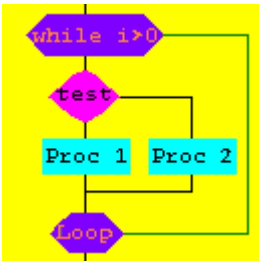


OmegaChart

Ver. 6.0

Digest Source Codes Painlessly

Table of Contents

1. Software Installation

1 System Requirements

This software package runs on Microsoft XP platform with .NET framework. The generated flowchart can be exported to Microsoft Excel, Microsoft Internet Explorer or Microsoft Visio. Microsoft Excel, Microsoft Internet Explorer and Microsoft Visio packages must be installed prior to the installation of OmegaChart. Also, Microsoft .NET framework redistribution package should be downloaded before installation.

2 Installation Steps

a) Create a temporary directory, e.g. c:\temp\

b) (Window Vista User please skip this step) Download "Microsoft .NET Framework 3.5 Service Pack 1" from Microsoft download center.

http://www.microsoft.com/downloads/details.aspx?FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7&displaylang=en

Save the executable file "dotnetfx.exe" in the temporary directory created in step a)

Due to its huge size, OmegaChart installation package does not include this file.

c) Download from the site http://www.omegachart.com/index.htm the OC60 installation package and save it in the previously created temporary directory. Uncompress the zip file into temporary directory.

Now this temporary directory should contain the following file and directory:

Directory:

Examples

Files:

setup.exe - setup utility

pp.dat

default.cfg
default_small.cfg
print.cfg
print_small.cfg
} - configuration files

run.log
OC_wall.jpg
Title.js
AltName.htm
} -files essential for the execution of OmegaChart

dotnetfx.exe
SVGView.exe
MathPlayerSetup.exe
} -Other Software Environments

Oc60.pdf - User manual

Interop.AcroPDFLib.dll } - Dynamic Link Library

AxInterop.AcroPDFLib.dll

Oc60.vsd - MS-Visio viewer
Oc60.xls - MS-Excel viewer

d) Run the setup.exe program to define in the setup menu the launch directory (for example "C:\OC60_launch\"). Check all the checkbox if this is the first time installation. Uncheck some of the checkbox to skip installation of the corresponding component which has been in the system already.

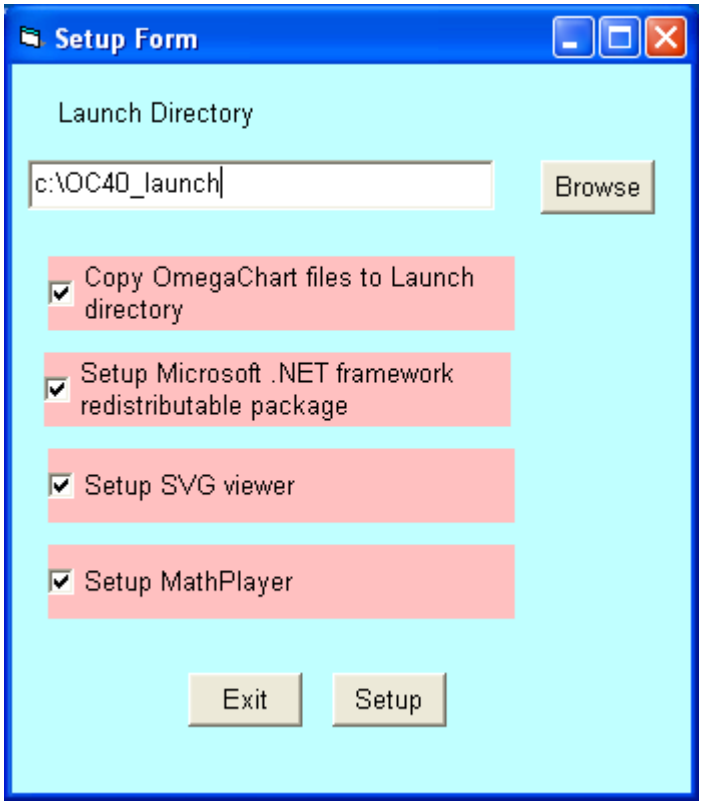


Fig. 1 OmegaChart setup menu

Setup program will notify you when the installation is completed. You must restart the computer before the installation takes effect.

e) Copy the “Example” directory to launch directory and delete the temporary directory created in step a)

2. Software Activation

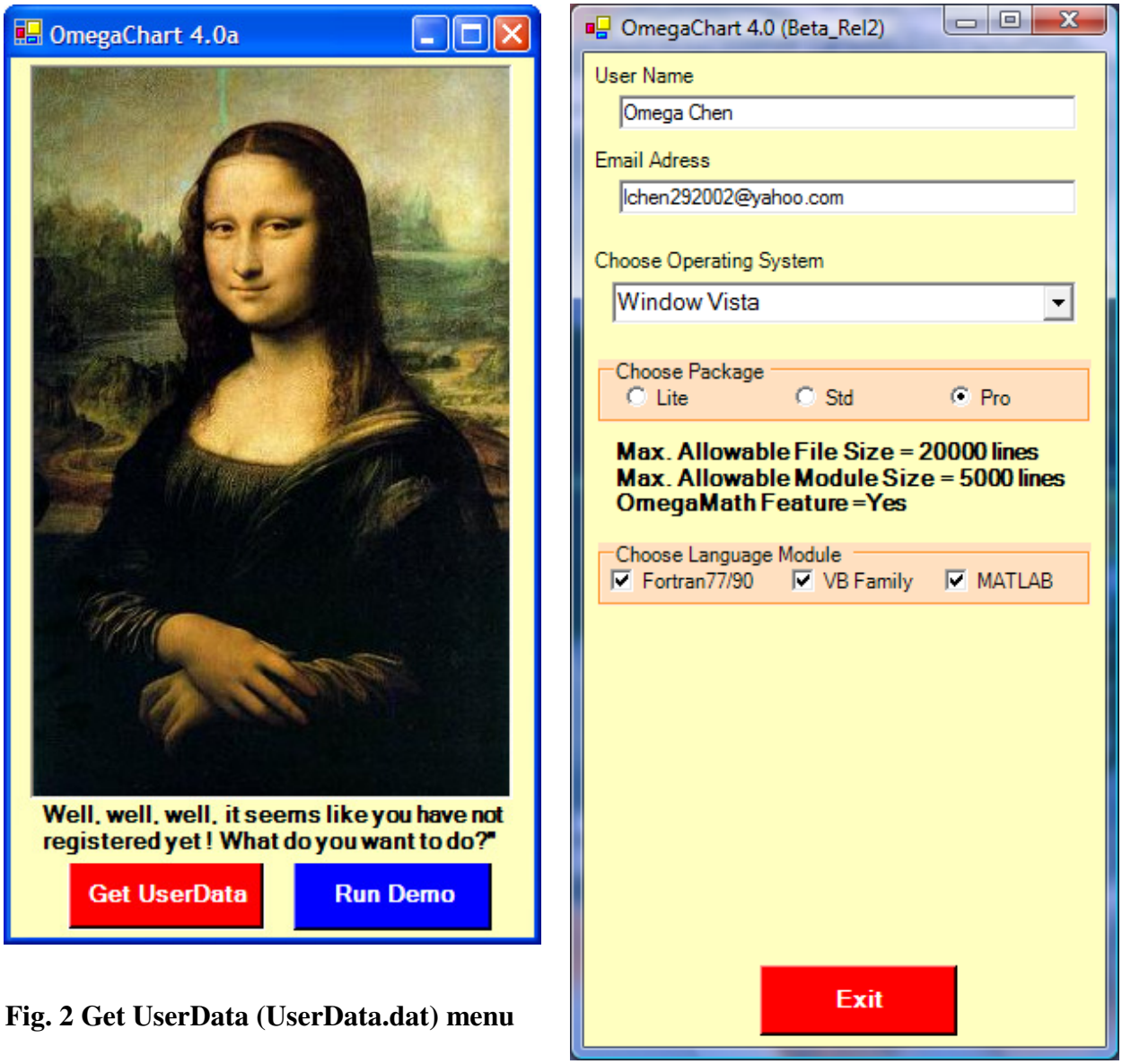


Fig. 2 Get UserData (UserData.dat) menu

When you first started OmegaChart, you will see the "Mona Lisa" screen (Fig. 2) asking about what to do next. You may run OmegaChart in demo mode to taste its capability or generate UserData.dat file for registration. Click on "Get UserData" button. Upon filled all the necessary information in the next menu, click "Generate UserData.dat". Userdata.dat is saved in the launch directory. Email this file to services@omegachart.com for registration. The activation license file (RegKey.dat) will be e-mail back to your address. Save the RegKey.dat file in the launch directory. If you re-install or reload OmegaChart program you will need new license file.

3. Uninstalling the OmegaChart

To uninstall the demo version of OmegaChart program, simply delete the launch directory. OmegaChart does not place any "mark" into Windows XP registers and therefore it is sufficient to delete ONLY the OmegaChart launch directory (Green Installation)

To remove Microsoft .NET framework redistributable package, select from Start Menu Control Panel and there Add and Remove Programs.

4. Running Analysis

After software activation, the OmegaChart master dialogue form (Fig. 3) will appear by launching the program.

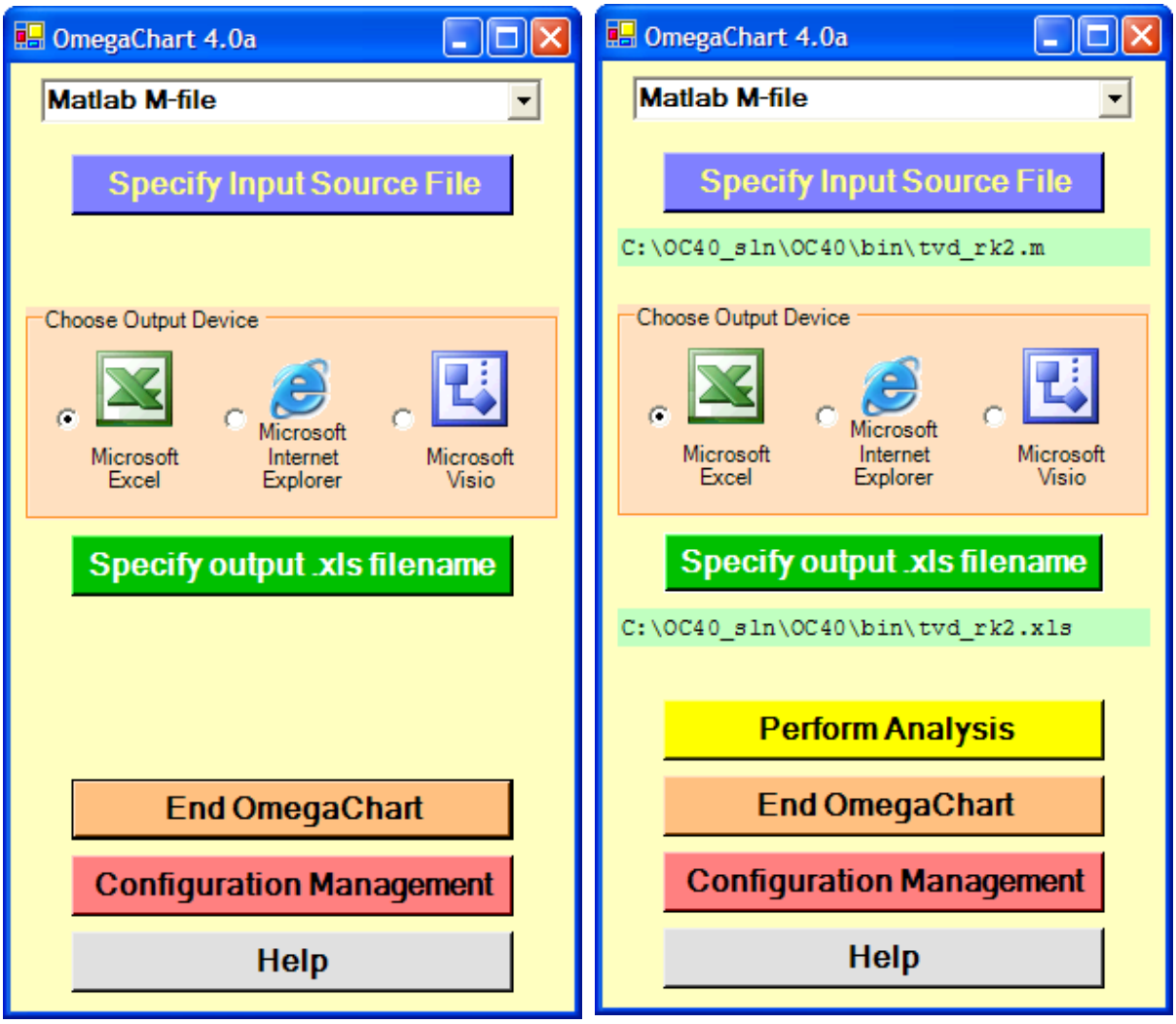


Fig. 3 Master Dialogue Form

To generate flowchart for source code, please follow the steps described below (working from top to bottom in the master dialogue form)

1. From the dropdown menu , specifies the language/form of the source code
   - Fixed form (Fortran77 and Fortarn90)
   - Free form (Fortran90)
   - Visual Basic Family
   - Matlab Scripting Language
2. Specifies the file to be analyzed.
3. Choose “output device” among the three: MS-Excel, MS-Internet Explorer or MS Visio
4. Specifies path/filename where the generated flowchart file should be saved. Depending on the current “Output Device” selection, either “Specify Output Excel File”, “Specify Output Directory” or “”Specify Output DAT file (for Visio)” will be displayed.

