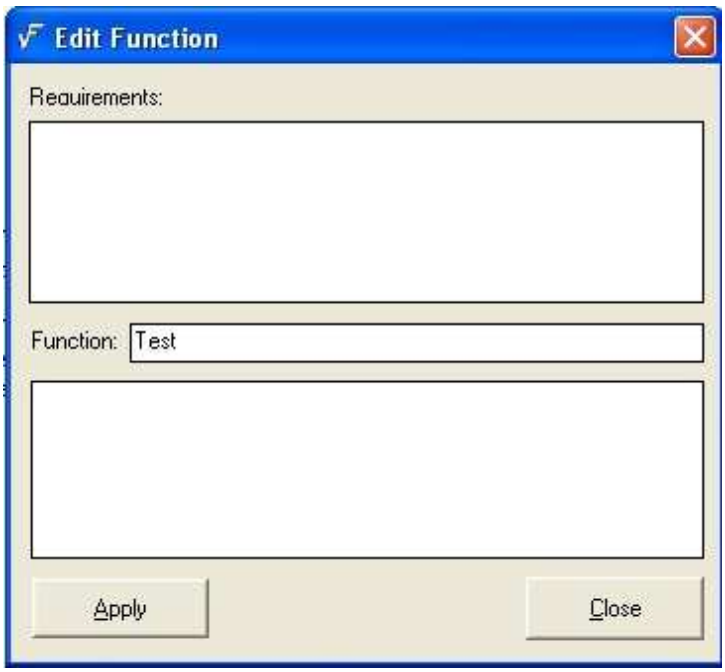
3.2 Next Steps

This works not only with a predefined word, we may define a new word here:



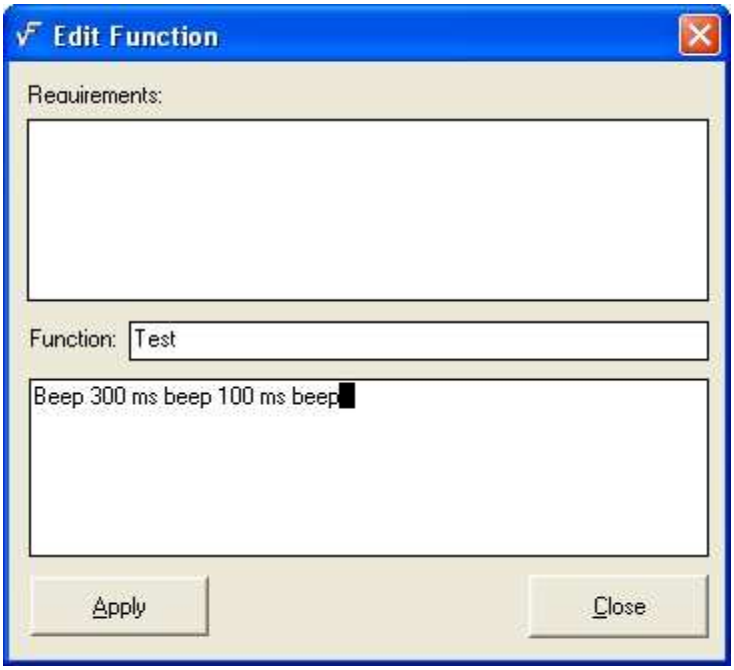
As you can see, instead of "Beep" I wrote the word "Test".
To accept this change, you need to click on "Apply".

No we like to define the action (function) of our new Word "Test".
A click on "Ed", far right side of function, opens another window:



"Test" is already there. Now we may write our new definitions into the Textbox beneath “Function:”.

These definitions will be executed with a click on “Button1” after compiling, one after the other.
Here you see an example for easy testing:

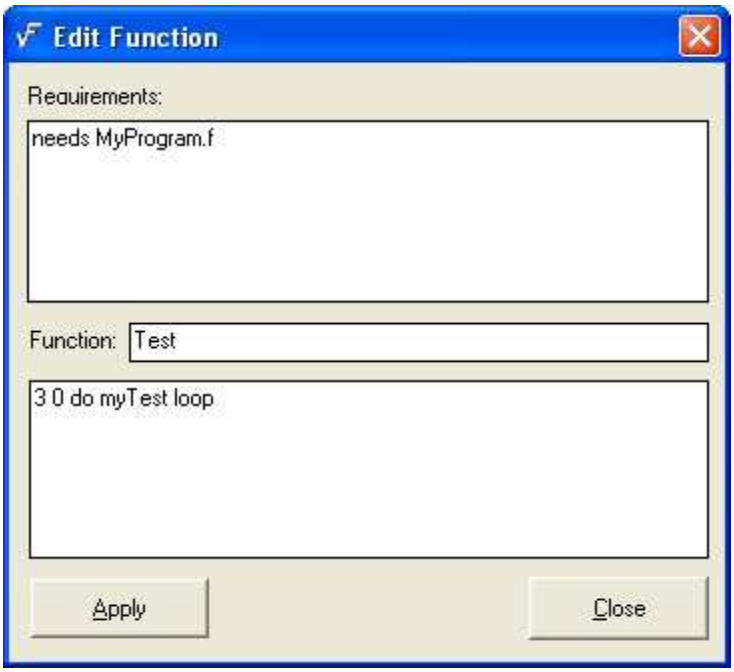


A click on "Apply" saves this input, a click on the little blue runner compiles this Form, ready.
A click on “Button1”, and we hear our Beep ... Beep ... Beep.
That's really simple, isn't it?

Of course we can do more complex programs:
A new click on "Ed" opens our Edit Window again.

Let's assume, we already have a Program which needs a GUI, a Graphical User Interfase.
Let's assume, this program's name is "MyProgram.f"; then we type:

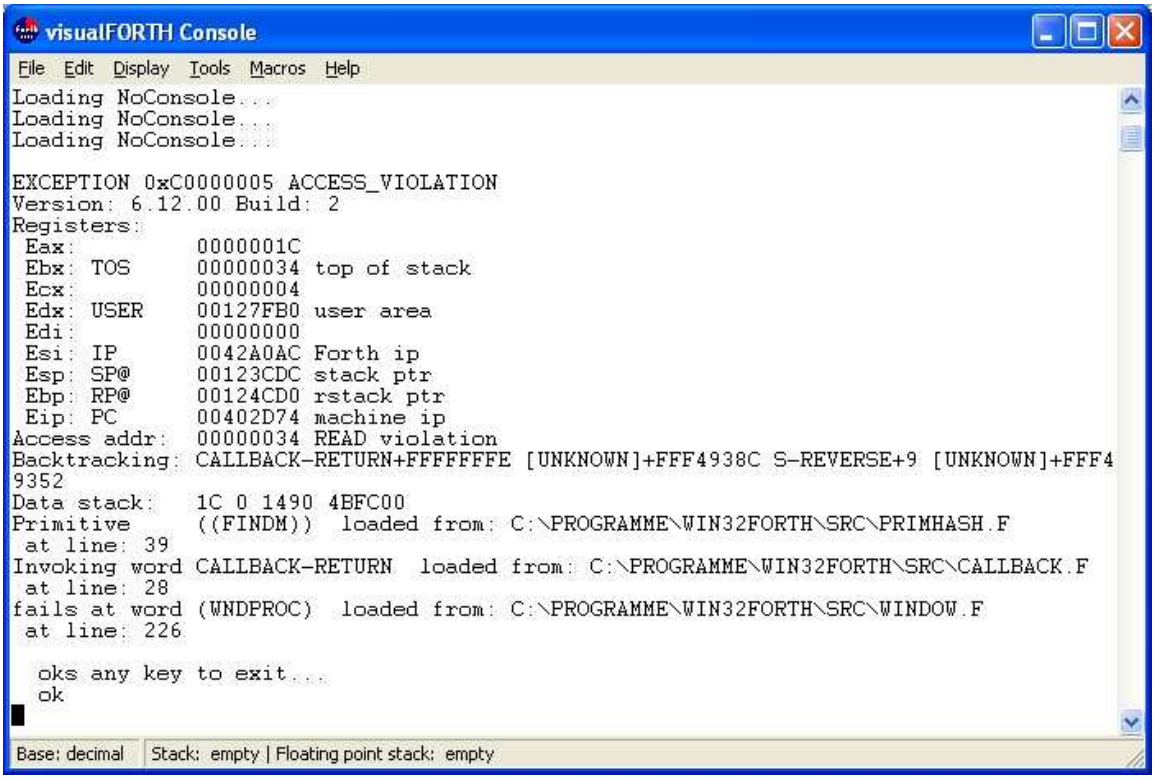
```
needs MyProgram.f
```



With this example we assume that MyProgram.f defines the Word "myTest", which has to be executed three times when clicking on „Button1“ (after compiling, of course).
Of course we may use "Requirements" to define Constants, Values, Variables Word-Definitions (written as usual with a colon at the beginning and a semicolon at the end) , as far as we do not exceed 128 Bytes including crlf.

Meanwhile I worked on bigger projects with VisualFORTH, but bugs never have been a problem.

Sometimes I got an error message as shown here, a click on Console and using the Enter key, and it worked again. Sometimes VisualFORTH aborted.



Therefore I recommend to click on"Save all modified" (Form-Menu) to save your work done.

As mentioned above, there is a limitation of 128 Bytes with each part of the “Edit Function” window (each new line adds 2 Bytes for CR-LF).

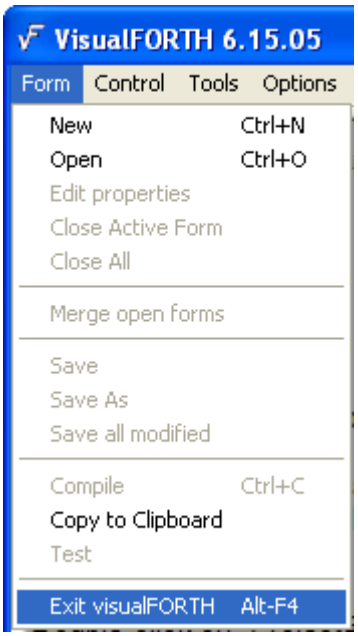
Normally this is enough. If you click on “Apply”, there will be a warning if the number of usable Bytes is exceeded. This limitation normally doesn't matter, the benefit to have everything together in one file, the .ff-file, by far exceeds the disadvantage.

Seldom you need more than 128 Bytes.
My first example-program was a “Calculator” (RPN-Calc.ff) with Reverse Polish Notation – as usual with Forth, and there was enough space for all functions needed, but I extremely modularized all definitions and used names as short as possible.

My biggest projects has been the 4E4th-IDE, an Integrated Development Envirement for TI's LaunchPads and other microprocessors.

3.3 Some Important Hints

To end VisualFORTH, it needs a click on "Exit VisualFORTH" in "Form"-Menu, or – as usual, using <Alt>+F4.



Some time ago my Laptop "froze" (some people say so), a lot of windows were open, and my work was not saved. In reality nothing was frozen, but the mouse didn't work anymore.
The keyboard I could use. That was important.

So I had to have an idea for this situation.

I did know, that I could use the Tab-keys to jump from action to action, but only inside the one active program I used before “freezing”. And to end a program using <Alt>+F4 came to my mind later.

My approach was to check all other Ctrl-keys together with Tab to look if anything helps.
I was lucky, with <Alt>+Tab a little window pops up with the names of all open programs, and it is possible to select one of these – marked with a blue frame – and with every Tab this frame goes to the next open program, as long as the <Alt>-key is pressed.

Even the title of the selected program is shown (really interesting if there are websites open), and with releasing the <Alt>-key, the marked selected program comes to foreground and may be used.

I was really happy that I could save my work and clear the screen to restart.

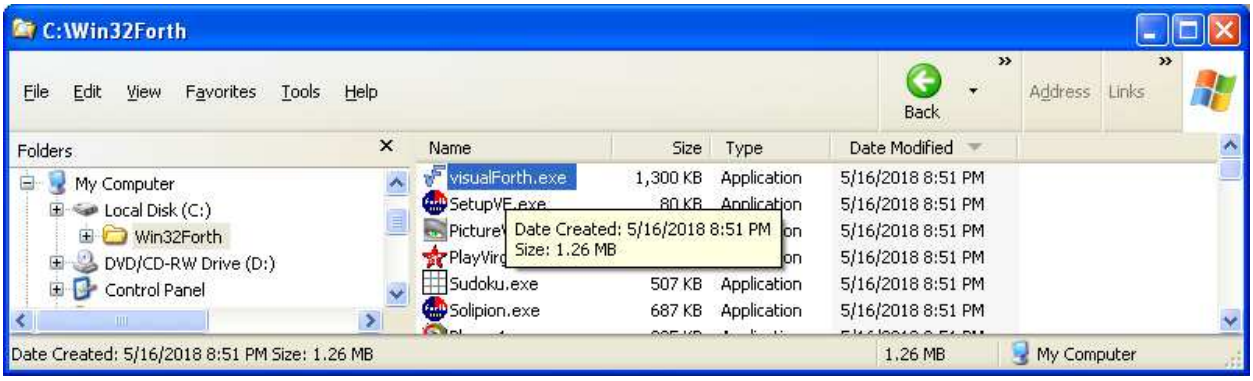
I never could leave VisualFORTH without being reminded to save my work, but there was one exception: in case of an action which aborted VisualFORTH immediately.

That's why it is so important to save your work before starting a special action.

