SIRILIC

Stephen Noyes
Ken Cottage, Upper Lane, Brighstone, Isle of Wight, PO30 4AT
steve@noyes.org.uk

CONTENTS

1. INTRODUCTION

SIRILIC is written for Microsoft Windows in Delphi, and is distributed free: there are too many proof programs about already for anyone to make any reasonable amount of money! It is available from my web site www.noyes.org.uk. Strictly it is “postcardware”, so if you use it (i.e. prove & then ring a composition) then you should please send me a postcard of a church in “Dove”; if you put your e-mail address on the card then I’ll send you messages as and when updates come out.

The basic idea of SIRILIC was to give me a “Windows” front end to the microSIRIL that I bought from Andrew Craddock in 1989; I am using Windows 2000 and 95. At that time microSIRIL had become the “Gold Standard” for peal proving programs and there was always going to be a risk involved in moving to another program or writing my own .. for example, the risk of ringing a false peal! A simple Windows program that submitted a file to the PROVE.COM engine in microSIRIL was simple enough to produce.

However, I soon wanted the program to do more things, such as finding the number of combination roll ups, higher numbers of bells, tests for “all the work”, 1440s of Minor, friendlier error message and so on. In the end, it was easier to construct SIRILIC as a full peal proving program in its own right, with an input file format identical to microSIRIL; then as a double check it could also send its input file off to microSIRIL to check that the output was *IDENTICAL*. It fails the check of course on more than 12 bells, because microSIRIL will not work beyond 12 bells.

The method of proof in SIRILIC is real “sledge hammer” stuff: generate all the changes as text strings, and put them in a Delphi list that produces an exception if you enter the same one twice! This has the advantage that it uses a completely independent method of proof from the elegant method of microSIRIL, giving an independent check on proof.

2. INSTALLATION

Firstly, download the zip file sirilic.zip from my web page www.noyes.org.uk. Put sirilic.zip in a folder where a new subfolder will be created to contain SIRILIC. Now, right click on sirilic.zip and select “Extract to folder ..\Sirilic; the Use folder names check box should be set in the Extract dialog box (it usually is by default). You should then see a new folder created called “Sirilic”. Move into this folder by double clicking, because that is where the program has been put.

The contents of the new Sirilic folder are as follows:

SIRILIC.exe	The SIRILIC program	
SIRILIC.ini	An initialisation file that stores setup parameters for SIRILIC.EXE; the registry is not used	
README.txt	A file giving very basic information, refers to this document for installation etc.	
microSIRIL	A subfolder containing:	
	PROVE.com	The Proof engine used my microSIRIL; see note below
	Prove Documentation.rtf	The original documentation for PROVE.COM
Examples	A subfolder with example input files for SIRILIC	
Method Library	A subfolder that could contain microSIRIL method libraries, see the link from my web page www.noyes.org.uk to get an up to date copy of these	

NOTE: the PROVE program is copyright 1987 by Sam Wilmott, and is based on programs and algorithms by Andrew Craddock & Hamish McNaughton; see the documentation accompanying the PROVE program for restrictions on its distribution.