5. Click the “Perform Analysis” button, the package then provides an estimated executing time and start the analysis. The generated flowchart will be automatically stored in the “output device” you specified. The message “Analysis Completed” will appear if flowcharts are successfully created.

Notes:

a) In some platforms, the generated Excel file may not be readily viewable by simply clicking on the file name from window explorer. You might need to start Excel first and then open the generated .xls file.

b) OmegaChart analysis is time extensive. When the analysis is finished, Excel will dismiss automatically or will ask if an existing file should be replaced.

c) The generated .DAT file requires further post-processing by “reader”. OmegaChart Ver 4.0 package provide two “reader”: one for Microsoft Visio (oc60.vsd) and the other for Microsoft Excel (oc60.xls). After launching the reader (with macro enabled), click on the OmegaChart logo and specifying the .DAT file name, the flowchart will then be displayed in the reader.

d) The “Reader” for Microsoft Visio may require an additional input: “Do you want to glue lines to shape? “. If the answer is “yes”, then the resulting flowchart “glue” the connecting lines to the shape objects. In this “Glue Mode”, if, for any reason, one of the objects in the flowchart is displaced, the neighboring objects connecting to this object are also affected. (Fig 4)

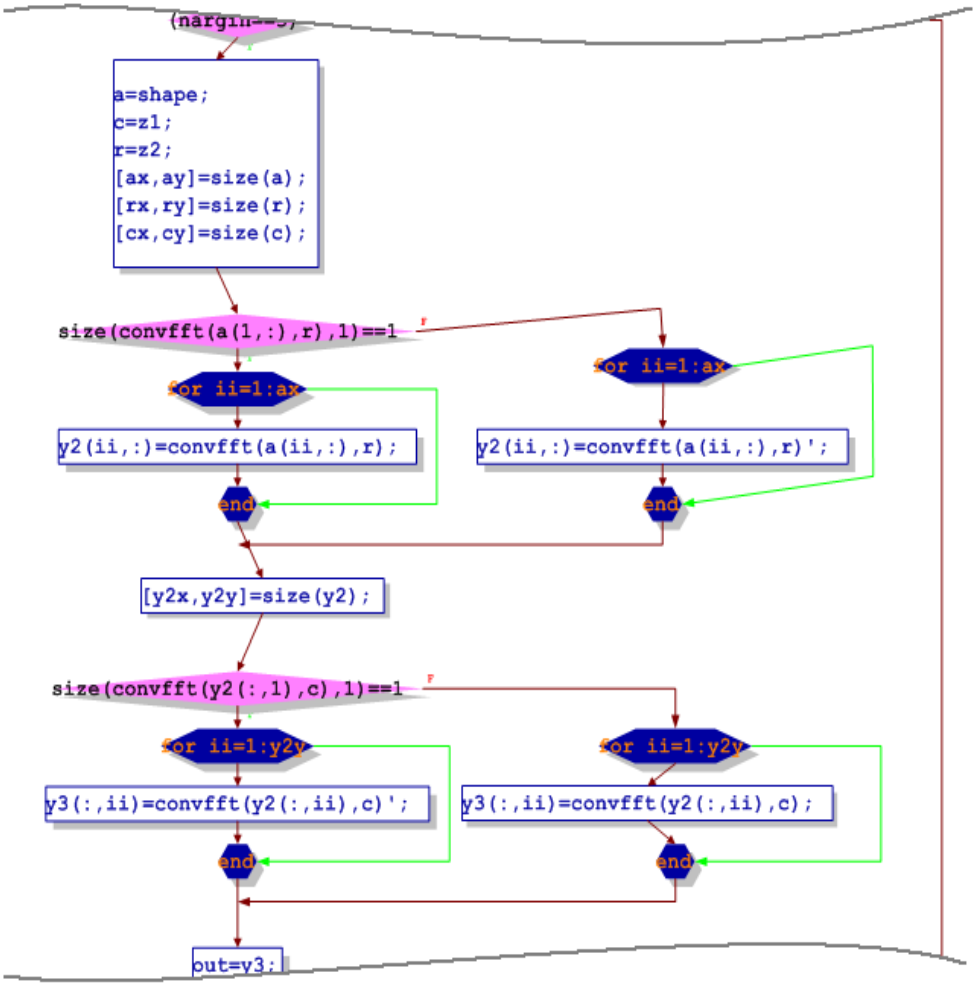


Fig. 4 An example of displaced object in

On he other hand, in the “glueless mode”, all objects are independently positioned. Moving any object around will not influence any other objects (Fig. 5)

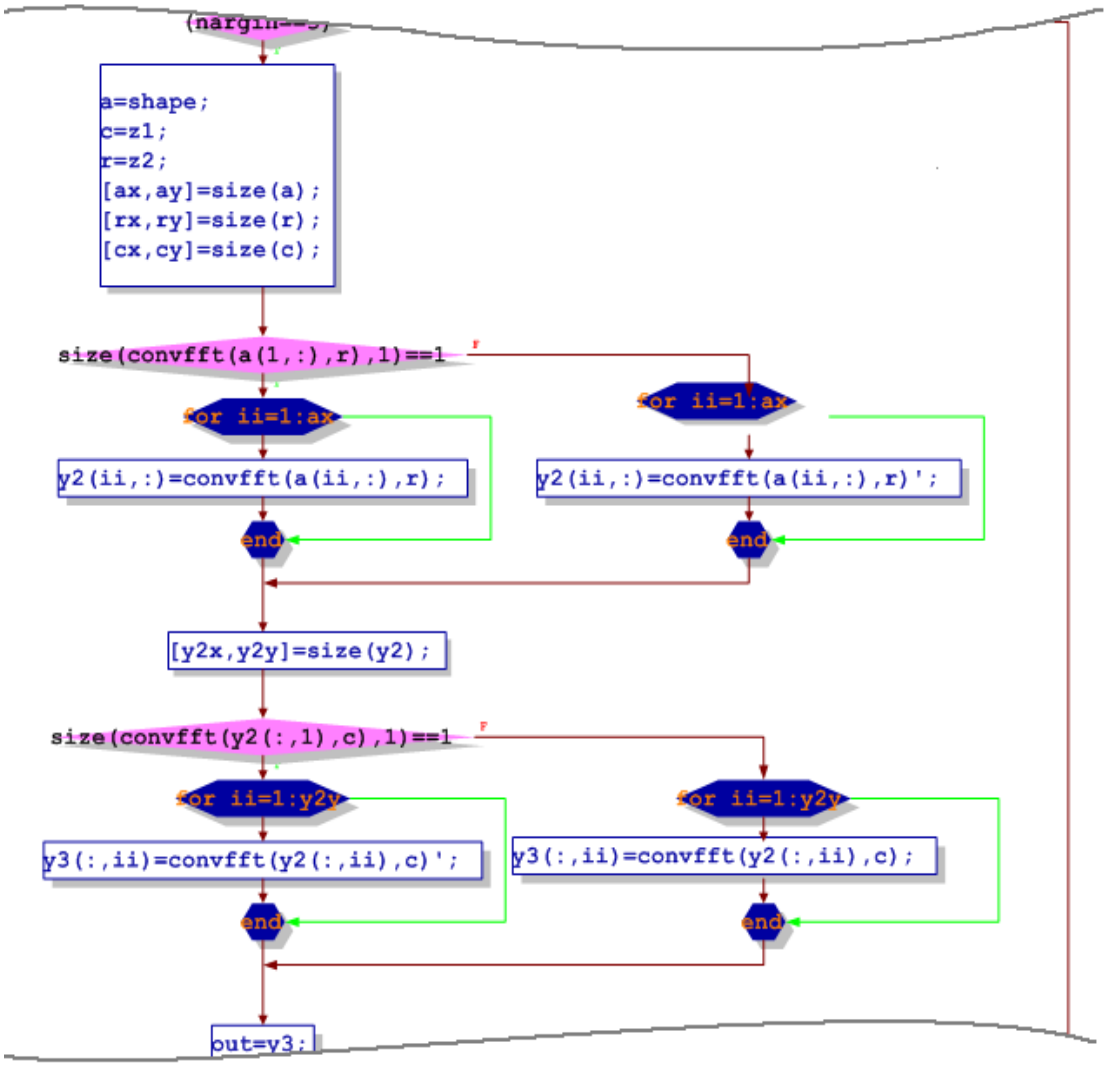


Fig. 5 Flowchart in “glueless mode”

This “gluing” property requires some extra post-processing time. For complex flowcharts generated from large source module, this additional effort may be very time consuming and should be use only if it is absolutely necessary.

The “Reader” for Excel (oc60.xls) does not support the “gluing” property.

e) The Excel and Visio handle macro in slightly different ways. In Excel, if the macro generates a new ‘worksheet’, then this new worksheet is displayed immediately. Therefore, user may be able to see the generated worksheets while the macro is running. In contrary, Visio display newly generated ‘pages’ only when the macro is completed. Therefore, when generating large flowcharts using Visio, all pages are display at once only after the analysis was completed. The user should be aware of this difference when selecting the output device.

5. Input Source Files

5.1 FORTRAN

The source files (both Fortran77 and Fortran90) must be the subset of the Fortan Language Standard [add Reference]. Any machine specific language features, which are not part of the Fortran Language Standard, are not permitted, and OmegaChart will stop processing when such command line is found in the analyzed source file.

5.1.1 Flattening of program Module

Processing of the Input source files by OmegaChart program:

1. Simple Case – One Level Input File

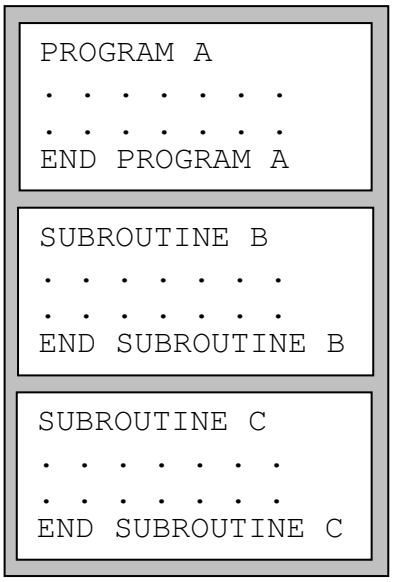


Fig. 6 One Level Input File

One Level Input File consists of one main program unit and several independent subroutines or functions (Fig. 6) . OmegaChart will process and analyze each routine, including the main program as an independent unit.

2. Advanced Case – Two Level Input File

Multiple level Input File is composed of several independent program units where at least one of them have internal program units as shown in Fig. 7. In Fig. 7a, internal subroutine “A1” is renamed to be “A~A1”, internal subroutine “B1” is renamed to be “B~B1”, and internal subroutine “B2” is renamed to be “B~B2”. All five routines (A, A~A1, B, B~B1, B~B2 and C) are analyzed as if they were all external routines.