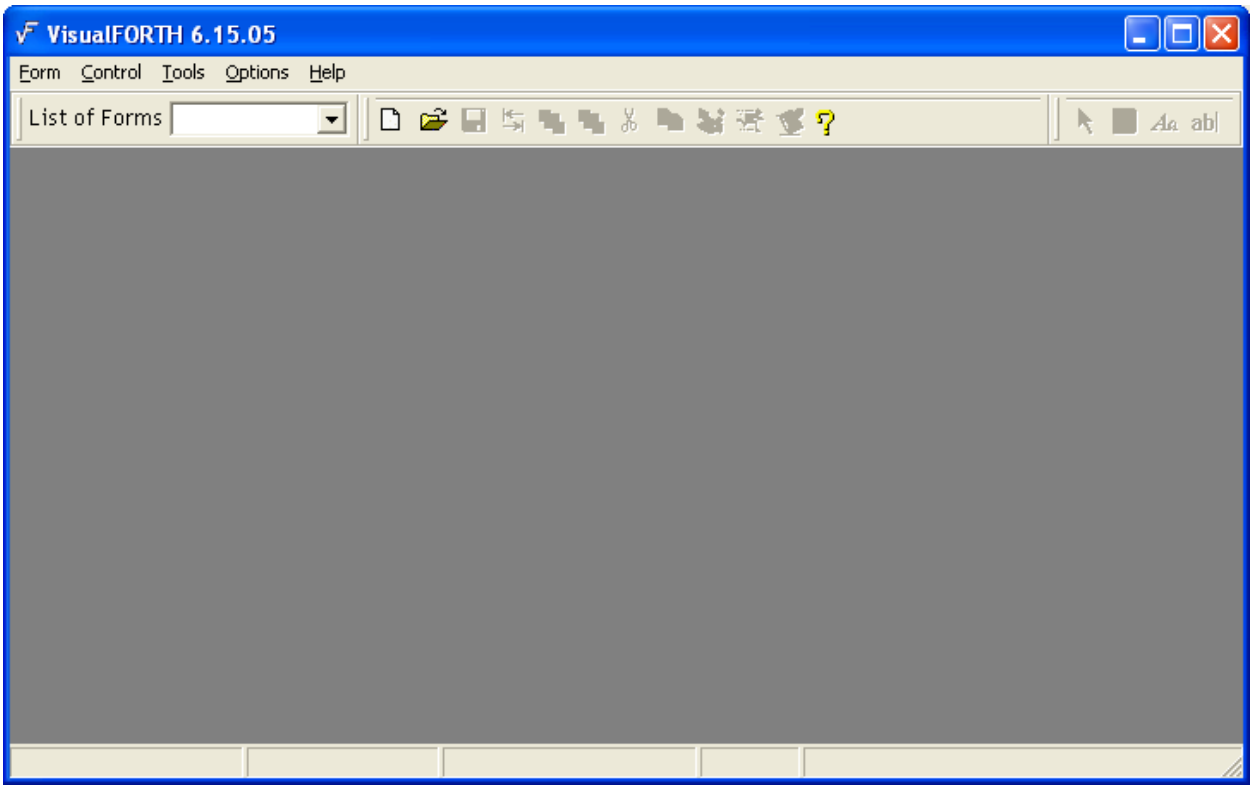
4. A Fresh New Start

The following example shows how to construct a Windows-GUI-Program with VisualFORTH from scratch.

A double click on "Win32Forth" opens this folder and shows its programs and folders:



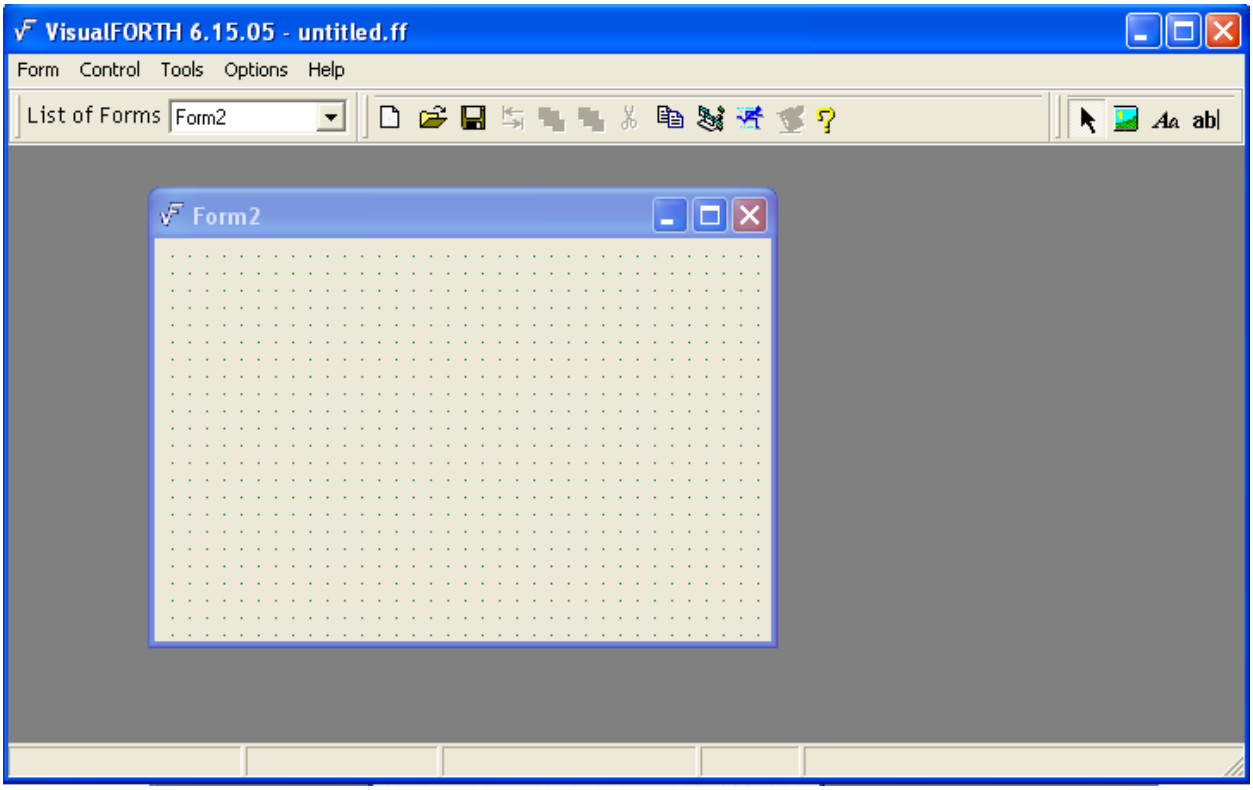
A double click on "visualFORTH.exe" opens VisualFORTH:



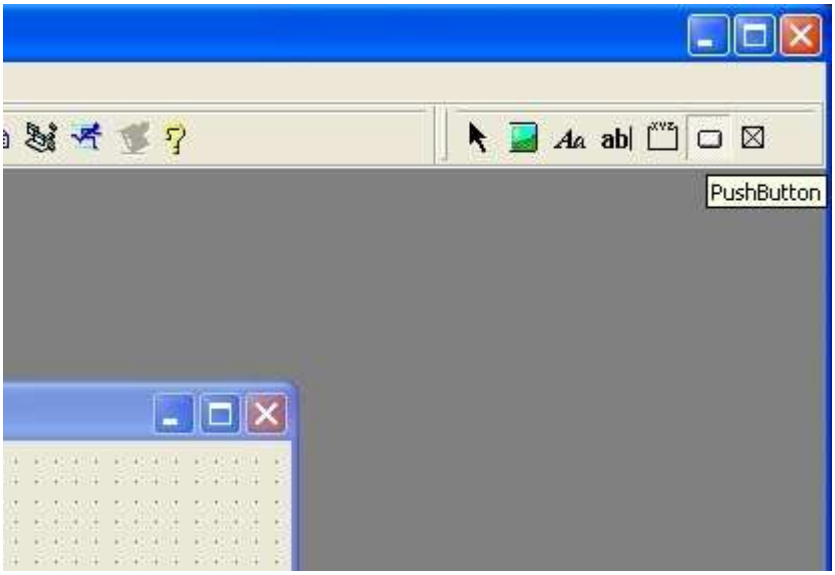
VisualFORTH show up this way after going to “Options”, “Set Preferences” and removing the little hook at “Show Release Notes” before leaving VisualFORTH. But you can go “Help” and click “Close Help Window” for this time.

<Strg>+N, <Strg>+O and other function shortcuts only work if the Help is closed – of course the Pulldown-Menus always work.