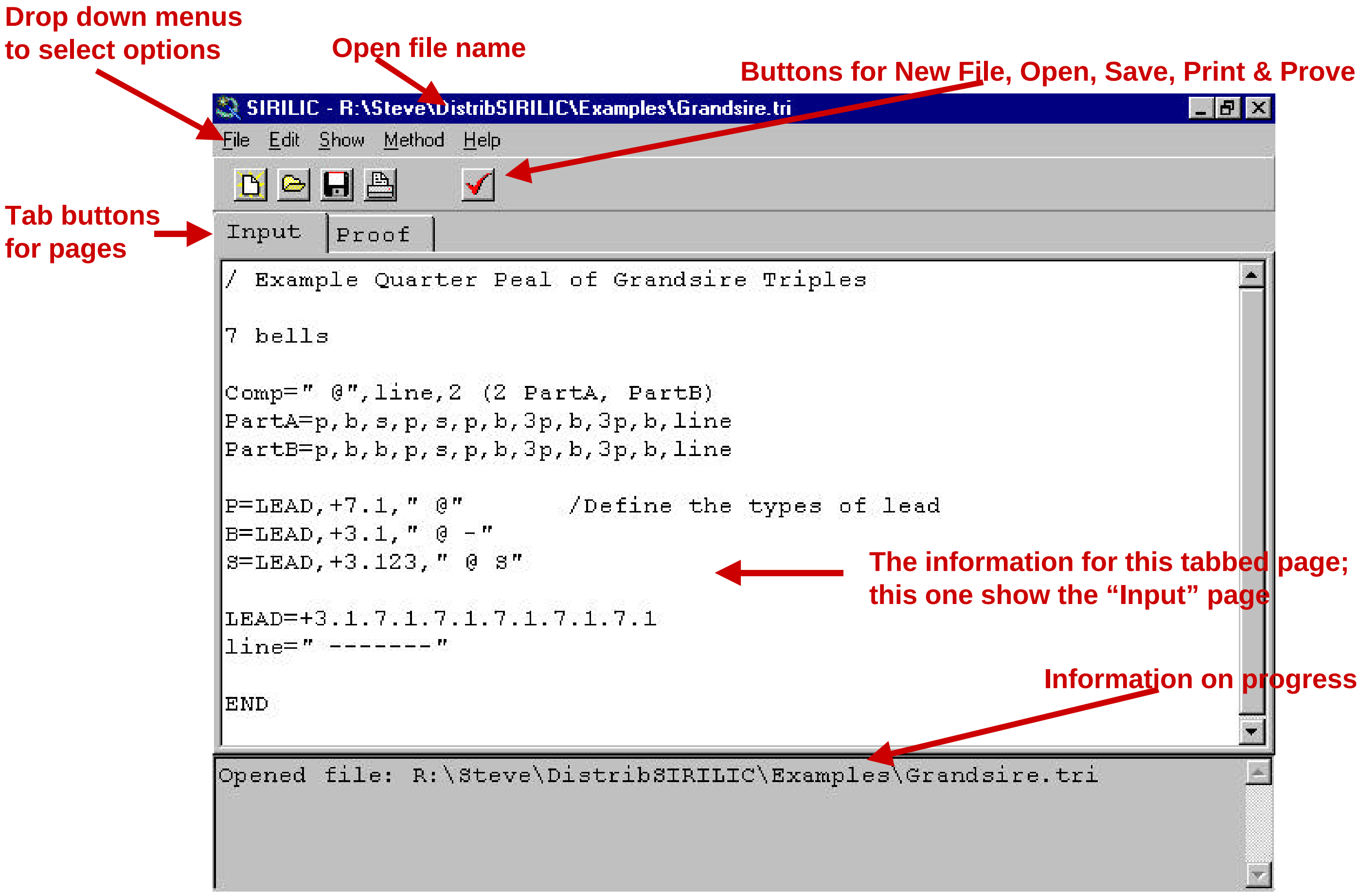
3. SETTING UP & RUNNING SIRILIC

Now that the files are unzipped, you will probably want to try it. Good! This guide aims to get you up and running with a prepared example and then you can try the fancy bits later. Double click on the SIRILIC.exe icon to run the program:

SIRILIC

The opening screen is pretty empty, and the "progress" information text at the bottom asks you to "Open an existing file or create a new one to start"; we will pick the first option. Either select File->Open->All from the menu or else click the folder icon:

All the icons come up with tips like "Open" if you point at them. The Open Composition File dialogue box appears, and shows the files in the "Examples" subfolder: select a file, for example Grandsire.tri by clicking once and selecting "Open", or by double clicking on Grandsire.tri. You should now see the following:



SIRILIC - R:\Steve\DistribSIRILIC\Examples\Grandsire.tri
File Edit Show Method Help
Input
Proof