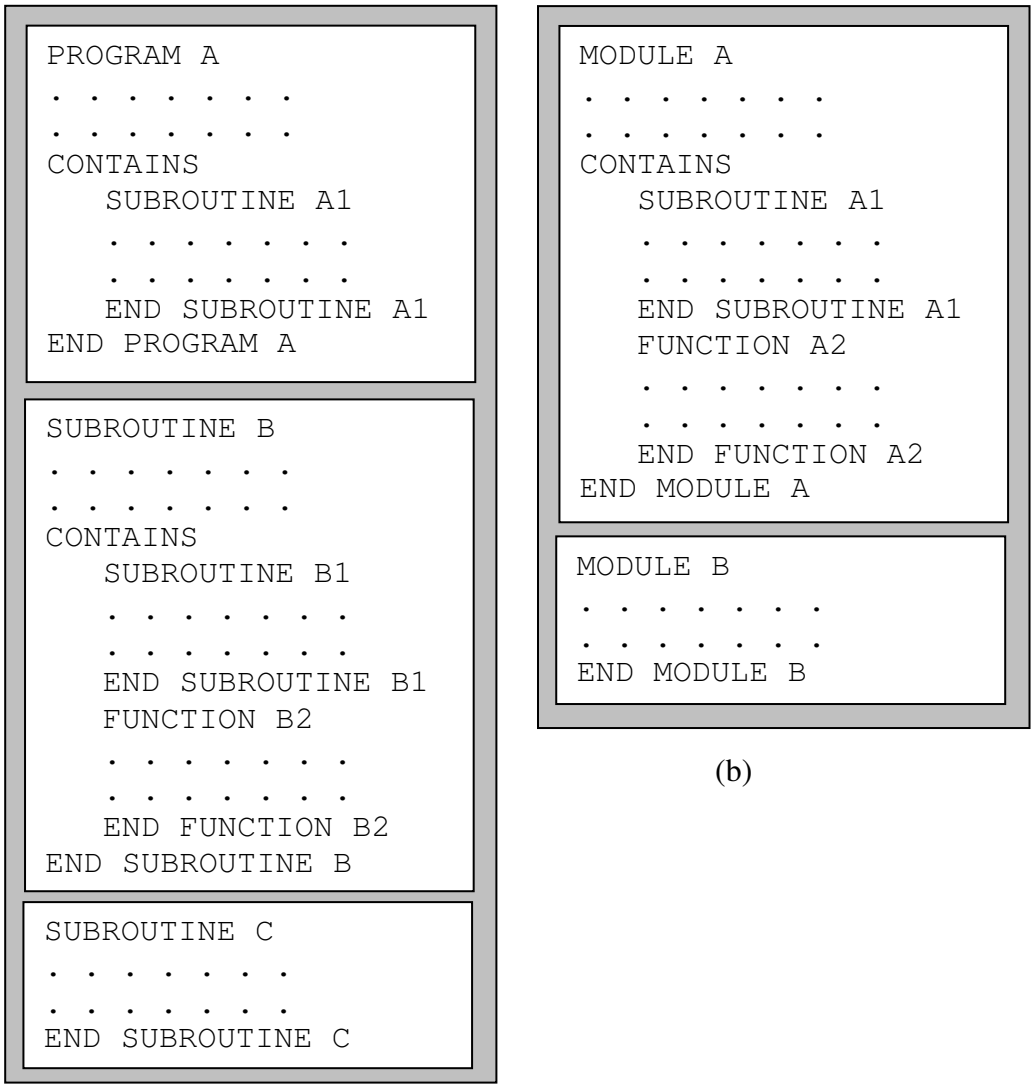


Fig. 7. Two Level Input File

Internal routines (A1, B1 and B2) would result in flowchart "title line" as

SUBROUTINE A~A1 (……)

while the "ending line" will remain the same

END SUBROUTINE A1

That is to say, only the "title line" has been renamed, although they still referring to the same subroutine. External routines (such as subroutine A, B and C) are not renamed, therefore the flowchart will indicate:

SUBROUTINE A (……)
……
END SUBROUTINE A

Similarly, for Fig. 7b, internal subroutine A1 and A2 will be renamed to A~A1 and A~A2 respectively, while module A and external subroutine B are not renamed. The resulting flowchart will have a form:

```
MODULE A (……)
……
END MODULE A

MODULE A~A1 (……)
……
END MODULE A1

MODULE A~A2 (……)
……
END MODULE A2

MODULE B (……)
……
END MODULE B
```

Note: The name of the subroutine is used not only in the presentation of the flowcharts, but also as the tab name in the Excel sheet.

Due to the fact that the new name of the internal subroutine is actually a combination of the name of its parent subroutine and the name of the internal routine itself, chances are they may be quite lengthy. For example, if the name of the "parent" subroutine is

InterpolatePsychrometricChart

while the internal function has a name

RelativeHumidity

Then the new name becomes

InterpolatePsychrometricChart~RelativeHumidity

Microsoft Excel places a limitation on the length of tab name characters (31 maximum). To avoid error due to an over lengthy tab name, only the rightmost 31 characters are use in the tab name. Therefore the tab name will be

hrometricChart~RelativeHumidity

The name in the "title line" in the flowchart remains in its full length. However, there is another limitation discussed in the Section 5 of this manual (PRINTFORM and LENLIMIT).

3. Advanced Case – Three Level Input File (with internal program units)

In this case, the input file consists of several independent "modules" in which at least one has internal "program unit", which in turn contains yet another level of internal program units, as illustrated in Fig. 8.

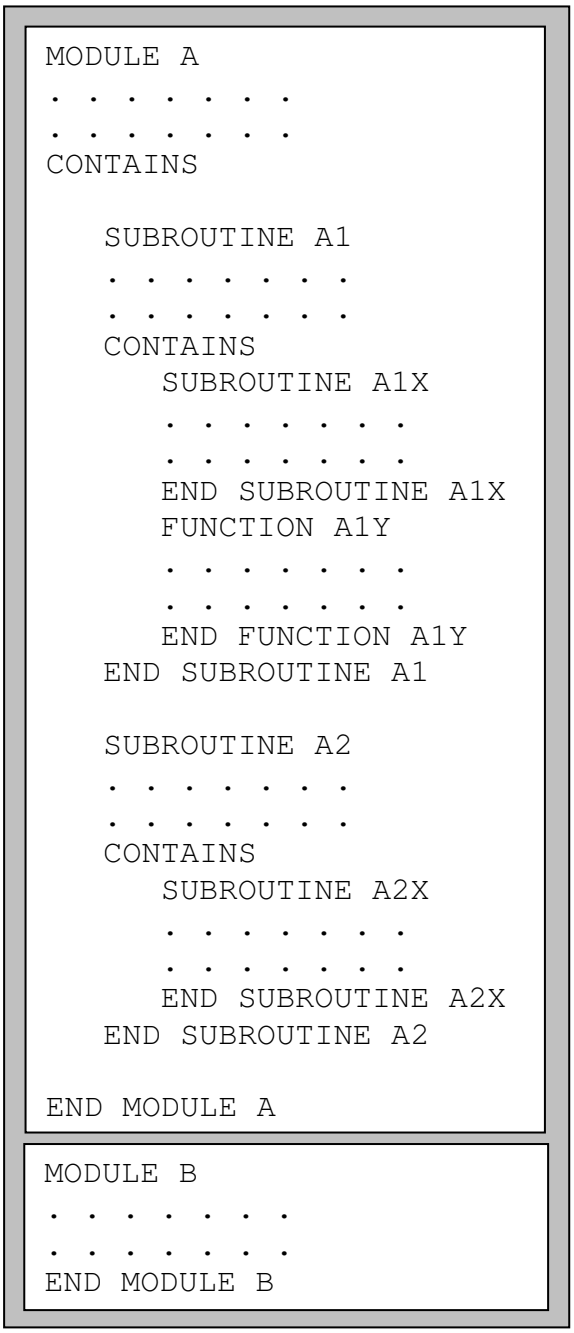


Fig. 8 Three Level Input File (with internal program units)

In this example, internal subroutine “A1” is renamed to “A~A1”, and the nested internal subroutine “A1X” is renamed to “A~A1~A1X”. Other nested internal subroutine “A1Y” is renamed to “A~A1~A1Y”, and internal subroutine “A2” is renamed to “A~A2”. The nested internal subroutine “A2X” is renamed to “A~A2~A2X”, and all eight program units (external or internal; A, A~A1, A~A1~A1X, A~A1~A1Y, A~A2, A~A2~A2X and B) are analyzed as if they were all external routines. The display convention is the same as those described earlier.

5.1.2 The language features not supported by OmegaChart

OmegaChart Version 4.0 and higher does not support the following features:

1. For Fortran77 and Fortran90 fixed form, the space between characters is not important. The statement

```
D O 10 I=1,LOOPEND
```

is the same as:

```
DO 10 I=1,L O O P E N D
```

Neither statement is supported by OmegaChart. The proper statement should have a form:

```
DO 10 I=1, LOOPEND or
DO 10 I = 1,LOOPEND
```

OmegaChart does not accept spaces in Fortran keywords (as in " D O") as well as spaces in variables (as in "L O O P E N D"). However, spaces between keywords, variables, operators etc. are accepted.

2. OmegaChart cannot analyze source code, which is a mix of fixed form and free form. The only one type of the form is accepted.

3. The older Fortran programs may contain the "conditional GOTO" inside of I/O statements. For example:

```
READ (1,ERR=99, END=999) X
........
999 CONTINUE
........
99 CONTINUE
........
```

Although OmegaChart supports this feature, it will not place a linking line between the I/O statement and the target (labeled) statement.

4. The CYCLE or EXIT a LOOP in Fortran90 are processed as following:
If a label is given to the CYCLE or EXIT statement, then a linking line between this CYCLE (or EXIT) statement and the labeled DO (or END DO) statement is drawn.
If no label is given to the CYCLE or EXIT statement (this defaults to the present inner loop, i.e. the loop where the CYCLE or EXIT statement being executed), no linking line will be drawn.

5. Hollerith constants are not supported by OmegaChart.

6. Exclamation mark "!" inside of a string is replaced by a space character. Also, to avoid a confusion in the Fortran90 Free Form, a semicolon ";" in a comment line (that starts with a "!" characters) is replaced by a space character.

7. (For Fortran 77 Fix form) Since TAB character (ASCII code Number 9) is non-standard in the FORTRAN 77 or FORTRAN 90 specification, they are replaced by 5 or 6 consecutive characters, depending on whether the line in question is a continuation lines or not.

   a. a sample snips of code

```
{TAB}x = a1 + a2 + a3 + a4 + a5 + a6 +
{TAB}& b1 + b2 + b3 + b4 + b5 + b6
```

   will become

```
bbbbbbx = a1 + a2 + a3 + a4 + a5 + a6 +
bbbbb& b1 + b2 + b3 + b4 + b5 + b6
```

   (where 'b' represent a 'blank space')

   The first line is not a continuation line, and so the TAB character is replaced by 6 spaces.
   The second line is a continuation line, and so the TAB character is replaced by 5 spaces.

   This line is OK for OmegaChart.

   b. However, a very similar snips of code

```
{TAB}x = a1 + a2 + a3 + a4 + a5 + a6 +
{TAB}&b1 + b2 + b3 + b4 + b5 + b6
```

   (note that there is no space between the character '&' and the first 'b' in second line)

   will become

```
bbbbbbx = a1 + a2 + a3 + a4 + a5 + a6 +
bbbbbb&b1 + b2 + b3 + b4 + b5 + b6
```

   and therefore fails the OmegaChart.

   The reason is that OmegaChart recognize the first single character in the second line ("&") as a continuation symbol (case a). However, in case b, the character '&' is not recognized as continuation symbol, rather, it is interpreted as part of the variable '&b1' and thus cause failure.

   As rule of thumb, ALWAYS LEAVE SPACE IN FRONT AND BACK OF CONTINUATION SYMBOL IN COLUMN 6.

   Note: To see the TAB characters, one way is to load the text file into MS-Word and turn on the  symbol

8. TAB characters may cause another issue if the source is download from UNIX system. Since TAB characters are replaced by six consecutive space characters (for non-continuation line), the source code

```
{TAB}READ (*,*) X
```

will be interpreted as

```
bbbbbbREAD (*,*) X
```

This line is fine because the character "R" in READ is on the seventh characters position.

However, a source line:

```
{TAB}10 READ (*,*) X
```

(the "10" is a label and should be positioned in column 1-5, the space between "0" and "R" should be on column 6.)

will become

```
bbbbbb10 READ (*,*) X
```

Such line will cause problems because then the "1" is in the 7th place position.

