

	<i>Document Name</i>	<i>Document Class</i>	<i>Date</i>	<i>Version</i>
	G1G2 Converter	Instruction	June 20 2011	1.3
	<i>Author</i>	<i>Document Nr</i>		<i>Page</i>
	Mattias Ericsson	4-100-12		1 (11)

GOOP1 and GOOP2 Converter Tool Instruction

	<i>Document Name</i>	<i>Document Class</i>	<i>Date</i>	<i>Version</i>
	G1G2 Converter	Instruction	June 20 2011	1.3
	<i>Author</i>	<i>Document Nr</i>		<i>Page</i>
	Mattias Ericsson	4-100-12		2 (11)

Contents

1	Document Change History	2
2	Documents and Abbreviations	2
2.1	Reference Documents	2
2.2	Abbreviations	2
3	Installation.....	3
4	Introduction.....	3
5	Convert Process	4
6	Finishing Conversion and Cleaning Up.....	7
7	Troubleshooting	10

1 Document Change History

Version	Comment	Author
0.1	First version.	Mattias Ericsson
0.2	Fixed spelling errors.	Mattias Ericsson
1.0	Updated after review.	Mattias Ericsson
1.1	Added information about the prepare tool.	Mattias Ericsson
1.2	Added more info about llb.	Mattias Ericsson
1.3	Updated document with information regarding "ExcludeClasses.ini" file.	Mattias Ericsson

2 Documents and Abbreviations

2.1 Reference Documents

Ref	Doc Nr	Category	Document Name	Content

2.2 Abbreviations

GOOP G Object-Oriented Programming

	Document Name	Document Class	Date	Version
	G1G2 Converter	Instruction	June 20 2011	1.3
Author	Document Nr		Page	
Mattias Ericsson	4-100-12		3 (11)	

3 Installation

1. Install G# Framework >= version 1.3.1.
2. Run the G1G2Converter_Installer.vi.
3. Restart LabVIEW.
4. Make sure *_goop2.llb* is installed in *<LabVIEW>\vi.lib\addons*

4 Introduction

The G1G2 converter is a tool to convert GOOP1 and GOOP2 classes to G#. There are a few things to consider when converting to G#. For a large system it is not possible to convert the entire system at once. It is necessary to convert the system piece by piece. Due to this it is important to analyze the system dependencies before converting. Convert about 10 classes at the time. Use the following rules of thumb when converting the system, ordered in priority:

1. Start with the simple classes that are aggregated by the other classes. In Figure 1, denoted *Aggregate1* and *Aggregate2*.
2. Convert entire inheritance class hierarchies at the same time if possible. In Figure 1, denoted *Class1*, *Class2* and *Class3*.
3. Convert the base class, before the subclasses. Make sure all aggregating classes are converted before.