Drop down menus to select options

Open file name

Buttons for New File, Open, Save, Print & Prove

Tab buttons for pages

The information for this tabbed page; this one show the "Input" page

Information on progress

```
/ Example Quarter Peal of Grandsire Triples

7 bells

Comp=" @",line,2 (2 PartA, PartB)
PartA=p,b,s,p,s,p,b,3p,b,3p,b,line
PartB=p,b,b,p,s,p,b,3p,b,3p,b,line

P=LEAD,+7.1," @"        /Define the types of lead
B=LEAD,+3.1," @ -"
S=LEAD,+3.123," @ S"

LEAD=+3.1.7.1.7.1.7.1.7.1.7.1
line=" -------"

END
```

```
Opened file: R:\Steve\DistribSIRILIC\Examples\Grandsire.tri
```

The screen shows two tabbed pages labelled "Input" and "Proof". Clicking the "Proof" and "Input" tab buttons selects the input or output of the proof process. If the text in the "Input" page is altered then the text on the "Proof" page turns red until you prove it again.

Three extra buttons have also appeared at the top of the SIRILIC page now that you have opened a file. Click on the red ticked "Prove" button:

The program proves the composition and automatically selects the "Proof" tab window to show the result of the proof process. If the Grandsire.tri file has not been fiddled with then it should say at the end of the composition listing:

```
1260 rows ending in 1234567
Touch is True
```

.. and then gives a list of the musical qualities of the composition.

You should now have SIRILIC working, but it is not yet calling the microSIRIL PROVE.COM engine. To select this, close SIRILIC and double click on the SIRILIC.ini file:

SIRILIC

You should see an option towards the end of the file:

```
[microSIRIL]
RUNmicroSIRIL=N
```

This needs to be changed to:

```
[microSIRIL]
RUNmicroSIRIL=Y
```

.. and then exit and save SIRILIC.ini

When you next run SIRILIC and open a composition file then you will be shown two extra tabbed pages labelled SIRIL and SIRILIC. When you next click on the red tick prove button these pages will be filled in with the results of the microSIRIL PROVE.COM engine and the equivalent compatible output from SIRILIC: these should be identical and you should have a progress message at the bottom of the page saying:

```
microSIRIL and SIRILIC agree
```

If you ever get the following alternative message (for 4 to 12 bells, the limits of microSIRIL):

```
****** microSIRIL and SIRILIC DIFFER ******
```

then I have a problem! All problems that I know about are posted on my web page www.noyes.org.uk ; if it isn't there then please would you let me know, and e-mail a copy of your input file. You should get identical output on true compositions, on false compositions and on input files with errors.

The Complete SIRILIC.ini file should now look like:

```
[SIRILIC]
MaxNumberBells=16                        .. between 12 and 16,
Working=.\Examples                       .. the location of your source files; define as you wish

[microSIRIL]
RUNmicroSIRIL=Y                          .. the Y/N option to run microSIRIL
Executable=.\microSIRIL\PROVE.com        .. the location of the microSIRIL PROVE.COM engine
Methodlib=.\Method Library               .. the location of the microSIRIL method libraries
OpenLibraryWith=Notepad                  .. the program used to open method library files
```

The [SIRILIC] section defines parameters for the SIRILC proof program; change the path specified for "Working" if you would prefer the file open dialogue to open in a different folder as the default.

The [microSIRIL] section defines the location and options on running PROVE.COM, and the location of the microSIRIL method libraries (see the link on my web page www.noyes.org.uk to find a current copy of these). The last line specifies the path of the program used to open the method library text file; for example you might want to change it to:

```
OpenLibraryWith=C:\Program Files\Windows NT\Accessories\wordpad.exe
```

4. THE microSIRIL LANGUAGE (with two special interpretations for SIRILIC)

The easiest way to describe the syntax is to pick apart an example, such as the Grandsire Triples used in section 3:

```
/ Example Quarter Peal of Grandsire Triples

7 bells

Comp=" @",line, 2 (2 PartA, PartB)
PartA=p,b,s,p,s,p,b,3p,b,3p,b,line
PartB=p,b,b,p,s,p,b,3p,b,3p,b,line

P=LEAD,+7.1," @"      /Define the types of lead
B=LEAD,+3.1," @ -"
S=LEAD,+3.123," @ S"

LEAD=+3.1.7.1.7.1.7.1.7.1.7.1
line=" -------"
END
```

Note that none of the input is case sensitive, except for output text that has been put between double quotes.

The first line has a "/" at the start, indicating that what follows is a comment line and not part of the proof instructions.