To avoid these problems, tab character should be removed manually and replaced with appropriate space characters PRIOR sending to OmegaChart..

9. In some cases, an awkward programming style may cause OmegaChart to fail. Two examples are listed below:

a) if-statment

```
If ( condition ) If ( counter) 10,20,30
```

Should be replace by:

```
If ( condition ) Then
If ( counter) 10,20,30
Endif
```

b) empty if-construct

```
If ( condition1 ) then
.........
elseif( condition2) then
else
end if
```

Should be replace by:

```
If ( condition1 ) then
.........
elseif ( condition2) then
! do nothing
else
end if
```

Notes:

It is acceptable to have nothing between "else" and "endif" (but it is not acceptable to have nothing between “elseif" and "else". At least one comment line must be presented between "elseif" and "else".

10. The OmegaChart package should be used as a documentation tool rather than a debugging tool. OmegaChart is not a syntax checker. If your code cannot pass the syntax compilation than almost certainly, OmegaChart will not be able to perform code analysis.

11. The source file has to be in MS-DOS ASCII file format. If you are downloading files from UNIX, make sure it is converted in to MS-DOS ASCII file, (not UNIX text file). A simple way to do this is read the downloaded file by the DOS utility "EDIT", and then re-save (and rename) the file. This will save the file into MS-DOS file. Also, be aware of certain trailing characters (non-ASCII) at the end of file.

12. The current analysis package is not performing analysis of “pre-process” FORTRAN programs. Therefore, meta-command such as “INCLUDE” is not supported. When the input file contains such command, the OmegaChart analysis will stop.

13. For both, Fortran77 and Fortran90 Fixed Form, a control parameter ITRUNCATE in OmegaChart defines the column after which the source code is ignored. For example, in standard Fortran 77, some users use the space between column 73 and column 80 for comments. These comments will be IGNORED by OmegaChart, and it is necessary to set ITRUNCATE=73, which tells OmegaChart to truncate all the character after column 73 before going to the pre-process steps of OmegaChart. Some compilers support “extended-source”, i.e., source line longer than 72 characters (80 or 132 characters maximum). In these cases, simply adjust ITRUNCATE to appropriate value.

    For Fortran90 free form, value of ITRUNCATE is not used and all the characters in a line are significant.

14. For Fortran90 Free Form, the comments after the “&” sign in a source line is ignored. For example:

```
x=a1 + & ! comment1
a2 + &   ! comment2
a3 + &   ! comment3
a4       ! comment4
```

    in this case, “comment1”, “comment2” and “comment3” are ignored, only “comment 4” will be shown in the generated flowchart.

    For both, Fortran77 and Fortran90 fixed form, the comment after "!" sign in a continuation line is ignored. For example:

```
x=a1 + ! comment1
```

```
& a2 + ! comment2
& a3 + ! comment3
& a4 + ! comment4
```

In this case, "comment1", "comment2", "comment3" are ignored. The only "comment4" will be shown in the generated flow chart.

15. The comment within a complete statement, even though is was spread out in several lines, will cause OmegaChart 4.0 to fail. For example, the following lines are not permitted.

```
 X = a1 + a2 + a3 + a4 + a5 + a6 +  ! a-sequence
&    b1 + b2 + b3 + b4 + b5 + b6 +  ! b-sequence
&    c1 + c2 + c3 + c4 + c5 + c6    ! c-sequence
```

Another example may be

```
 call SubProcedure (a1, a2, a3, a4, a5, a6,  ! input  a's
&                   b1, b2, b3, b4, b5, b6 ) ! output b's
```

5.2 Visual Basic Family of Languages

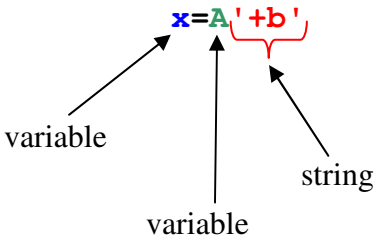
1. Since ONLY the program execution flows are of interest to OmegaChart, all other non-executable declarations, particularly those appeared in the "public area", i.e. in between program modules, is treated as "comment".

5.3 Matlab Scripting Language

1. Matlab uses single quote (') to denotes matrix transpose, while also uses two single quote as delimiter for string. The utilization of one symbol in two irrelevant contexts confuses OmegaChart. For example, a statement such as

   `x=A'+B'` (assign the sum of the transpose of matrix A and the transpose of matrix A to variable x)

   will be treated as



   and therefore cause error. Generally speaking, if there are two single quotes appeared in the same line of statement, it will be treated as a string.

   OmegaChart provide several methods to warn the user on this potential faulty situation:

   a.) If there is only one single quote, then this is a matrix transpose notation.
   b.) If there are other odd number of single quote, OmegaChart will send you a warning message and quite processing any further because there are high possibility to crash the execution.
   c.) If there are even numbers of single quote, they are treated as pairs of string delimiter, regardless of the situation. Sometime this may still crash the execution, sometime it may not.
   To provide details for this scenario, OmegaChart will generate a .log file (in the directory where source code resides) recording those lines with more than two single quotes. User may then be able to manually edit these lines to make it comply with OmegaChart.
   d.) The easiest way to handle this situation is simply remove these single quote and, upon finishing generate the flow chart and OmegaMath expression, manually put it back on the flow chart.

5.4 NCL Macro Language

5.5 TCL/Tk Scripting Language

1. OmegaChart support those TCL/TK command used in Hypermesh package.

5.6 C/C++ Language

1. OmegaChart is used for the analysis of ACTUAL command line and not Meta-command. Therefore, LOGICS behind lines such as

```
#INCLUDE
```

   and

```
#IF
#ELSE
#ENDIF

#ifdef
```

   are ignored and these lines are considered as comments

2. Since C/C++ language support OVERLOADED module, functions with same name are renamed sequentially. For example, the first occurrence of a function "`fun`" is not renamed. The second occurrence of the function "fun" is renamed as "`fun_OVERLOAD_2″`. The third occurrence of the function "fun" is renamed as "`fun_OVERLOAD_3″` etc.

6. Viewing and Printing of Generated Flow Charts

6.1 Output to Microsoft Excel

This option creates a single Excel file, which contains many pages, each corresponds to a program module in the input source code (see Fig. 9). The Excel file is generated WHILE OmegaChart is running. In other words, OmegaChart uses Excel as one of its sub-procedure, which perfroms the output generation. User may actually see the generating process. When OmegaChart completed generation of flowcharts, Excel will dismiss automatically or prompting you whether to replace an existing file.

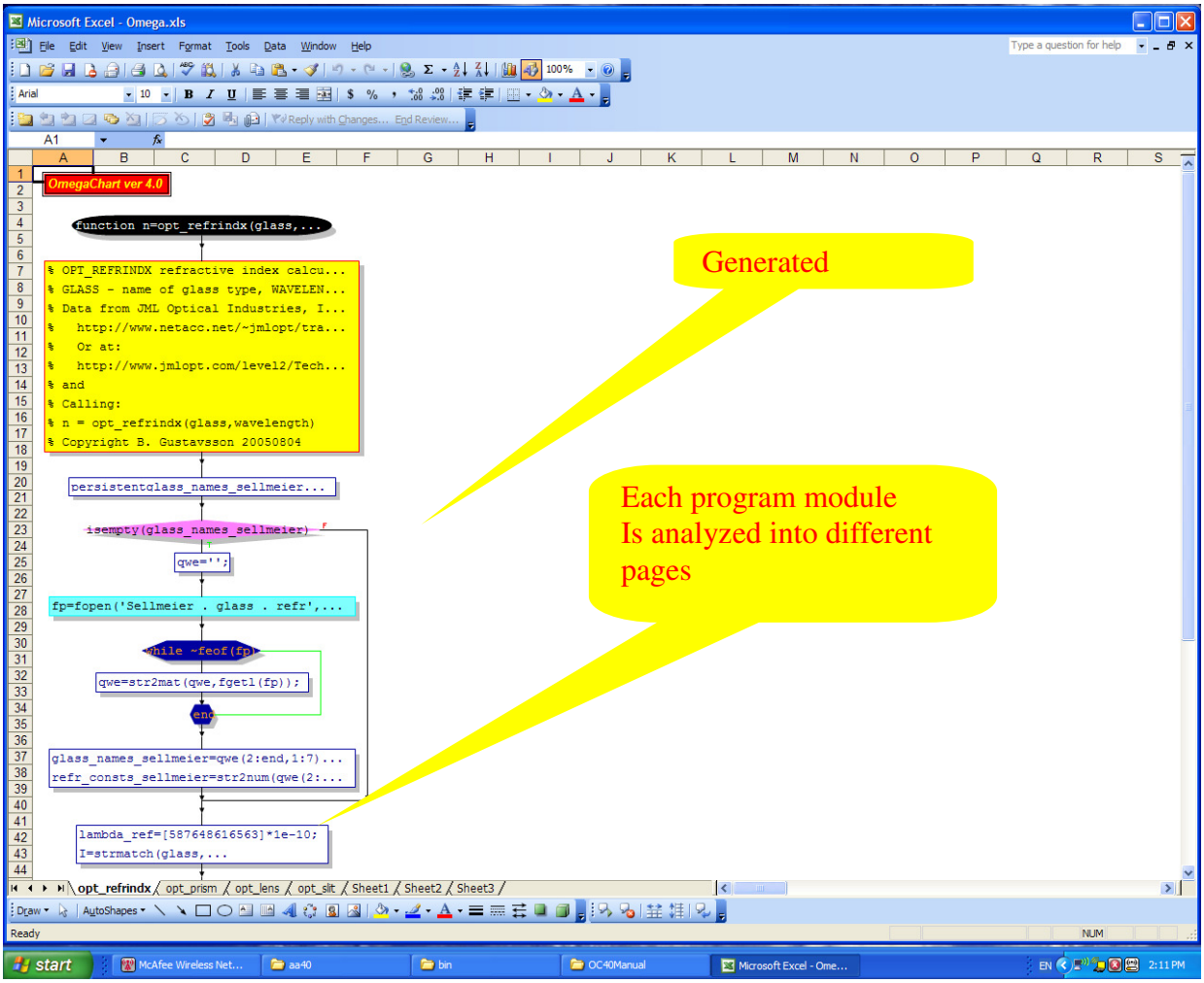


Fig. 9 Example of the Excel output

Printing of the flowchart can be accomplished by using windows' printer drivers just like printing of any other Excel documents. The size of the generated flowcharts varies, depending on the complexity of the routine being analyzed. Quite often, the user may encounter extraordinary large flowchart and chances are it will not fit into a standard A4 or letter size paper. If a large size plotter is accessible, make every effort to use it. If only letter size printer is available, the following is suggested:

(1) Print the flowchart on a continuous paper (8.5" x 11") with zero margin setting (top, bottom, left and right, this depends on the printer you are using). This may produces several long paper strips. Then you tape them together side by side to make the final flowchart.
(2) Print the generated flowchart on standard A4 paper and also generate an outline view of the overall flowchart as "index" of these pages [from Visio→ print preview → screen dump] (Fig. 10)
(3) Print the generated flowchart on standard A4 paper and then paste them on a large A0 sized paper (Fig. 11)