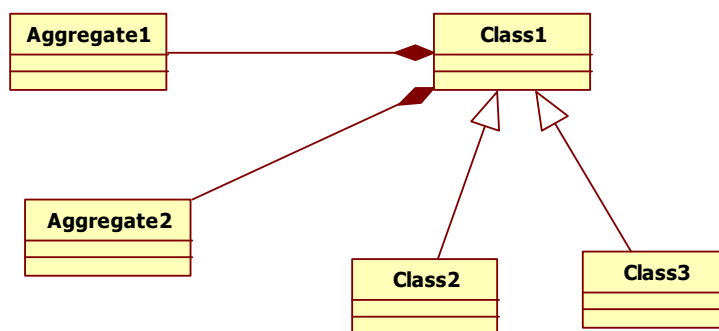


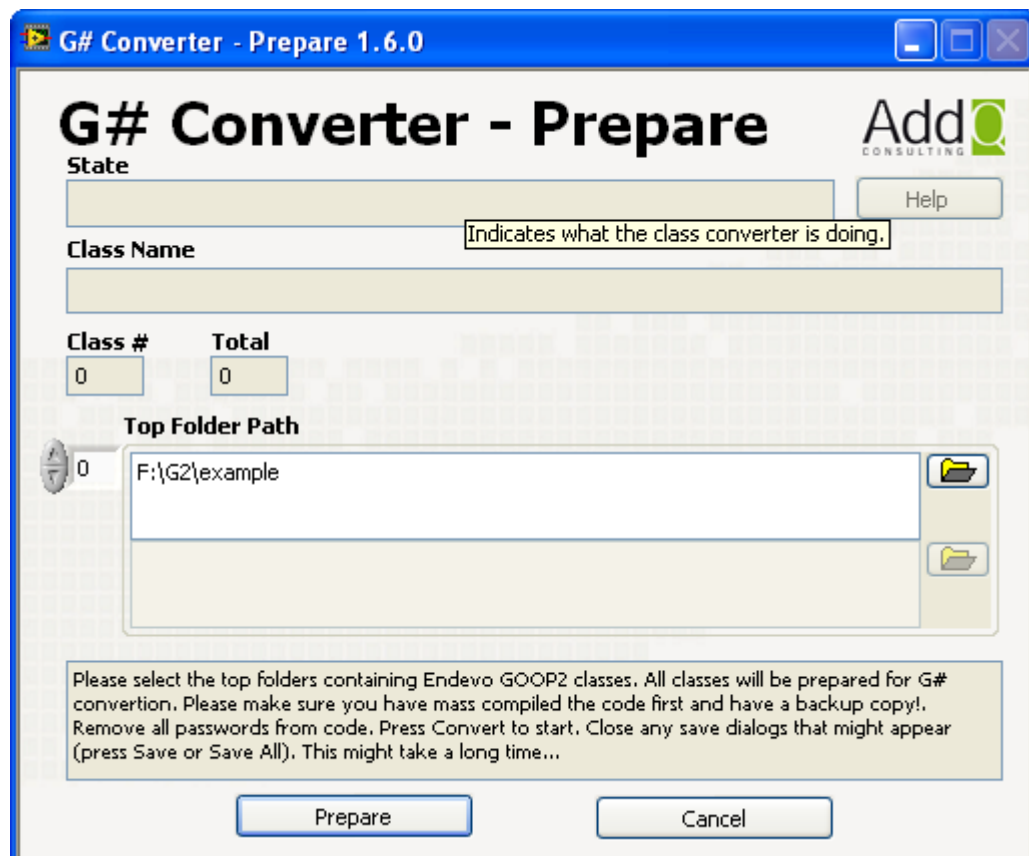
Figure 1. Typical class diagram.

When converting classes, a temporary project and a log file is created in a folder: *<Temporary Directory>\TempProject*. These are overwritten when a conversion is started. Note that all GOOP1 and GOOP2 methods will keep their class prefix in the name. If the prefix were to be removed there would be problems with broken links in your application. Only virtual subclass methods will be renamed, since it is necessary for the LabVIEW dynamic dispatch to work correctly.

	Document Name	Document Class	Date	Version
	G1G2 Converter	Instruction	June 20 2011	1.3
	Author	Document Nr		Page
	Mattias Ericsson	4-100-12		4 (11)