The first command line must always specify the number of bells, in this case 7 bells. The file should also finish with the command END, although it doesn't actually matter if this is omitted.

The next line starts Comp= and defines a name for the composition to be proved, in this case Comp. The name Comp is defined in terms of other symbols to the right of the equals sign, each defined later in the file, and separated by commas. Only the first name found in the file, "Comp" in this example, will be proved, all later lines can be in any order.

The first element defining Comp is `" @"`, which tells SIRILIC to print out a string; the proof isn't much use without any output, so strings in double quotes control what you will see. The special @ character means "the current change"; see the later section 4.1 on "Formatting Text Output". Comp then calls up line which is defined as a string of dashes later in the file, and then Comp moves on to `2 (2 PartA, PartB)` that defines the composition itself.

The composition is a 6 part composition, defined as a line of plain, bobbed and singled leads in PartA; however, in parts 3 & 6 you need to call a single instead of a bob using the similar calling defined as PartB. The name Comp reflects this structure by calling up 2 lots of 2 PartA and a PartB; the 2 outside the brackets doubles up the calling for the two halves of the composition. In all you get: PartA, PartA, PartB, PartA, PartA, PartB. You could also use 2*PartA if you prefer to indicate multiple use of a name by a star.

Looking at the definitions of PartA and PartB, they use the names P, B, S and line; line has already been dealt with. The definitions of P, B and S are very similar and represent the plain, bobbed and singled leads. Taking B as an example, it first uses LEAD that defines the first 12 changes of the place notation for a lead of Grandsire Triples, and then adds on a +3.1 that is the last two changes of the bobbed lead: a total of 14 changes. Finally, the " @ -" allows the printout of each lead end .. note the special @ character again.

Place notations start with a + but more usually start with an & that indicates a symmetrical lead method. This saves on typing by assuming that the place notation of the second half lead is the reverse of the first. So, a lead of St Clements Minor could be:

StClem=&x16x36x36,+12   or: StClem=+x16x36x36x36x16x12

The place notation indicates the places made as you go from one change to the next, with x or – indicating all pairs swapping; a dot is used to separate two consecutive changes that both have places made in them. The digits 1 to 9 indicate places up to 9ths; zero for tenths, eleven/twelve are ET and 14 thru 16 are ABCD. You can also let the program assume places made at lead or at the back in any change with other places, so London S Minor could be reduced to:

London=&3x3.4x2x3.4x4.3,2

Lines of text can get quite long. You can either break them up into chunks by defining each chunk as a separate name in its own right, or you can use continuation lines with a comma ending the previous line, e.g.

```
Comp=3p,b,
        4p,s,5p,
        s,6p,b,p              .. is the same as:        Comp=3p,b,4p,s,5p,s,6p,b,p
```

Names can be formed from any combination of letters, digits and the three characters % ! or -, but must start with a letter. Names are NOT case sensitive. Spaces and tabs are ignored; tabs can be useful for lining things up nicely, both for the input and in the output strings.

4.1 FORMATTING TEXT OUTPUT

Text strings enclosed in “” are used to define output, as already seen in the example above. Some characters have special meanings as follows:

@ displays the current change
# displays the number of changes so far
$ displays the number of duplicated changes that have been found so far
\ displays the following character, even if it is a @ # $_ or \. It is also used just before the closing quote to suppress the usual line feed.
_ turns on/off underlining from this point onwards on the current output line (not interpreted in microSIRIL), so for example, “You can _underline this bit_ and not the rest, and _ still prints the character”
$$ halts further processing of the input file

4.2 PREDEFINED COMMANDS FOR TAKING SPECIAL ACTIONS

Some predefined names have been set up and are called by default at the end of the proving process, to say if the composition is false, comes round true, or fails to come round at the end of the defined composition. They are:

```
TRUE       = "# rows ending in @", "Touch is True$$"
NOTROUND   = "# rows ending in @", "Is this Ok? "
FALSE      = "# rows ending in @", "Touch is False in $ rows"
```

These names, and the other special names defined in this section, may be redefined (but only once each) in your input file if you want different output or different actions taken.