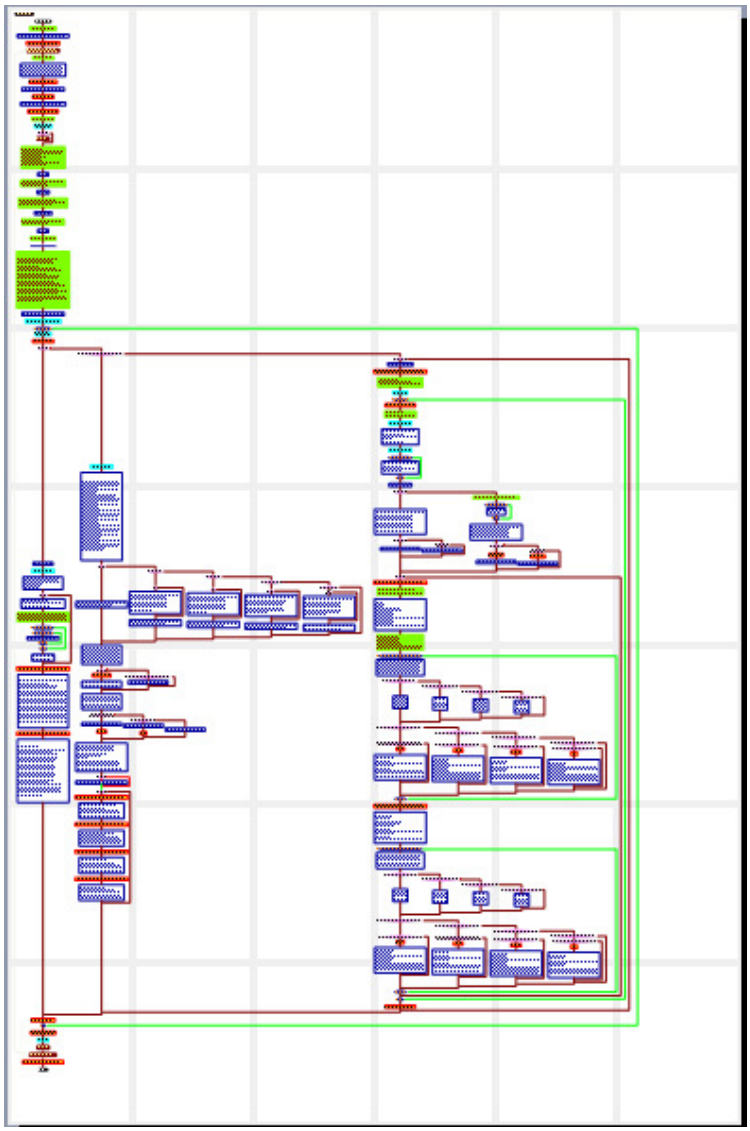


Fig. 10 Index of pages

Fig. 11 Several pages pasted on a large sheet of paper

6.2 Output to SVG file (viewable via Microsoft Internet Explorer)

This option creates many files and directories for a single source code. The flowcharts are viewable only when OmegaChart is completely done with the process. When finished, there will be several files and sub-directories in the specified output directory:

File "**`index.htm`**": This is the most important output file. By opening this file, user may see a browser shown in Fig 12. This is where you may start digging into the source code. (Explained below)

File "**`menu.htm`**": A supplementary file that provides the drop-down menu for selection of the module.

File "**`AltName.htm`**": A supplementary file for use with OmegaMath, see Section 7.

Sub-directory "**`code`**": This directory contains many .html file, each contains html tagged-source-code for the corresponding module in the source file.

Sub-directory "**`oc60`**": This directory contains many .svg files (Scalable Vector Graphics), each contains SVG definition for the corresponding module in the source file. The SVG definition described how the flowchart is drawn. Also in this directory are yet another set of .html files. These html files are for printing purpose (explained later).

Sub-directory "**`math`**": This directory contains many .html files; each contains MATHML definition for the corresponding module in the source file. The MATHML definition is used in OmegaMath display.

The Internet Explorer Browser is divided into four panels:

-Module Selection Panel
-Source Code Viewing Panel
-FlowChart Viewing Panel
-OmegaMath Viewing Panel

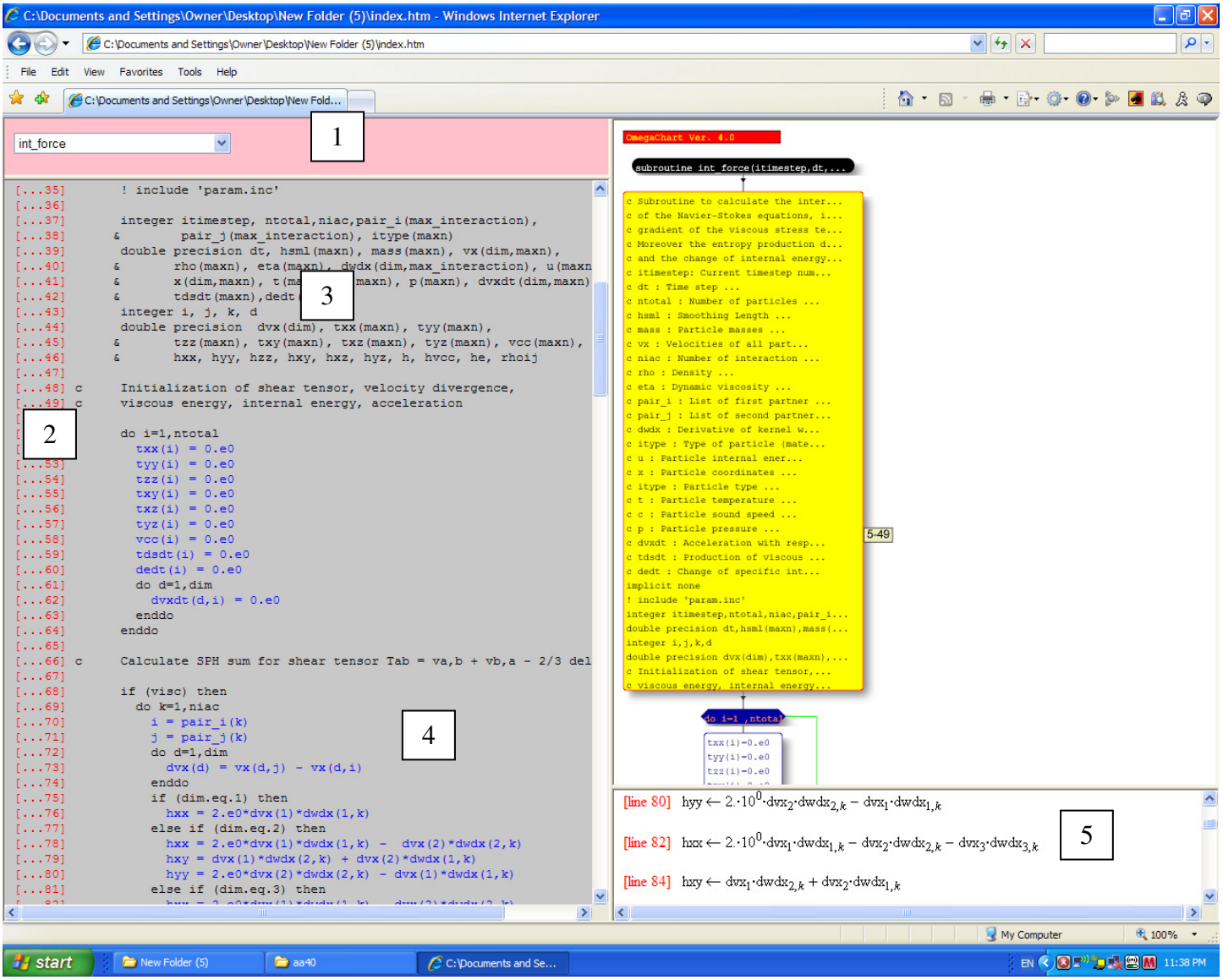


Fig. 12 An example of OmegaChart output generated using Microsoft Internet Explorer

Each panel has its own sub-components.

- The drop down menu (1) allows the selection of module in the file, which should be display. The displayed information includes original source code (displayed in Source Code Viewing Panel), corresponding flowchart (displayed in Flowchart Viewing Panel) and corresponding 2-D math expressions for all assignment statements and "method calls".

- The line numbers for the original source code are indicated in the column (2). These line numbers are referenced in the generated flowchart.
  - The black text (3) corresponds to the non-assignment statements.
  - The blue text corresponds to the assignment statements and "method calls". These texts acts as hyperlink to the corresponding mathematical expressions. User may click on the texts to see the mathematical expression in the OmegaMath Viewing Panel (5).

- The generated Flowchart is displayed in the right part of the Microsoft Internet Explorer panel.

Navigation in SVG viewer

Right-click in the SVG image area (FlowChart Viewing Panel) will open the contextual pop-up menu. This will reveal the commands and options for interacting with the SVG image.

Panning: Hold the alt key and click-and-drag with the mouse to pan an SVG image. When the scroll lock is enabled and Adobe SVG Viewer has the focus, the arrow keys may be use to pan the image.

Zooming: Hold the control key and click to zoom in at the mouse pointer location. Hold the control key and click-and-drag to select a region to zoom in. Hold the control and shift keys and click to zoom out. You can also use the zoom commands in the context menu.

Printing of SVG Image

User may not be able to print the SVG image (flowchart) directly from the web browser windows. In order to generate hard copy, it is recommended to copy and past the SVG image (via contextual pop-up menu, "copy SVG" option) onto Microsoft Paint program (on MS-Windows system, Start → All Program→Accessories .→Paint). However, if you perform this operation from index.htm (Fig 12), you may end up with copying only a portion of the SVG image. To resolve this problem, OmegaChart save a set of the html files (located in the OC60 directory, one for each module.) After launching the Microsoft Internet Explorer and loading appropriate html file from OC60 directory, this file may be "Copy and Paste" to Microsoft Paint and printed from there.

Printing large flowcharts from Microsoft Paint may be a challenge. To produce a hard copy of the large flowcharts, it is recommended to generate the flowchart in Excel, rather than SVG file. On the other hand, for on-screen viewing, SVG mode is more interactive.

6.3 Output to DAT file (viewable via Microsoft Visio READER and Microsoft Excel READER)

This option creates a single .DAT file, which will be processed later by "VIEWER" (included with the package" Two viewer is provided: OC60.vsd (Microsoft Visio, Fig 13) and OC60.xls (Microsoft Excel). The operation is relatively easy:

a.) start the viewer & enable macro [note]
b.) click on the OmegaChart ver. 4.0 icon, a standard "open file dialogue box" appears, asking about the .DAT file
c.) User specifies the filename and click "open", and then the flowchart will be generated.
d.) For Microsoft Visio, an addition question will be asked: "Do you want to glue lines to shape? ". Please refer Section 4, Note d) in this manual for details on this option.

Printing of the generated flowchart from MS-Excel or MS-Visio follow the same rule as described in secion 6.1. MS-Visio user may have more flexibility on formatting output.

Note: user need to set the security level to "medium" in order to run the macro in MS-Excel or MS-Visio. Please go to Tools→ Macro→ Security to set the security level for MS-Excel or MS-Visio.