With <Strg>+N we open a new Form:



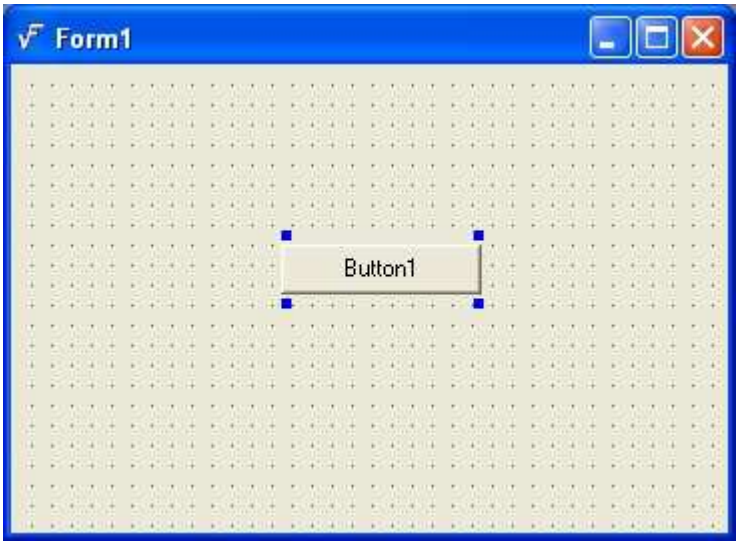
Now we set a PushButton into this Form with a click onto the right Menu-Line Symbol:



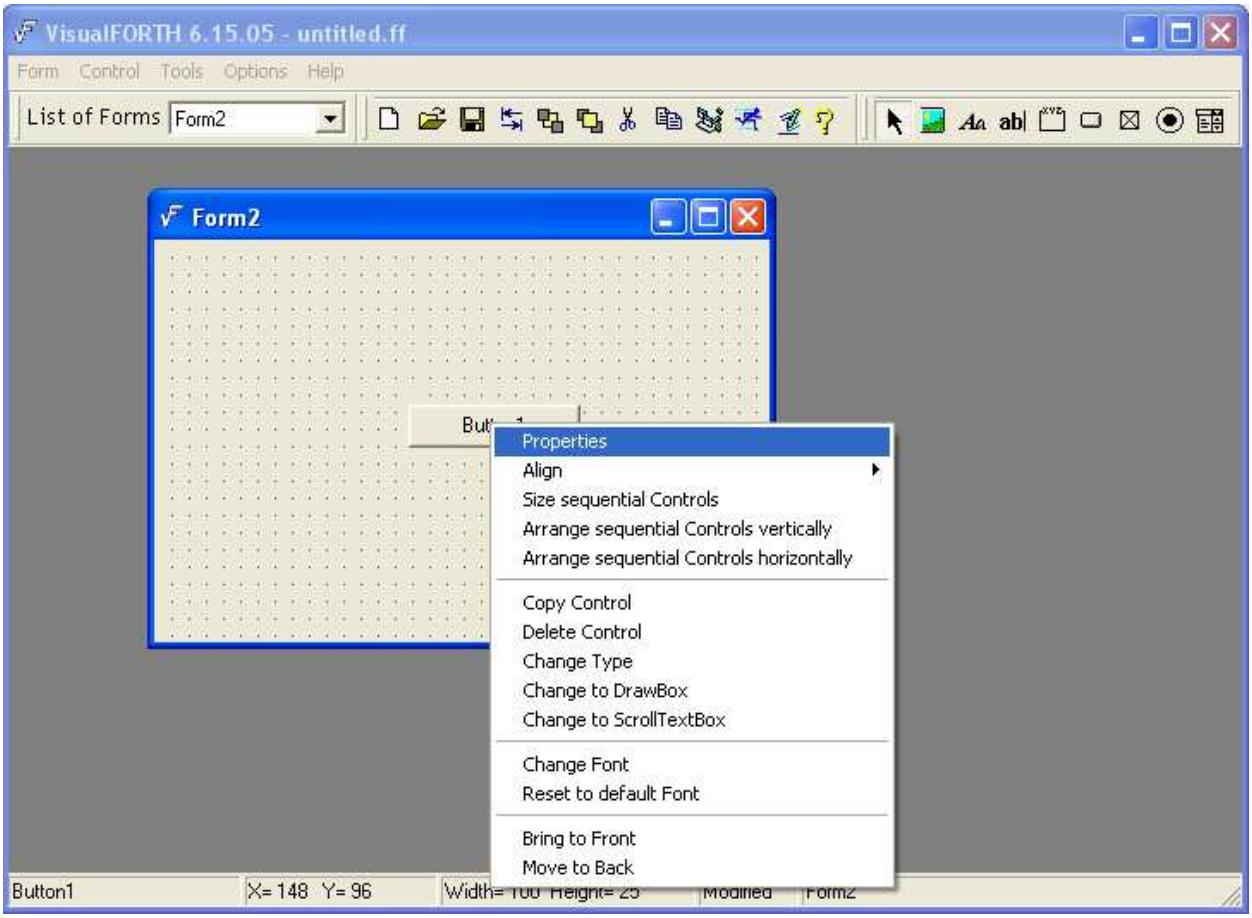
The embossed Pushbutton Symbol shows it is selected (see above), and we release the mouse button - we do **not** have a "Drag-and-Drop"-System here.

Then we move the mouse pointer into the Form to position our button.
The mouse pointer marks the top leftmost corner of the Pushbutton to get.

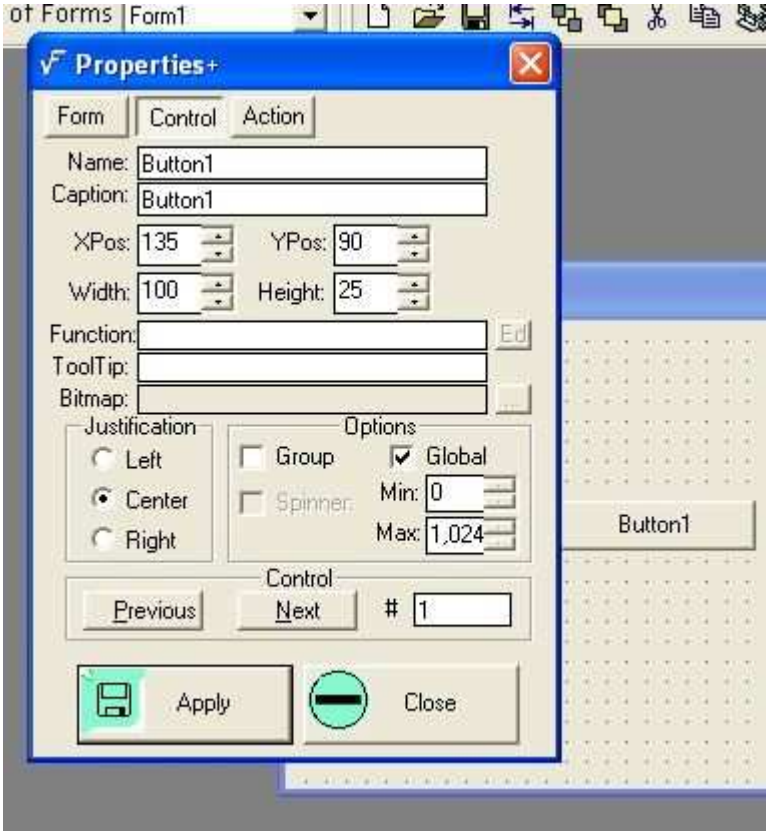
A click with the left mouse key places our first Pushbutton:



Then a **right** mouse click onto the PushButton-Symbol, and our well known "Properties" menu pops up:



A left mouse click on "Properties+", and we get the "Properties+" window:

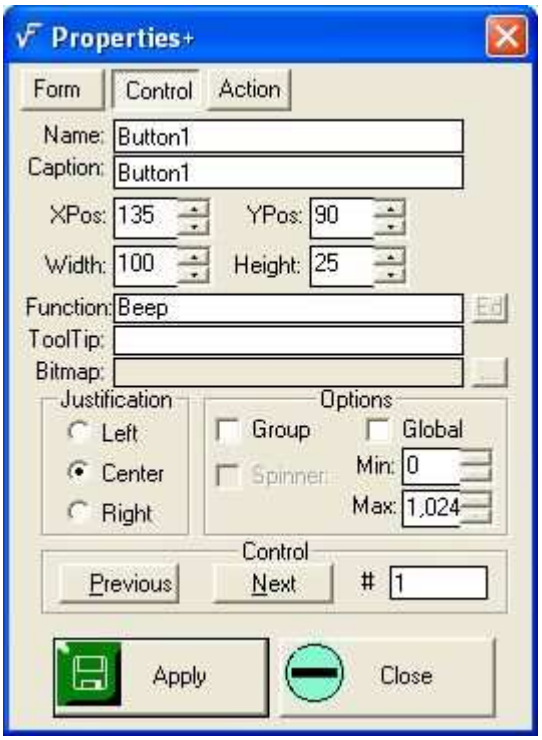


The "Properties+" window shoes position and dimensions of the PushButton we just posted and these may be changed.

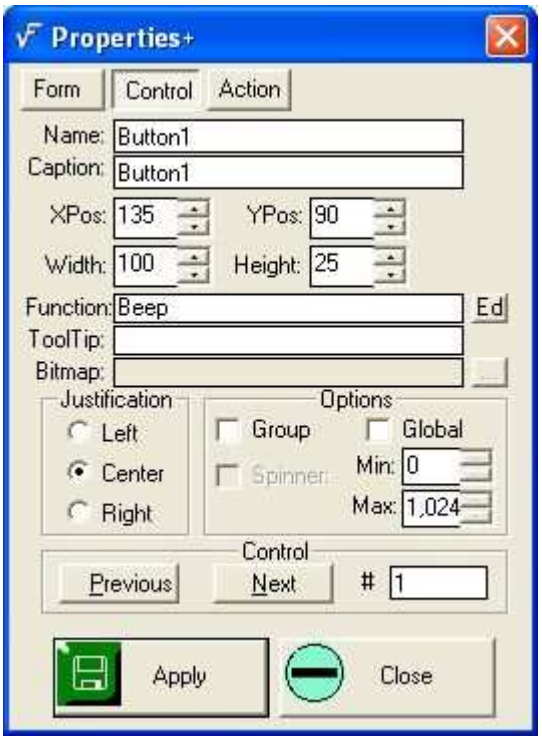
To change the position or dimensions, we type the new number into the corresponding field (or use the spinner control), and with a click on “Apply” the systems takes these values and changes the PushButton accordingly.

Now we can type our famous "Beep" for testing purposes:

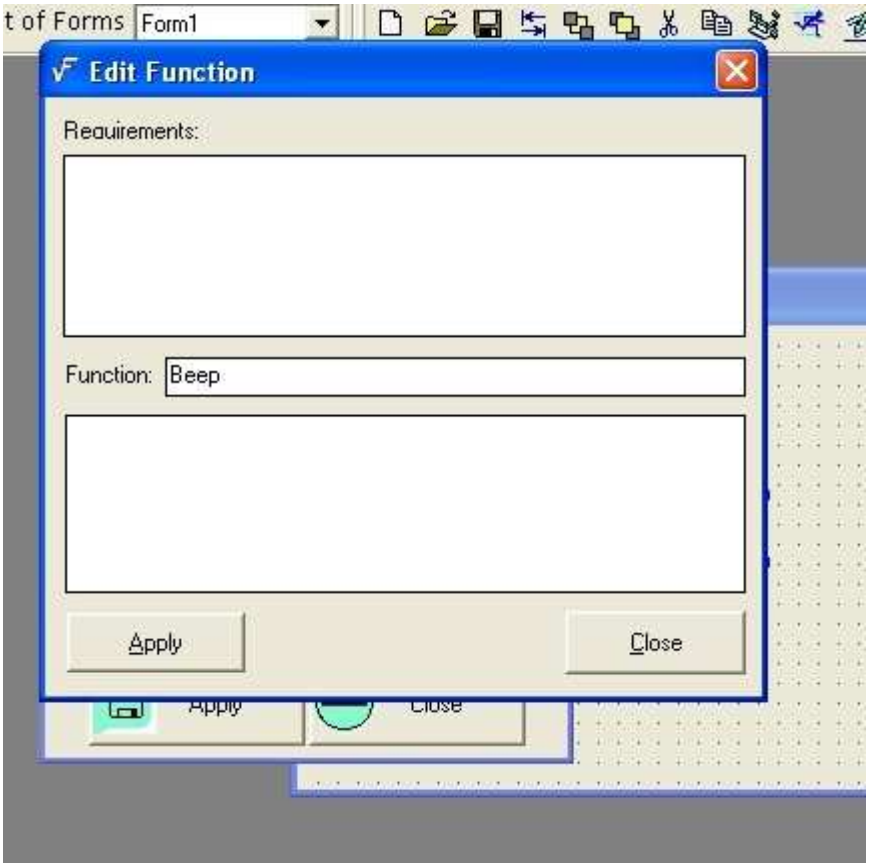
To make it work there has to be a little hook at "Global"!



With a click on "Apply" (see above) the function is taken over, you recognize this with the little "Ed" button enabled:

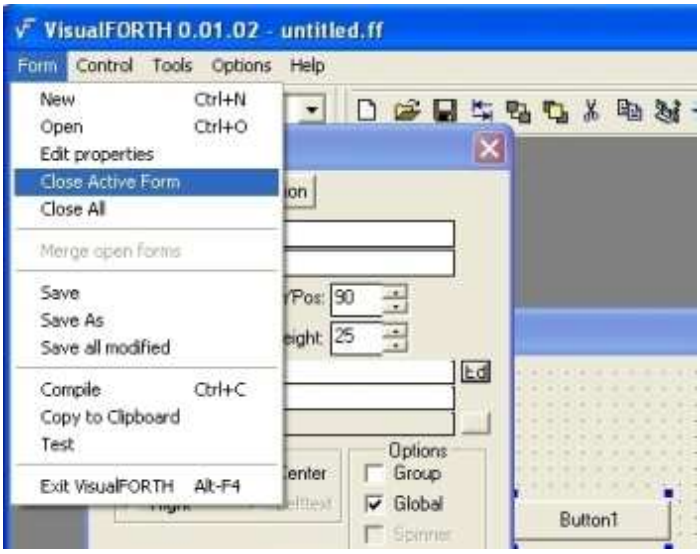


A click on "Ed" opens the “Edit Function” window again:

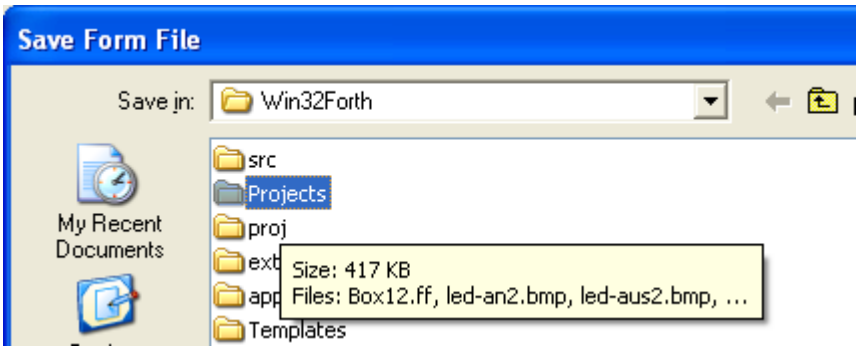


I guess you know meanwhile how to proceed from here.
In case you have forgotten, reread paragraph 3.2

When clicking on "Close active Form" in the "Form"-Menu we get a reminder to save our new Design.