The other special predefined commands have been set up as follows, and again you can change them by supplying your own. The first shows the output & action to take when a duplicate change is found. The next two control what to do when rounds is found before the end of the composition, and provide a way of performing some action at every change; for example, setting EVERYROW=”# @” will create a numbered list of every change in the composition.

```
CONFLICT   = "# rows ending in @", "Touch not completed due to False row$$"
ROUNDS     =
EVERYROW   =
MUSIC      = "DEFAULT"
```

Note the special $$ in CONFLICT that halts processing at the first false change! The last of these, `MUSIC=..`, has a special interpretation for SIRILIC, but has taken the same format as any other name definition in microSIRIL so that it is still a valid line. It creates a list of the musical changes found at the end of the proof; giving statistics for odd changes (usually handstroke) and even changes (usually backstroke) to help with suitability for half muffled ringing. For the “DEFAULT” it searches for:

Combination Roll Ups .. the changes with the bells above 6th as rounds and two out of 4,5,6 in 5-6.
Queens, Tittums, Kings
Whittingtons (taken as 3124975680 on ten, etc.)

You can redefine your own, and specify runs of bells off the back and front, for example, on 8 bells you might want:

```
MUSIC="DEFAULT", "?468", "876?"
```

Which would find all the xxxxx468’s and 876xxxxx’s as well as the default set of changes.

On 10 bells, MUSIC can also take a predefined parameter “7890s” that finds all the combinations of 7-8-9-0 off the front (i.e. all combinations of 7-8-9-0 in 1-2-3-4), for example:

```
MUSIC="DEFAULT","7890s"
```

Note that the default definition of `CONFLICT="..$$”` will stop the processing the first time that a repeated change is found. SIRILIC will automatically check blocks that repeat changes (such as 1440 or 5040 blocks of Minor for a peal, or 1260s of Minor for a quarter) by setting the string so that it does not terminate. A useful setting is:

```
CONFLICT=
```

SIRILIC then produces statistics on the number of different changes rung and the maximum number of times any change was repeated. See the examples of 5040 Plain Bob Minor & 1280 Plain & Little Minor provided in the Examples folder.

4.3 FINDING WHICH BELLS RING WHICH LEADS OF A METHOD

Is your composition “all the work”? SIRILIC can list the number of times each bell rings each place bell of a method if the method is defined in a special way. This is also useful for finding out which leads you will actually have to learn if you have been assigned a bell in a peal. The following snippet gives a typical example of method definition for Spliced Surprise Royal:

```
LON = "@ London",       "[LON]", &30x30.14x12x30.14x14.50.16x16.70.14.58.14.90
CAM = "@ Cambridge",    "[CAM]", &x30x14x1250x36x1470x58x16x70x18x90
YOR = "@ Yorkshire",    "[YOR]", &x30x14x50x16x1270x38x14x50x16x90
```

The “[LON]” etc. tells SIRILIC to look at the position of each bell at the current change (the first of the lead in the above), and make a list under LON. At the end of the proof the following statistics will be listed out:

- The number of times that LON was used by the composition (i.e. the number of leads of LON rung)
- The number of changes of LON rung
- The number of times each bell rang each place bell for LON
- The number of times the method was changed between LON, CAM and YOR etc.

It has been quite interesting checking the claimed “com” and “atw” etc in other people’s compositions over the years!

Example of input & output follow at the end of this documentation, and a 1280 of spliced Plain & Little Minor is provided in the Examples folder.

6 THE SIRILIC PROGRAM WINDOW

Section 3 introduced the ideas of pressing the tab buttons to switch between the pages of input/output. It also used a couple of the following buttons:

🗋 Open a new input file

📂 Open an existing file, opens the file selection dialogue box in the folder defined as Working in SIRILIC.ini

💾 Save the current input file

🖶 Print the text in the currently selected tabbed page

✓ Prove the composition

Note also that by clicking the right mouse button whilst the mouse pointer is within the “Input” tabbed page will pop up the display of line numbers for each line (useful if you get an error in line 57 and want to find out which is line 57). Right clicking again gets rid of them. Note that you can’t edit the input text whilst line numbers are displayed, so right click first.