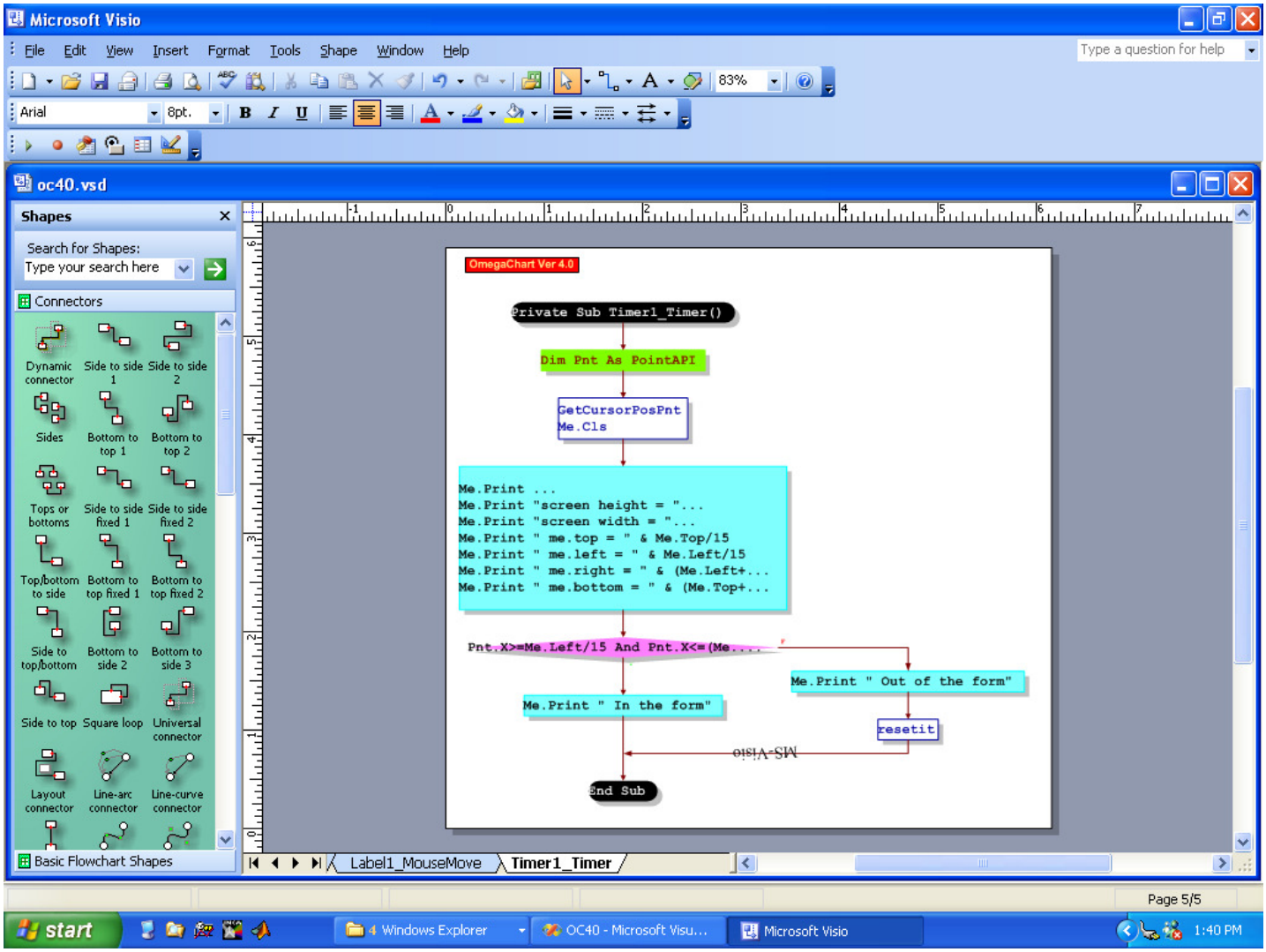


Fig. 13 An example of OmegaChart output generated using Microsoft Visio

7. OmegaMath Features

OmegaMath (ver. 4.0) is a specialized module of OmegaChart which is capable of performing statement-level analysis of source code. In most cases, the mathematical calculation is performed when an assignment statement or a "method calls" is been executed. Therefore, it is the right hand side of the assignment statement that is to be analyzed. For example, the quadratic formula, when coded into Fortran language, may look like

```
X1= (-BETA +sqrt(BETA* BETA -4* ALPHA *GAMMA))/(2*ALPHA)
X2= (-BETA -sqrt(BETA* BETA -4* ALPHA *GAMMA))/(2*ALPHA)
```

These “one-dimensional” expressions can be converted, via OmegaMath, to their original “two dimensional” mathematical expression

$$X1 \leftarrow \frac{-\beta + \sqrt{\beta^2 - 4 \cdot \alpha \cdot \gamma}}{2 \cdot \alpha}$$

$$X2 \leftarrow \frac{-\beta - \sqrt{\beta^2 - 4 \cdot \alpha \cdot \gamma}}{2 \cdot \alpha}$$

Expressions with the radical symbol, super/subscript and/or fractions provide a more vivid view of the mathematical contents behind the program code.

Variable Replacement Feature

Mathematicians have typically more freedom than programmers do because they may use symbols and notations that are NOT supported by ASCII specifications. Programmers are limited to "translate" Greek letters into their respective ASCII counterparts. Therefore, to represent the symbol α that mathematician uses, programmer has to spell it out **ALPHA**. Nevertheless, the letter α still provides more insight into the meaning due to user familiarity. The above example (quadratic formula) could have been converted into

$$X1 \leftarrow \frac{-BETA + \sqrt{BETA^2 - 4 \cdot ALPHA \cdot GAMMA}}{2 \cdot ALPHA}$$

$$X2 \leftarrow \frac{-BETA - \sqrt{BETA^2 - 4 \cdot ALPHA \cdot GAMMA}}{2 \cdot ALPHA}$$

However, this form is less familiar.

For this reason, OmegaMath provides a table to guide the conversion process. This table can be view by loading the file "AltName.htm" (from the user specified Output directory) into Microsoft Internet Explorer.

The format how variables are displayed is shown in Fig. 14. "alpha" is to be displayed as α, variable "beta" is to be displayed as β, etc.

Module: All

alpha	α
beta	β
gamma	γ
delta	δ
epsilon	ε
zeta	ζ
	η

Fig. 14 Relation between variable names in the source code and in OmegaMath.

In the case when the variable "**AFA**" and not "**ALPHA**" should be displayed as α, user must edit the html file using any ASCII editor. An example of such change is shown in Fig. 15.

```
<html>

<H2>Module: All </H2>

<table border="1" cellpadding="5">
<tr> <td> AFA   </td> <td> <i> &#x03B1; </i></td> </tr>
<tr> <td> beta  </td> <td> <i> &#x03B2; </i></td> </tr>
<tr> <td> gamma </td> <td> <i> &#x03B3; </i></td> </tr>
.....................................................
<table>


</html>
```

Fig. 15 An example of edited html file

Since OmegaChart (and OmegaMath) performs the analysis of several modules in one single run, a variable called "**AFA**" in subroutine "**SUB1**" may actually representing the same parameter as the variable "**ALPHA**" in main program "**MAIN**" and both variables should be displayed as α. In these cases, the AltName.htm file needs to be edited as shown in Fig. 16 (the highlighted areas).

```
<html>


<H2>Module: All </H2>
<table border="1" cellpadding="5">
<tr> <td> ALPHA </td> <td> <i> &#x03B1; </i></td> </tr>
<tr> <td> beta  </td> <td> <i> &#x03B2; </i></td> </tr>
<tr> <td> gamma </td> <td> <i> &#x03B3; </i></td> </tr>
.....................................................
<table>


<H2>Module: MAIN </H2>
<table border="1" cellpadding="5">
<tr> <td> ALPHA </td> <td> <i> &#x03B1; </i></td> </tr>
<table>


<H2>Module: SUB1 </H2>
<table border="1" cellpadding="5">
<tr> <td> AFA </td> <td> <i> &#x03B1; </i></td> </tr>
<table>


</html>
```

Fig. 16 An example of edited html file

Notes:

1. The module named “ALL” is the “default”, and may be customized by editing and/or appending additional module name and specification lines after it.
2. If a file named “AltName.htm” is found in the specified output directory, it will be used by OmegaChart. If file does not exist a “default” AltName.htm will be place in the output directory.

8. Configuration Management

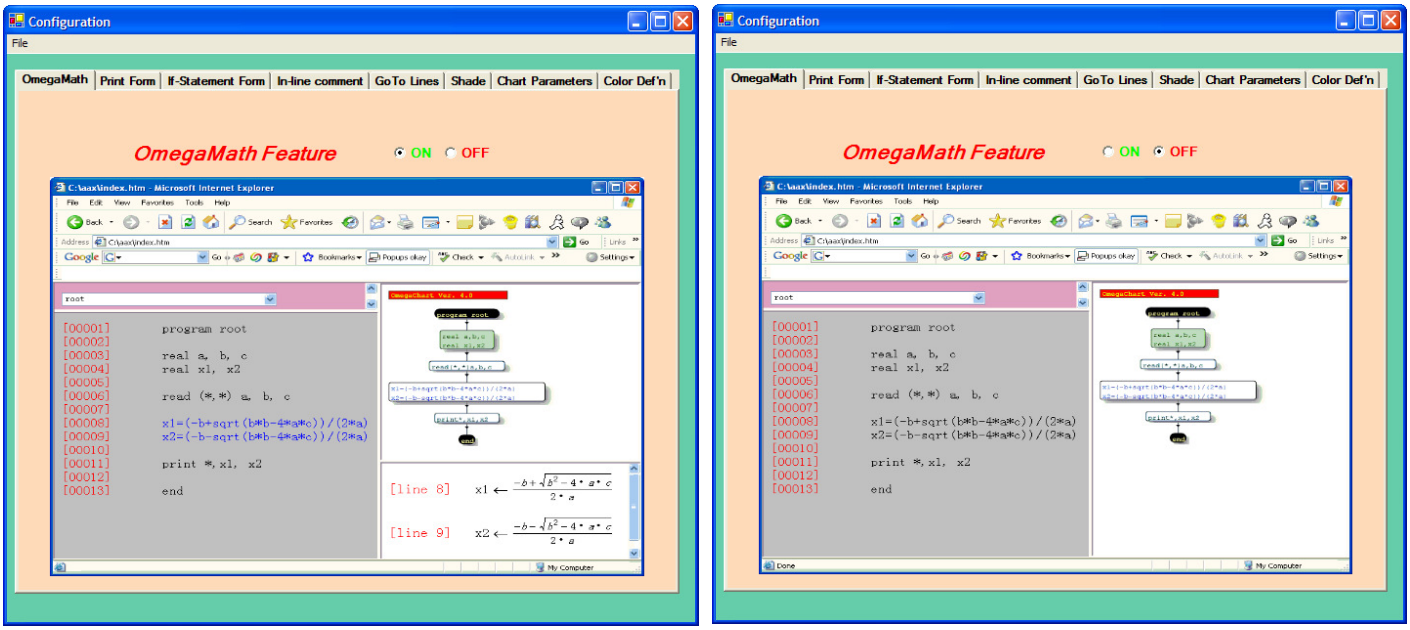
Omega Chart provides several options for the flowchart generating process. The default configuration file is setup during the program installation (default.cnf). However, the user may “Load” a configuration file (*.cfg), “Edit”, and “Save” his own modified file. Omega Chart provided three additional configuration files:

- default_small.cnf: same as default.cnf, but use a smaller font size.
- print.cnf: generates flowchart with no background color.
- pring_small.cnf: same as print.cnf, but use a smaller font size.

The Configuration file options:

1. OmegaMath

   User may disable OmegaMath if it is not use (Fig. 17).

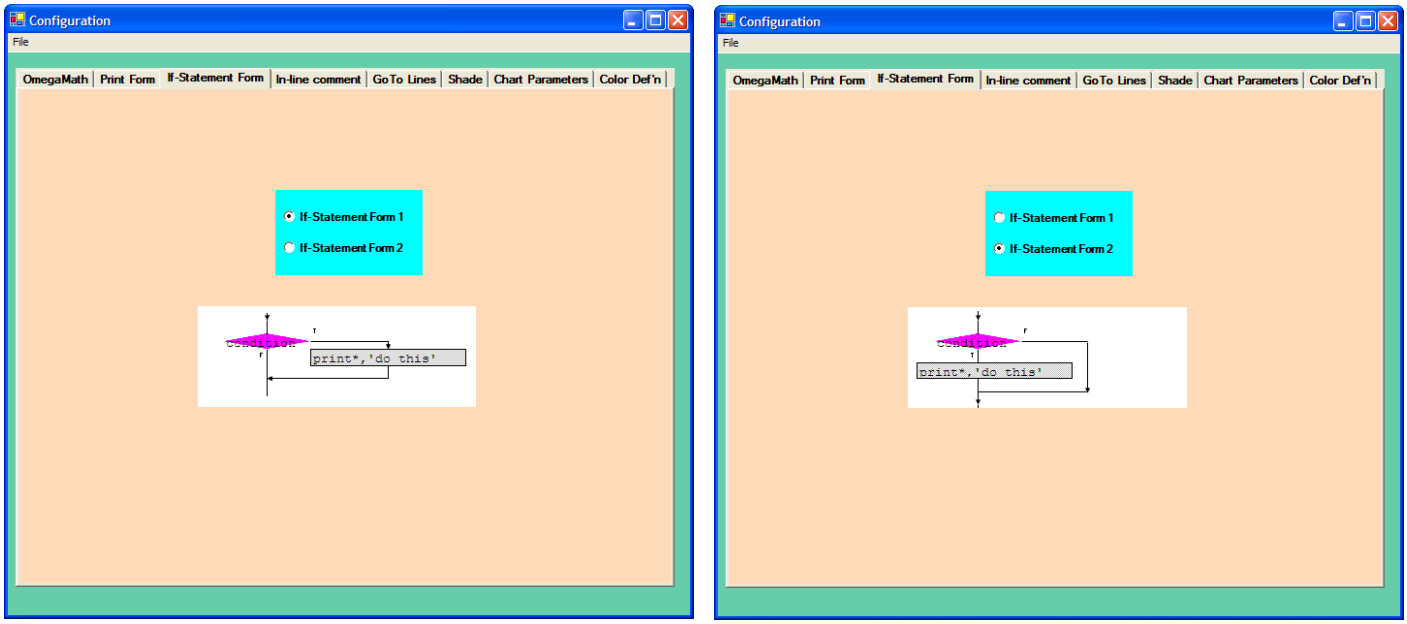


(a) **(b)**

Fig. 17 OmegaMath options

2. If-Statement Form

User may specify two forms of the IF statement. If the FORM 1 is selected, the "conditional task" part of the statement is considered as a "side track" of the main program flow (Fig. 18a). If the FORM 2 is selected, this "conditional task" part is located inside the main program flow as shown in Fig.18b.