Don't forget to use the folder "Projects" for saving, this is the only way to later get an One-Click-EXE-File!

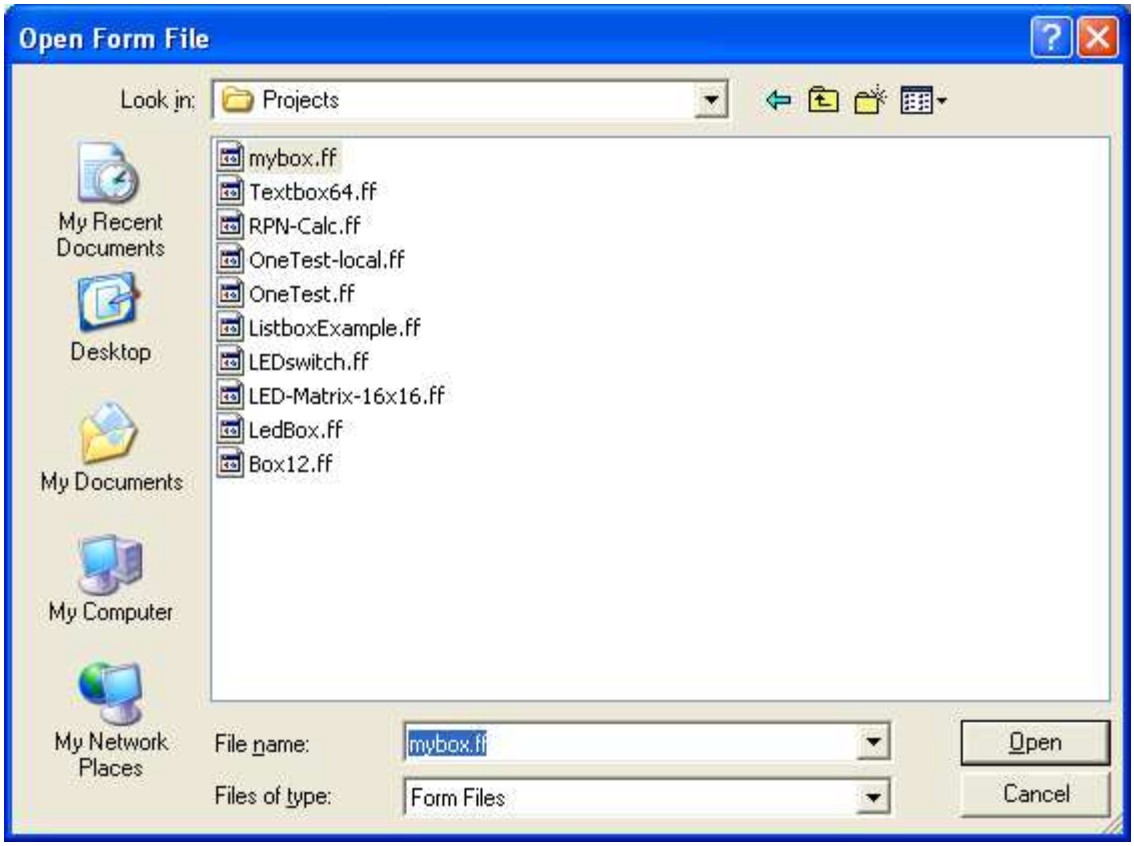


A doubleclick on "Projects" opens this folder and we save our project “mybox”:

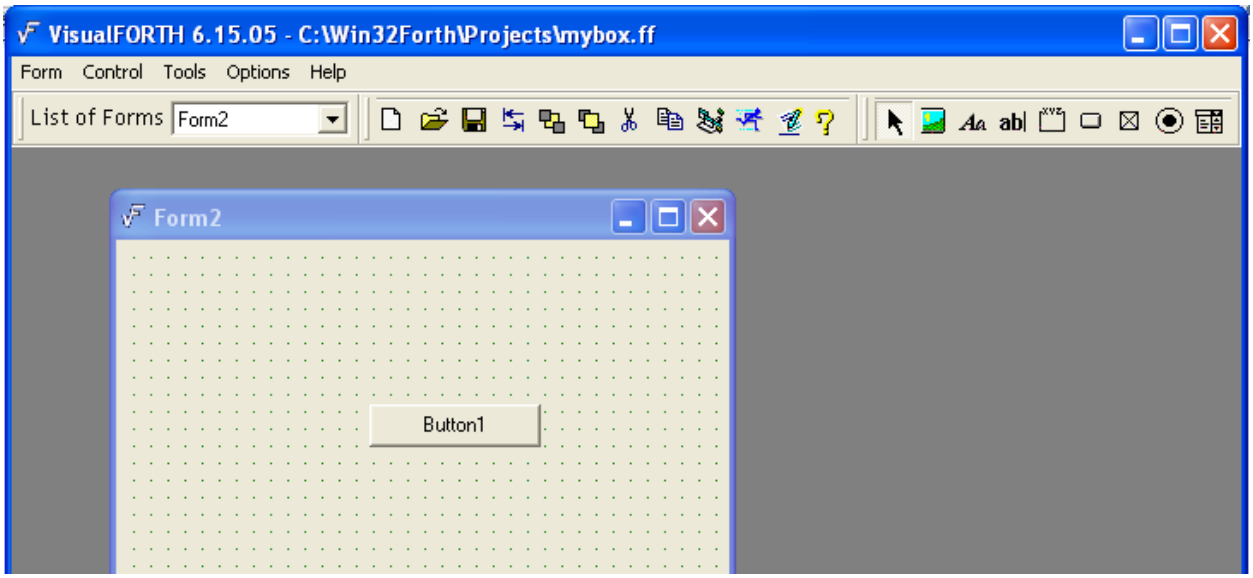
5. One-Click EXE-File

Now let us look how to automatically generate EXE-Files.