The input may be edited in usual Windows fashion, so for example you can point and left click to position the cursor and type in new text. The usual cut/copy/paste commands are available from the Edit menu or by the usual keyboard shortcuts (CtrlX, CtrlC, CtrlV).

After you press the “Prove” button to check your composition for proof, any errors found in your input page are automatically shown on the “Proof” tabbed page. I have tried to make the error messages that appear here useful, and they try to identify what the program doesn’t like about the input file. The errors in the SIRIL and SIRILIC windows are the same ones that SIRIL reports (I hope!), and they should in fact be identical.

The pull down menus provide the following facilities:

File ->	New	Opens a new (unnamed) input file
	Open	Open an existing file, opens the file selection dialogue box in the folder defined as Working in SIRILIC.ini
	Save	Save the input file
	Save As…	Save the input file as..
	Prove	Run the proof for the input
	Explorer…	Open Windows Explorer
	Print…	Print the current tabbed page
	Printer Setup…	Open the printer setup dialogue
	Exit	Exit from SIRILIC
Edit ->	Del	Delete the highlighted text
	Cut	Cut out the highlighted text to the paste buffer
	Copy	Copy the highlighted text to the paste buffer
	Paste	Paste from the paste buffer
	Select All	Select all text in the current tabbed page
Show ->	Musical Changes	Checked, lists for musical changes (on 6 bells or more) when proved
	Work for Each Bell	Checked, enables place bell output list when proved
	microSIRIL Output	Checked, enables the SIRIL output tabbed page
Method ->	Search microSIRIL Libraries	Opens a dialogue to select a method library file
Help ->	About	.. you can try these for yourself!
	Predefined Names	
	Special Printable Characters	
	Example	

Note that “Help->About” will tell you which version of SIRILIC you have, so that you can check it against the current version on my web page www.noyes.org.uk. This User Guide is dated on each page, and the date on the version on the web page will tell you if it is the current verson.

6 PROBLEMS & FAULT FINDING

Please check that you are running the current version of SIRILIC before you moan to me about errors; you can check the version of your program by using Help->About, and then compare it with the version on my web page at www.noyes.org.uk. If your version is up to date and your error isn't on the problem list page (also on the web page) then it is my problem and not yours, so please let me know what it is! Send an e-mail to steve@noyes.org.uk with a description of the problem or include the input file that produces the errors.

6.1 I CAN'T UNZIP THE SIRILIC.ZIP FILE THAT I DOWNLOADED FROM THE WEB

Do you have a copy of WinZip? It is available on the cover discs on all sorts of computer magazines, or try web sites such as www.tucows.com and do a search for WinZip to find the download.

6.2 THE microSIRIL PROVE PROGRAM WILL NOT RUN

Check that the path for PROVE.COM is specified correctly in the EXECUTABLE=.. line of SIRILIC.ini One of the first things that SIRILIC does if you have set RUNmicroSIRIL=Y is to check that PROVE.COM exists at the specified location, and if not then it pops up a warning message. The value given in the SIRIIC.ini file that was in Zip file that you downloaded was:

```
Executable=.\microSIRIL\PROVE.COM
```

6.3 THE mircoSIRIL PROVE PROGRAM WILL NOT ACCEPT A HYPHEN AT THE END OF A METHOD NAME

There are actually TWO versions of PROVE.COM; I checked with Andrew Craddock to find out the differences between them. The file sizes and dates of creation are:

PROVE.COM	4138 bytes	Created 1-Jan-1980

and, *despite the apparent creation date* the *earlier* version is:

PROVE.COM	4126 bytes	Created 20-Sept-1987

The 4138 file has the additional ability to accept a hyphen in a method name, usually used to indicate a bobbed lead, and a % to indicate a singled lead. The 4138 file is distributed as part of the zip file you downloaded from my web page.

6.4 I CAN'T EDIT THE INPUT TEXT

It is not possible to edit the input text while line numbers are displayed (the text should also be blue when line numbers are displayed); click the right button on the input page to get rid of them and you should be OK.

6.5 I CAN'T SAVE THE FILE I HAVE EDITED

This is probably because the file is "Read-only". To change it, right click the file, select "properties" and make sure that "Read-only" is unchecked.