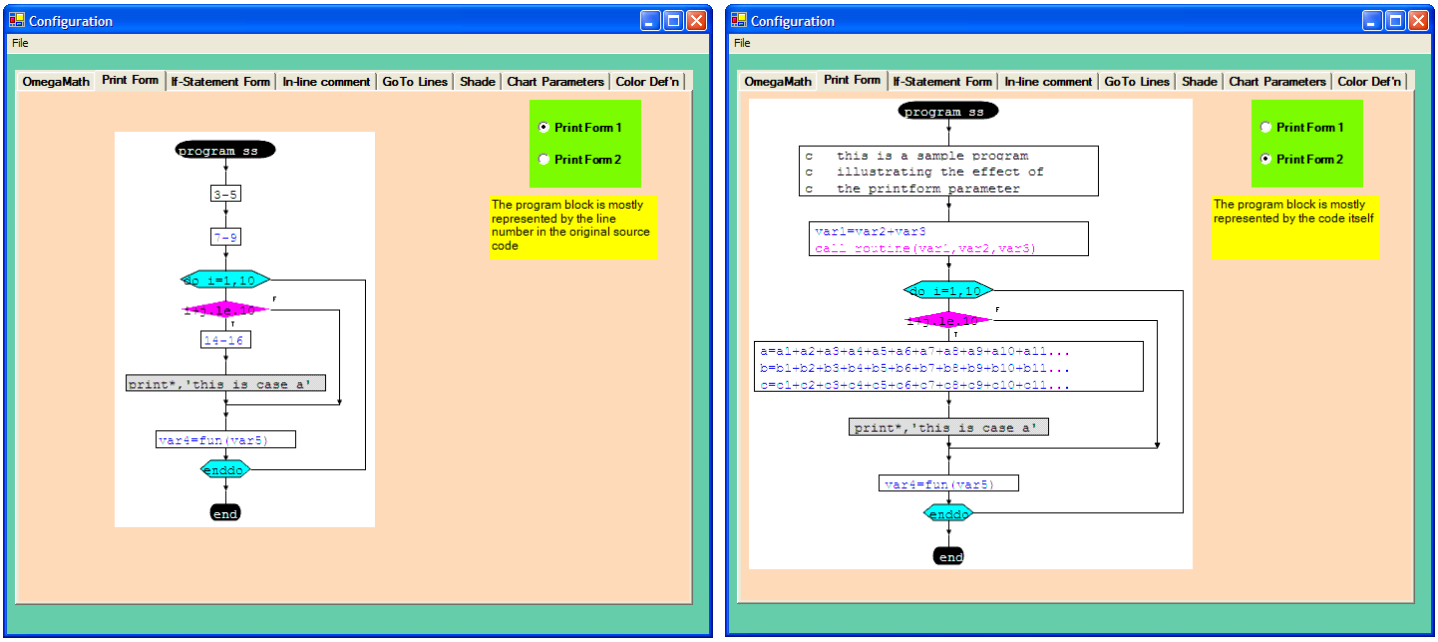


(a) **(b)**

Fig. 18 Specification in the IF-Statement Form

3. Print Form

There are two ways how to present the program flowchart. If the parameter PRINT FORM=1, is selected, the program line numbers are displayed in the generated flowchart (Fig. 19a). If the parameter PRINT FORM=2 is specified, the generated flowchart use only the source program statement (Fig. 19b). Because the actual program statements may be lengthy, only the left portion of the statement is shown. The parameter LENLIMIT is used to specify the maximum length (number of characters) of the statement before truncation.

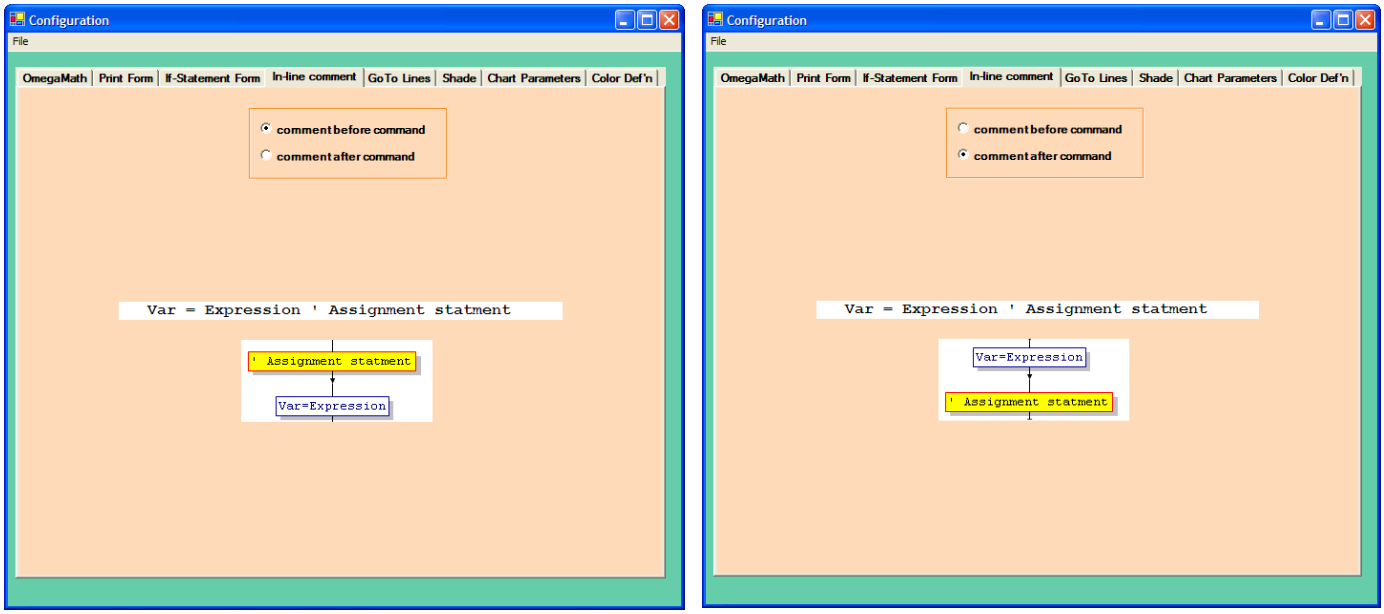


(a) **(b)**

Fig. 19 Specification in the Print Form

4. Inline comment

This option controls the way an inline comment is presented, either before the attached command (Fig 20a) or after it (Fig 20b)

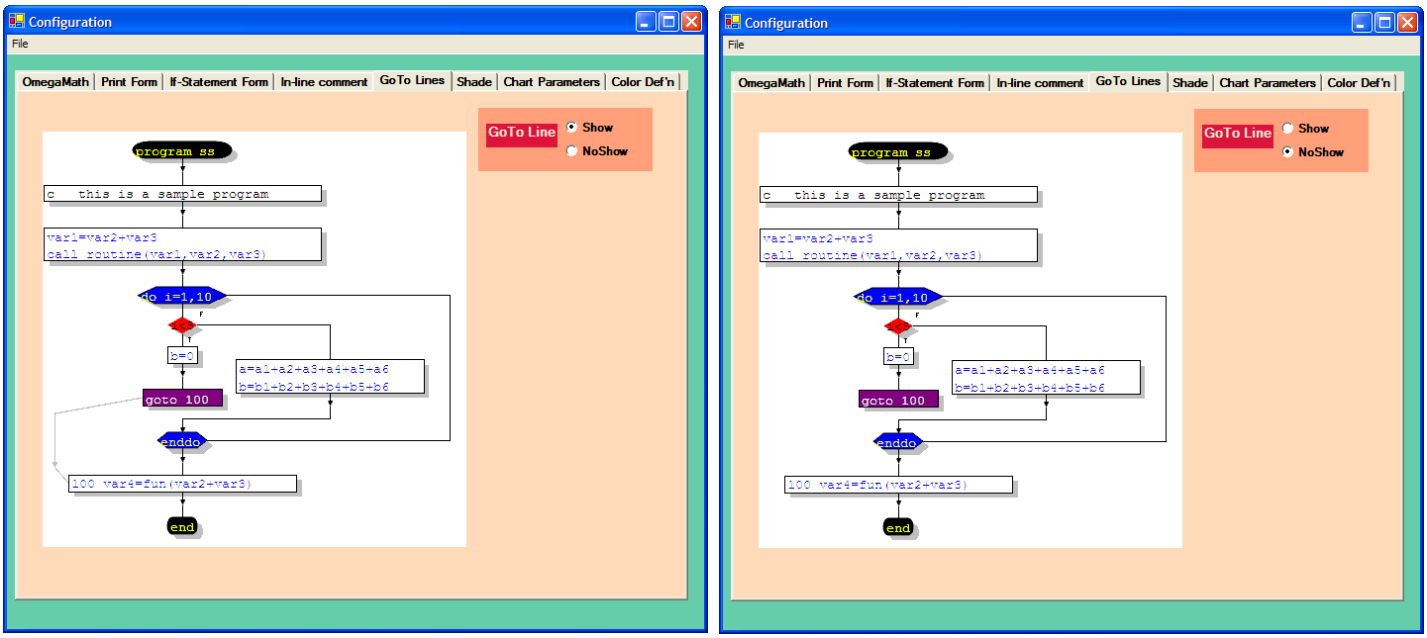


(a) **(b)**

Fig. 20 Specification for Inline comment

5. GoTo Lines

This Boolean value determines whether to draw the GoTo Lines. In some cases, drawing such lines may be very confusing.

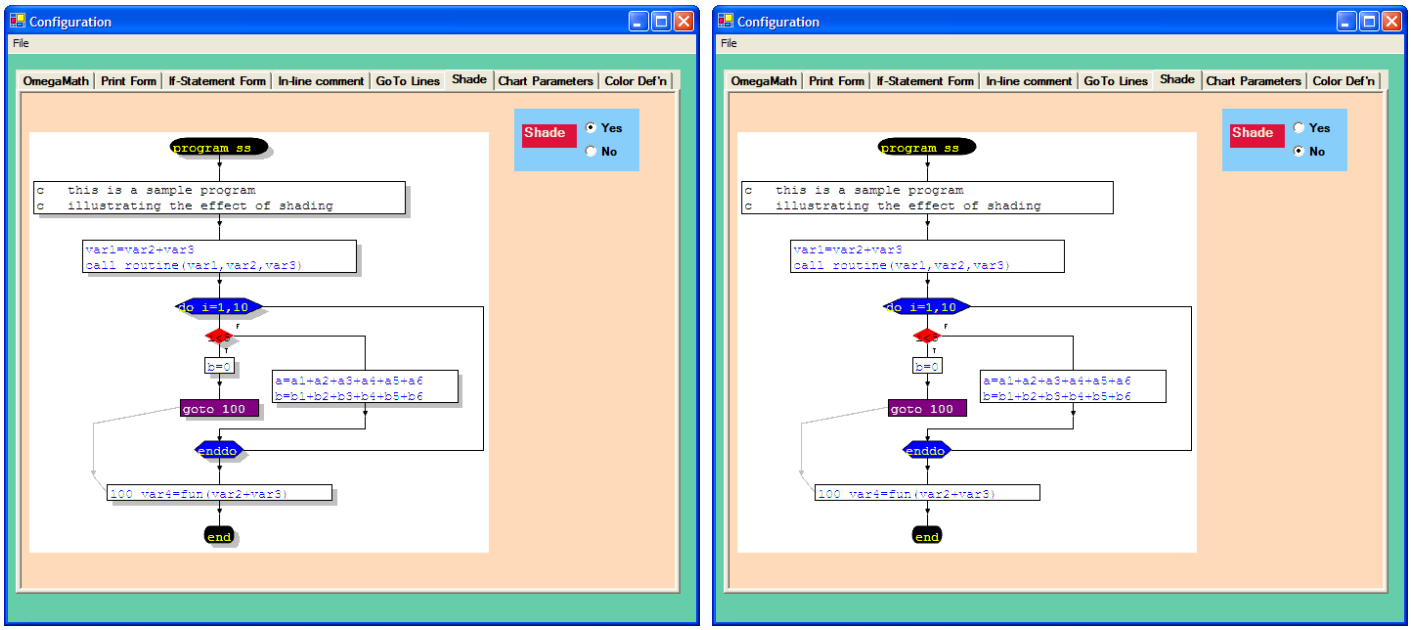


(a) **(b)**

Fig. 21 GoTo line specification

6. Shade

The Boolean SHADE value determines the outlook of the generated chart. If the SHADE is enabled the Excel files may be approximately 15% - 20% larger in comparison with files without shadowing.