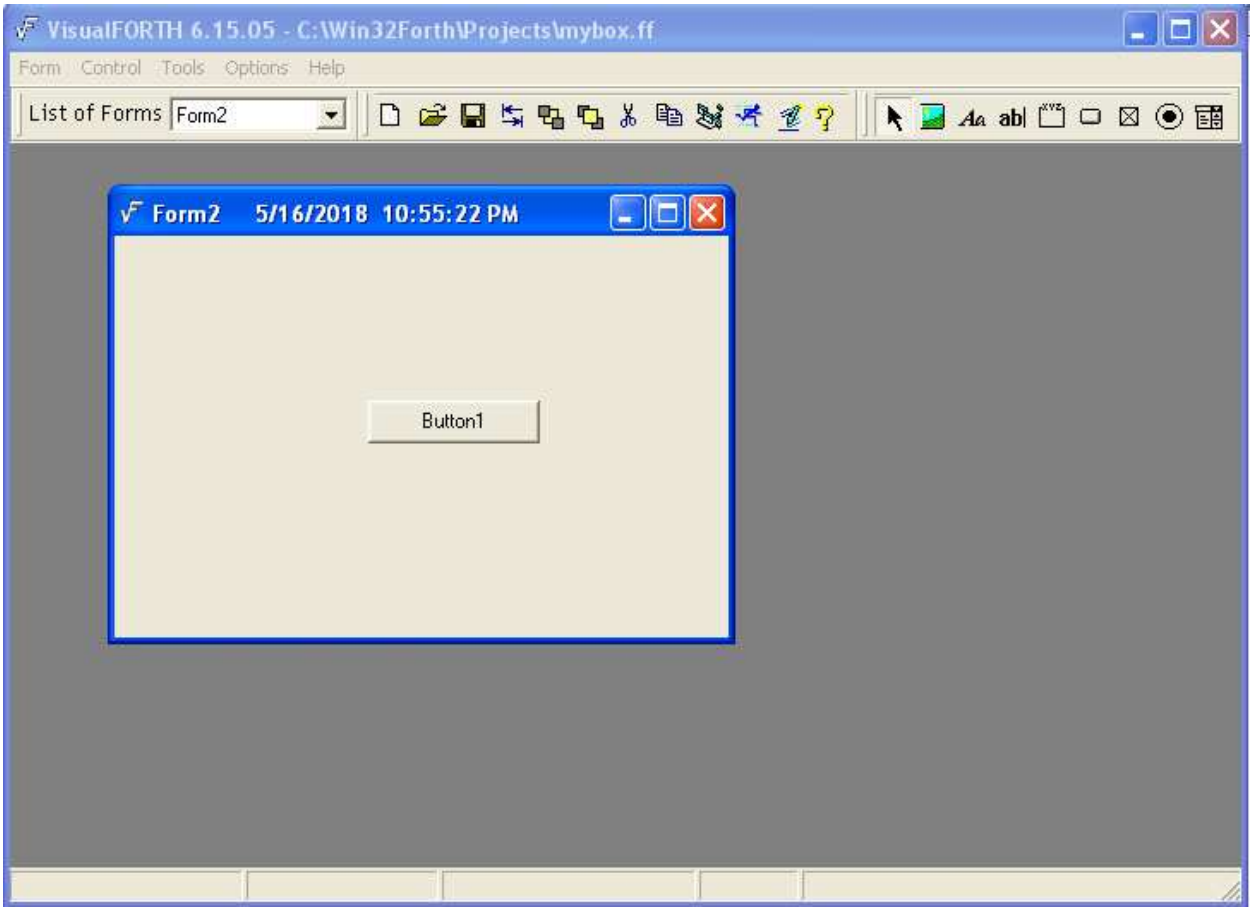
In VisualFORTH we open with <Strg>+O our file. "Open Form File" shows up:



A click on "mybox.ff" inside the window and a click on "Open", and there is our little project:



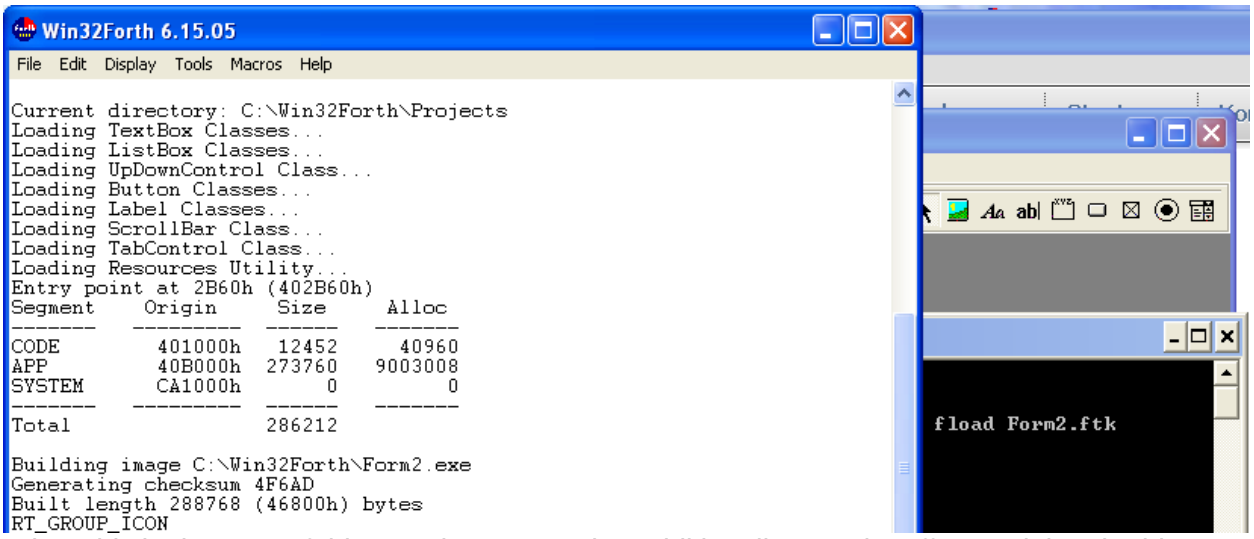
To test, click on “Run” (the little blue runner):



A click on "Button1", and we hear a Beep, as we did last time.

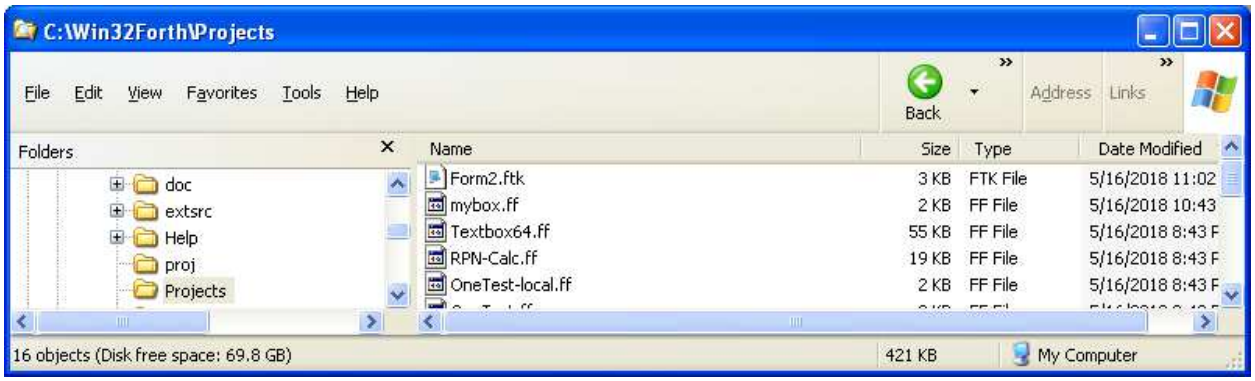
To get an Exe-File, click the Symbol between "Test" und "Help":

Shortly you may see two windows building the EXE:



When this is done, our folder "Projects" contains additionally to mybox.ff, containing the binary