6.6 PROVE.COM SOMETIMES LOCKS UP WHEN TRYING TO PROVE A COMPOSITION

I don't know why either, but I didn't write PROVE.COM. It only seems to happen on files with an error; to recover, close the DOS window to kill PROVE.COM. For example, the following locks up PROVE.COM:

```
6 BELLS                    Even if you open a DOS window & use:
PEAL=2*PART                      $ PROVE < THISCOMP.TXT
PART=c1                    then it still locks up!
c1=p,b,p
c1=p,b,p                   This line is the error; a duplicated name
P=PB,+12
B=PB,+14," @ -"
PB=&x1x1x1
```

6.7 ERRORS IN PRV$In.txt & PRV$Out.txt FOR PROVE.COM

What should happen .. the SIRILIC program creates an input file called PRV$In.txt in the same directory as the PROVE.COM program, spawns PROVE.COM and directs the output to PRV$Out.txt. This output is then displayed in the SIRIL pane of SIRILIC.

There are a number of reasons why there may be problems on this (incorrect folder name, file protection on the file or folder hierarchy and so on) but you can find a bit more about what is happening and where the process is going wrong by enabling the DEBUG mode in the SIRILIC.ini file. To enable this mode, add the following line into the [microSIRIL] section of SIRILIC.ini:

```
Debug=Y
```

For example, this section on my computer now looks like:

```
[microSIRIL]
RunmicroSIRIL=Y
Debug=Y
Executable=..\Msiril\Executables\PROVE.COM
Methodlib=R:\Delphi\Msiril\Executables\MSIRIL 1999\Method Libraries
```

Exit and save SIRILIC.ini, and when you next run the program a number of lines should appear in the progress window at the bottom of the page (you will need to scroll up/down to see them) that look something like:

```
============ START OF CODE TO SPAWN PROVE.COM ============
Try to save input file to = "..\Msiril\Executables\PRV$In.txt"
.. The file was created
Try to change current working directory to "..\Msiril\Executables\"
.. current directory changed
Clear the microSIRIL output window & setup status blocks to spawn process
.. complete & ready to spawn process
Spawn the prove engine with command line="PROVE.COM" < PRV$In.txt  > PRV$Out.txt
.. process spawned
Get exit process code and wait until STILL_ACTIVE goes false
.. spawned process has finished
Set the working directory back to "R:\Delphi\SIRILIC"
.. have reset working directory
Load the output file data into the window, from file "..\Msiril\Executables\PRV$Out.txt
ALL FINISHED
.. would now usually delete the PRV$Out.txt & PRV$In.txt but DEBUG mode leaves them for
inspection
============ END OF CODE TO SPAWN PROVE.COM ============
```

Note also that in the Debug mode the PRV$In.txt and PRV$Out.txt files are NOT deleted after PROVE.COM closes down.

Were the folder names correct? Look at the listing (similar to above) and check that the folder names that SIRILIC is trying to open actually exist.

So were these files both created correctly? Try opening them up & find out.

Was the command to spawn the files going wrong? Try opening an MS-DOS window, move to the directory with PROVE.COM in it and try executing the command line yourself with:

```
$ PROVE.COM < PRV$In.txt > PRV$Out.txt
```

If it still goes wrong then try omitting the last parameter to send the output to the screen with

```
$ PROVE.COM < PRV$In.txt
```

Hopefully, these steps should shed a bit of light of what is going wrong. If not, highlight the output lines you got in the progress window, press Ctrl C to copy them, and past then into an e-mail to me!