(a) **(b)**

Fig. 22 SHADE specification

7. Chart Parameters

When the mouse is hovered over a label or text box, a graphical illustration is displayed and shows the meaning of particular variable (in red, see Fig. 23). The variables are:

- Horizontal Size
- Vertical Size
- Horizontal Spacing
- Vertical Spacing
- LenLimit
- Font size
- ITruncate
- DividerRatio

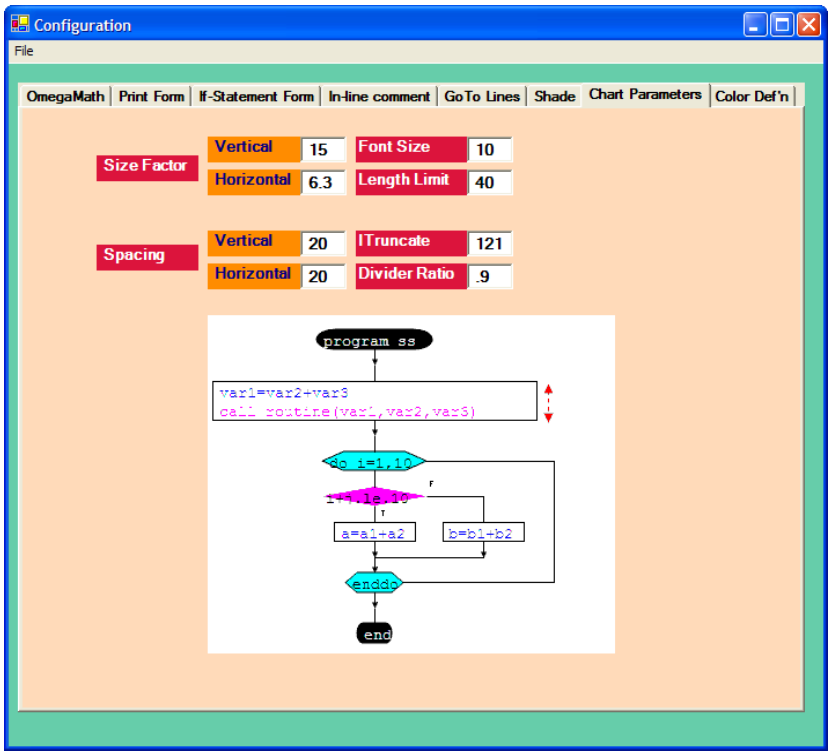


Fig. 23 Chart Parameters

• Horizontal and Vertical Size are the space taken up by a SINGLE character. The larger value of the "Font Size" variable, the larger value of Horizontal and Vertical size should be.

• Horizontal and Vertical Spacing are spacing between blocks of flowchart unit.

• The parameter "Font Size" is self-explanatory.

• Please refer to the parameter PRINTFORM for explanation of the variable LENLIMIT.

• Please refer to Section 5.1.1, item 14 of this manual for explanation of the variable ITRUNCATE.

• The variable DividerRatio is the threshold of the following quantity :

$$\left(\frac{\text{The Number of Occurance of the MOST Frequent Character in a line}}{\text{Total Number of Characters in a line}} \right)$$

For example, consider the line

```
C---------------------- Sample --------------------------
```

In this line, the total number of character is 58 (including the space character), of which 49 is the "most frequent character", "-". So the ratio in question is 49/58 = 0.845. Since it is greater than the prescribed (default) value of 0.7, THIS LINE IS REGARDED AS A DIVIDER AND WILL BE IGNORED FROM THE FLOWCHART.

On the other hand, consider the line

```
C------ Sample ------
```

The total number of character is 22 (including the space character), of which 12 is the "most frequent character", "-". So the ratio in question is 12/22 = 0.545 < 0.7. This line WILL NOT be regarded as a divider and therefore WILL BE included in the flowchart AS A COMMENT LINE.

This mechanism is used to skip some unwanted comment line in the flowchart and therefore save some paper when hardcopy is produced.

8. Color Definitions

This page documents the color definition of various “components” of the flowchart. The user may click on the icon (text, colored box or frame) to change the corresponding color (See Fig. 24).

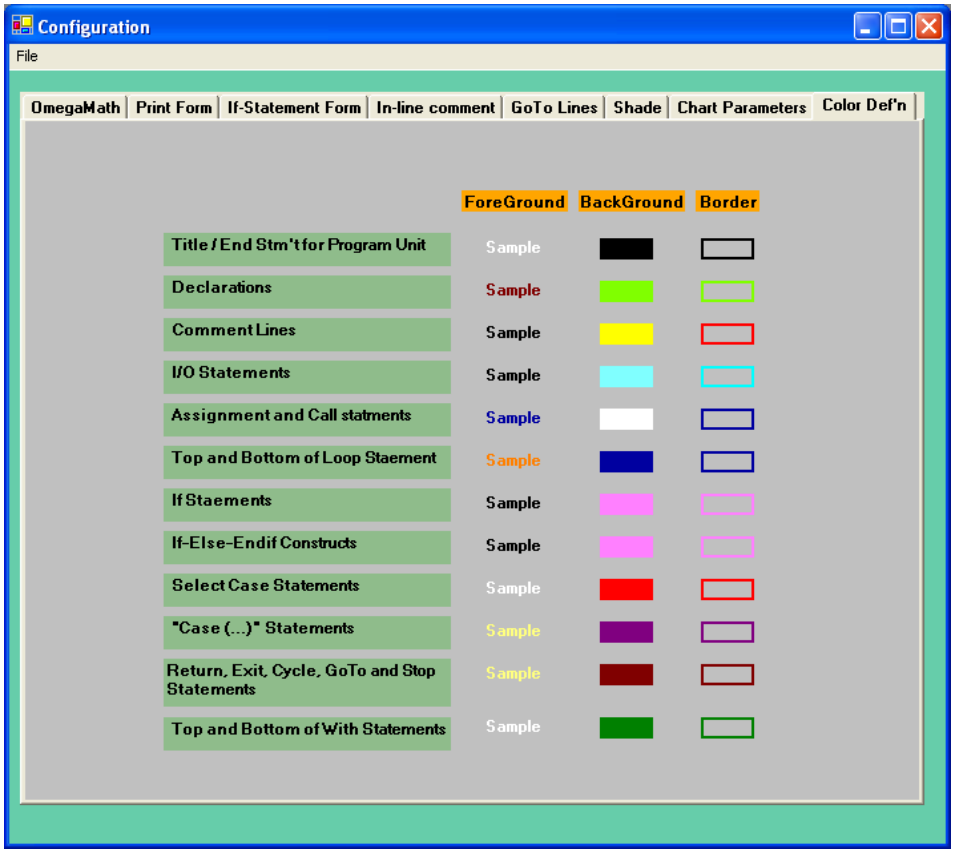


Fig. 24 Color Definition

9. Information in the “run.log” file

There is a file named “run.log” in the launch directory. This file is a chronological log of the application of OmegaChart. It is also used to estimate the “current program execution time”. The file is simply an excel .csv file and consisted of three fields:

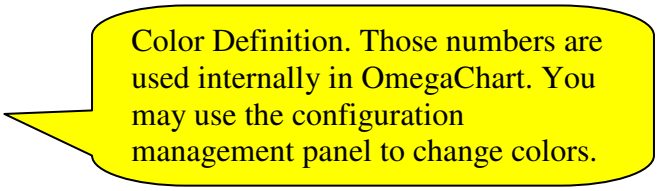
```
Program file name, Number of line in that file, Execution time.
```

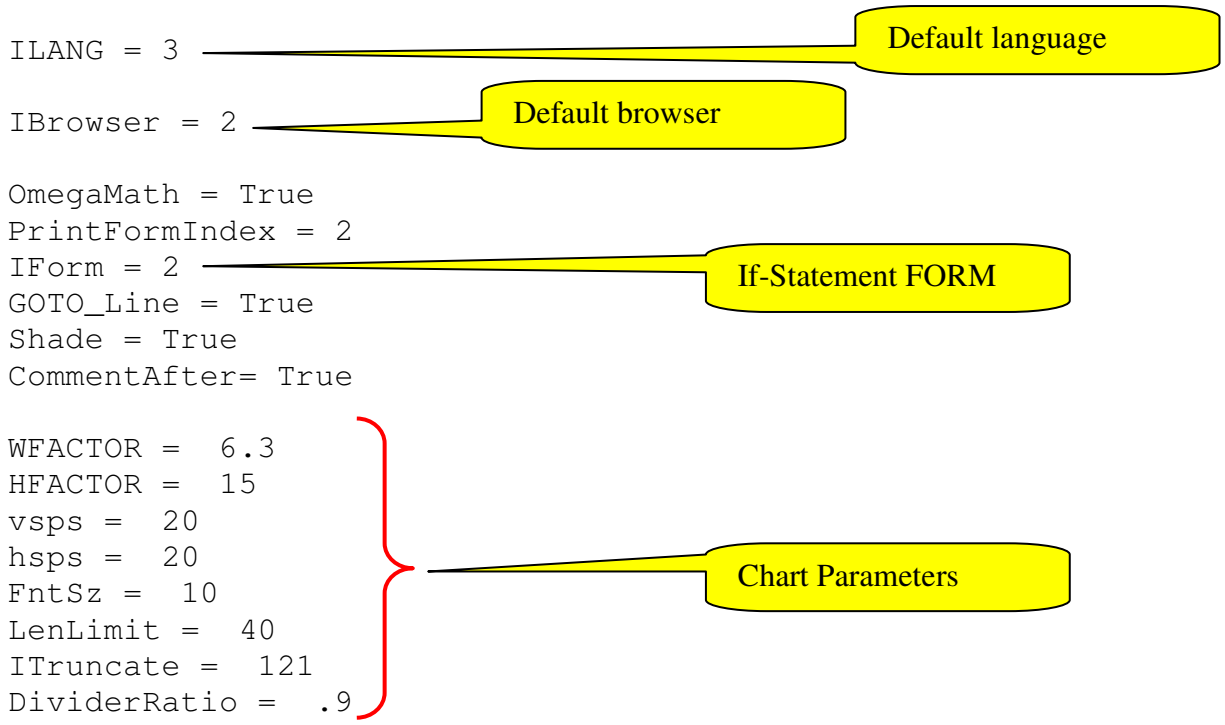
When this file is too big, user may edit it and delete some of the lines.

10. Information on Configuration Files

The .cfg file is an ASCII file that consists of the following system parameters. Depending on user preference, any one of these parameters may be customized.

```
FG1 = -1
BG1 = -16777216
BDR1 = -16777216
FG2 = -8388608
BG2 = -8323328
BDR2 = -8323328
FG3 = -16777216
BG3 = -256
BDR3 = -65536
FG4 = -16777216
BG4 = -8323073
BDR4 = -16711681
FG5 = -16777056
BG5 = -1
BDR5 = -16777056
FG6 = -32768
BG6 = -16777056
BDR6 = -16777056
FG7 = -16777216
BG7 = -32513
BDR7 = -32513
FG8 = -16777216
BG8 = -32513
BDR8 = -32513
FG9 = -1
BG9 = -65536
BDR9 = -65536
FG10 = -128
BG10 = -8388480
BDR10 = -8388480
FG11 = -128
BG11 = -8388608
BDR11 = -8388608
FG12 = -1
BG12 = -16744448
BDR12 = -16744448
```





11. Why does OmegaChart Fail?

In the following list are several reasons why the OmegaChart program may not be able to complete the requested analysis:

- The source code cannot pass the syntax compilation of the Fortran code.
- The source code contains the Hollerith constants
- The parameter ITRUNCATE is not setup correctly.
- The source code contains the {TAB} characters.
- The source code is a mix of fixed form and free form. OmegaChart can only perform the analysis of one form at a time.

12. Technical Support & Contact Information

For technical support, question and suggestions, please send email to

services@omegachart.com
Please report any Bug to Technical Support at services@omegachart.com