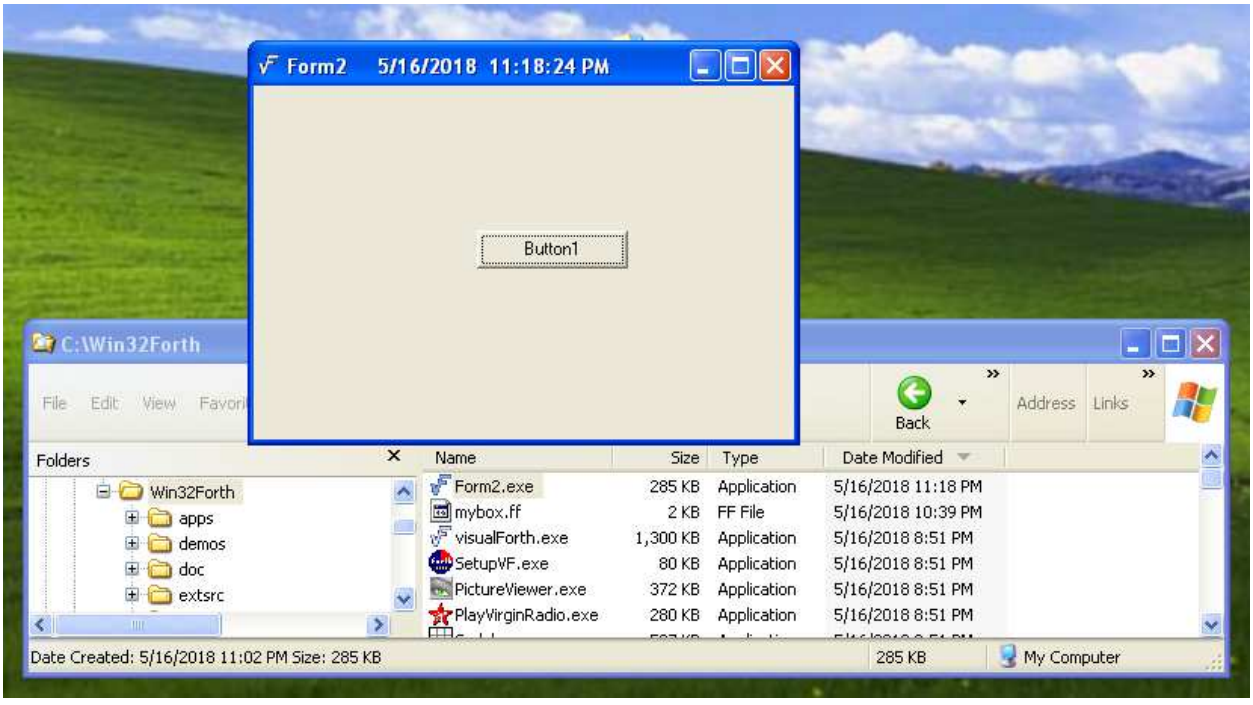
data of our little project, a new File, Form2.ftk:



Form2.ftk is an automatically generated Forth-File with Turnkey-additions needed to get a Turnkey-Program.

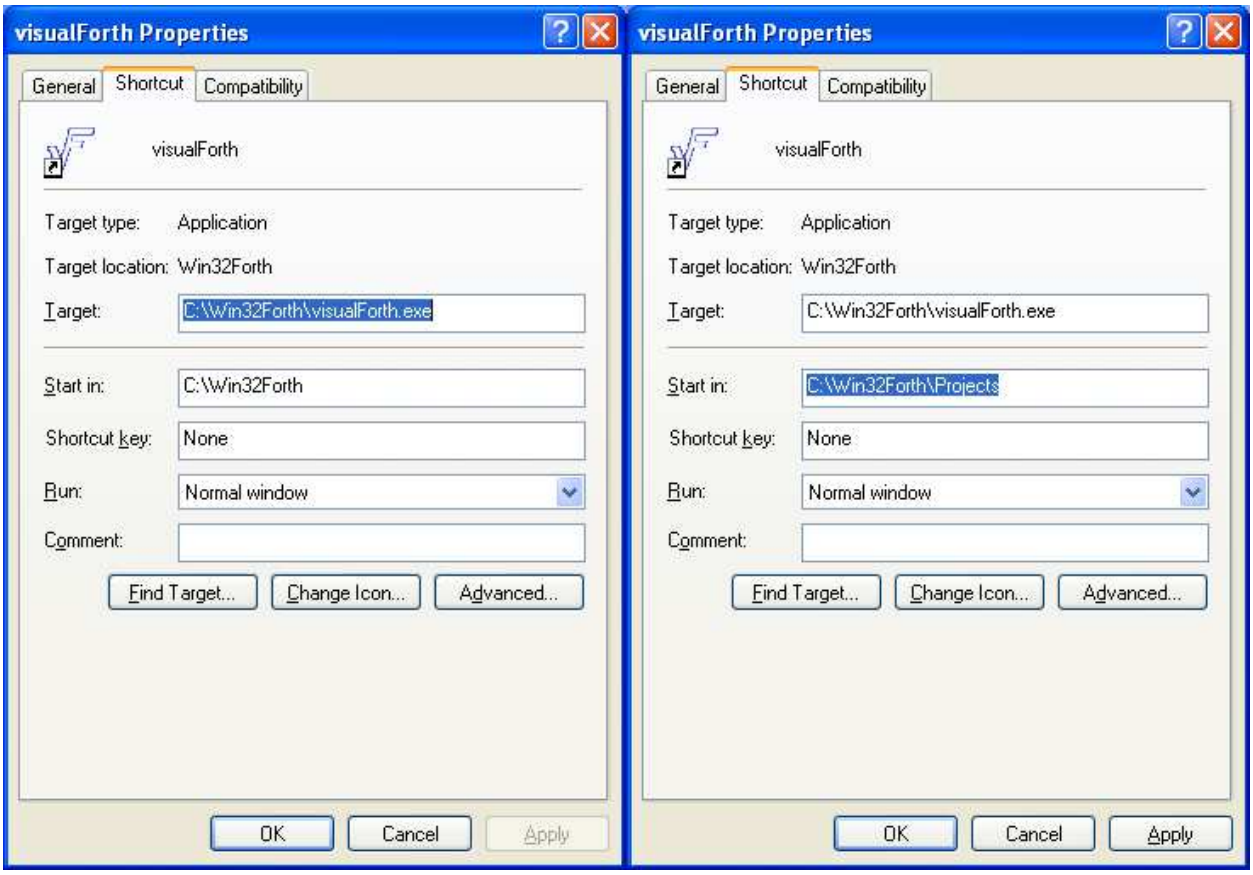
You can use Form2.ftk with Win32Forth, use fload Form2.ftk, and Win32Forth generates an EXE-File. This may be helpful in case of problems for debugging.

Now you may close VisualFORTH, but you don't need to.
A click on Form2.exe in our Win32Forth folder, which you will find on top, will start this file, and as before, a click on Button1, and we get our Beep.



6. Adjustments

It is a good idea to get VisualFORTH always started with the "Projects" folder.
Below as it was first, on the right side with „Start in“ Projects added.



A click on “Apply” saves this new adjustment.

This may not work with Windows 10.

Have Fun with Testing!

As we all know, the Forth programming language bears immense possibilities.

Now we have VisualFORTH, and with VisualFORTH this potential of Forth will be unleashed! There will be additions, supplements and extensions which may be used much easier now with VisualFORTH.

For help and to communicate your experience with VisualFORTH you may contact:

http://de.groups.yahoo.com/group/visualFORTH/ or email me at Dirk@4e4th.eu

If there are any problems or questions, contact this Yahoo-Group, this will help to communicate solutions.

June 2009 – May 2018
Dirk Bruehl