7 EXAMPLE INPUT AND OUTPUT

An example of the input illustrating these features is:

```
/ 5040 3 SPLICED S. ROYAL (CYL), 40com, atw,
/ almost a regular 3 part  - Stephen P Noyes

10 BELLS

PEAL=PART1,2 PART2

/ Define a few lines that are continued across a line

PART1=C1,C2,C3,C6,
      C7,C8,C9,LINE
PART2=C1A,C2,C3A,C6,
      C7,C8,C9,LINE


/ Define the musical changes to search for

MUSIC="DEFAULT","?67890"

C1=5(YOR,P),LON,B
C1A=YOR,P,LON,P,LON,B
C2=8(LON,P),LON,B
C3=LON,P,CAM,B,LON,P
C3A=LON,P,CAM,B,YOR,P,YOR,P,YOR,P,YOR,P
C6=LON,P,CAM,B,YOR,P,YOR,P,YOR,P,YOR,B
C7=LON,P,CAM,P,YOR,P,YOR,P,YOR,P,YOR,B
C8=CAM,P,CAM,P,CAM,P,CAM,B,CAM,P,CAM,P,CAM,P,CAM,P,CAM,B
C9=LON,P,CAM,B,LON,B
LINE="------------"

P = +12,"  @"
B = +14,"- @"

/ Define the methods, with the name in square brackets so that we have an
/ “all the work” check

LON="@ London","[LON]",&30x30.14x12x30.14x14.50.16x16.70.14.58.14.90
CAM="@ Cambridge","[CAM]",&x30x14x1250x36x1470x58x16x70x18x90
YOR="@ Yorkshire","[YOR]",&x30x14x50x16x1270x38x14x50x16x90
END
```

The output resulting from the previous input is the usual microSIRIL stuff, then:

```
<snip>
5040 rows ending in 1234567890
Touch is True

Touch contains:
               Odd     Even      Total
xxxx567890 =    0  +     7    =     7
xxxx657890 =    0  +     5    =     5
xxxx467890 =    0  +     3    =     3
xxxx647890 =    0  +     6    =     6
xxxx457890 =    0  +     6    =     6
xxxx547890 =    0  +     5    =     5
1357924680 =    0  +     0    =     0
3124975680 =    0  +     0    =     0
1627384950 =    0  +     0    =     0
9753124680 =    0  +     0    =     0
xxxxx67890 =    0  +    32    =    32

CRU        =    0  +    32    =    32

NumberLeads=126

[LON]  48 leads   1920 changes

      1pb 2pb 3pb 4pb 5pb 6pb 7pb 8pb 9pb 0pb
  1:  48   -   -   -   -   -   -   -   -   -
  2:   -   6   3  15   3   9   3   3   3   3
  3:   -  10   6   8   9   3   3   3   3   3
  4:   -   7   9   4   5   8   3   5   3   4
  5:   -   6   8   5   5   8   3   6   3   4
  6:   -   7   8   4   5   8   3   7   3   3
  7:   -   3   5   3  10   3  15   3   3   3
  8:   -   3   3   3   3   3   3  15   5  10
  9:   -   3   3   3   5   3  10   3  15   3
  0:   -   3   3   3   3   3   5   3  10  15


[CAM]  39 leads   1560 changes

      1pb 2pb 3pb 4pb 5pb 6pb 7pb 8pb 9pb 0pb
  1:  39   -   -   -   -   -   -   -   -   -
  2:   -  12   6   3   3   3   3   3   3   3
  3:   -   3   6   3   6   3   9   3   3   3
  4:   -   4   5   7   6   3   5   3   3   3
  5:   -   4   5   7   6   3   5   3   3   3
  6:   -   4   5   7   6   3   5   3   3   3
  7:   -   3   3   3   3   3   3   3  15   3
  8:   -   3   3   3   3  15   3   3   3   3
  9:   -   3   3   3   3   3   3   3   3  15
  0:   -   3   3   3   3   3   3  15   3   3

[YOR]  39 leads   1560 changes

      1pb 2pb 3pb 4pb 5pb 6pb 7pb 8pb 9pb 0pb
  1:  39   -   -   -   -   -   -   -   -   -
  2:   -   3   6   3   3   6   6   3   3   6
  3:   -   2   9   7   1   2   2   7   7   2
  4:   -   6   1   3   7   4   3   6   6   3
  5:   -   6   2   3   5   5   4   5   5   4
  6:   -   4   3   5   5   4   4   5   5   4
  7:   -   8   1   1   8   8   3   1   1   8
  8:   -   1   8   8   1   1   8   3   1   8
  9:   -   1   8   8   1   1   8   8   3   1
  0:   -   8   1   1   8   8   1   1   8   3

40 changes of method
```