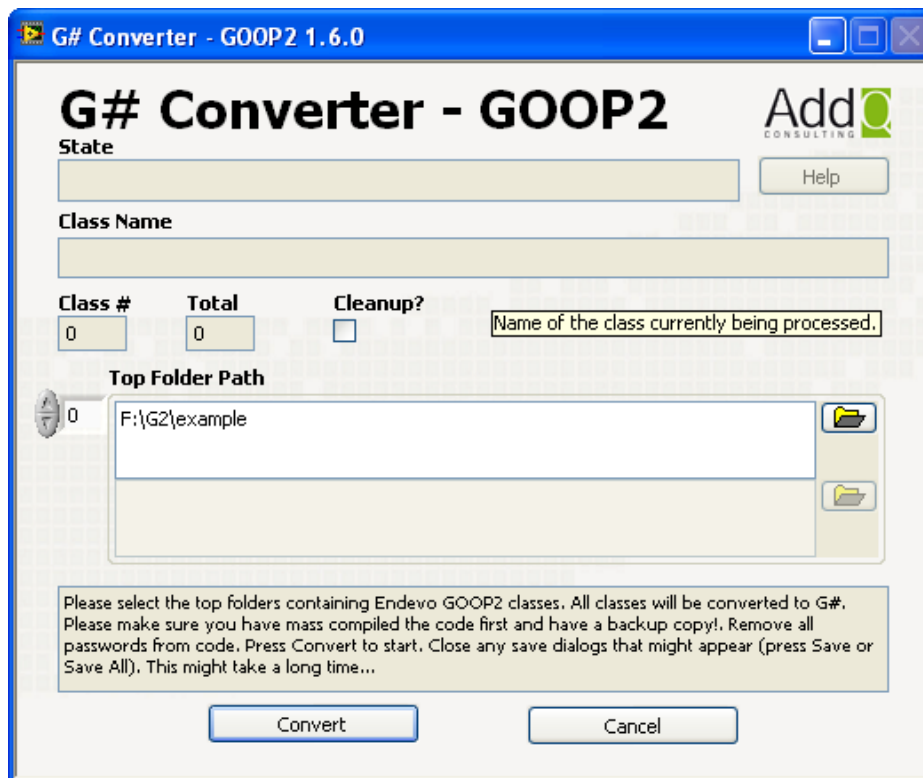
5 Convert Process

1. Decide which classes to convert according to the discussion in previous section.
2. Before starting make sure:
 - You have a backup copy of the code to convert
 - You have access rights to the folder and the code is writable
 - Use a local disk on the computer and not a server location
 - `_goop2.llb` is located in the `<LabVIEW>\vi.lib\addons`
3. Start the *G# Converter – Prepare tool*. This will mass compile, fix incorrect use of private method and analyze the code. It is not necessary if the GOOP2 classes don't contain any "forbidden" use of private methods to run the prepare tool.
4. Select a top folder containing the classes or select the classes one by one. Press *Prepare*!



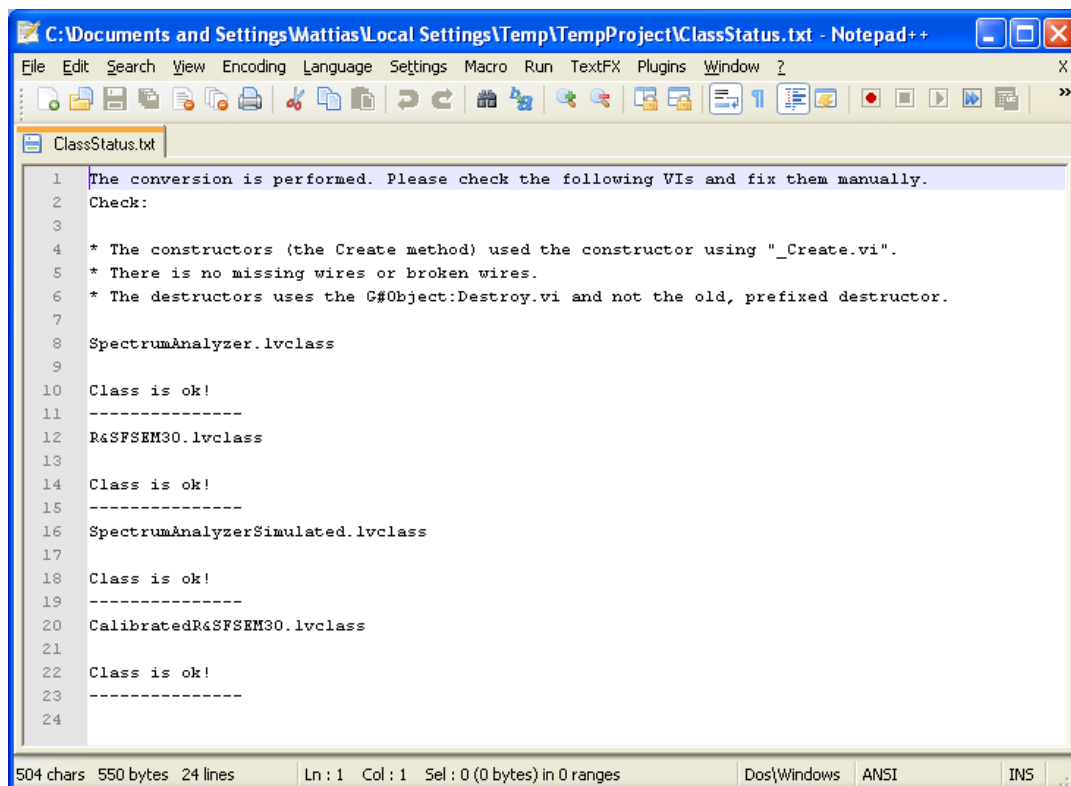
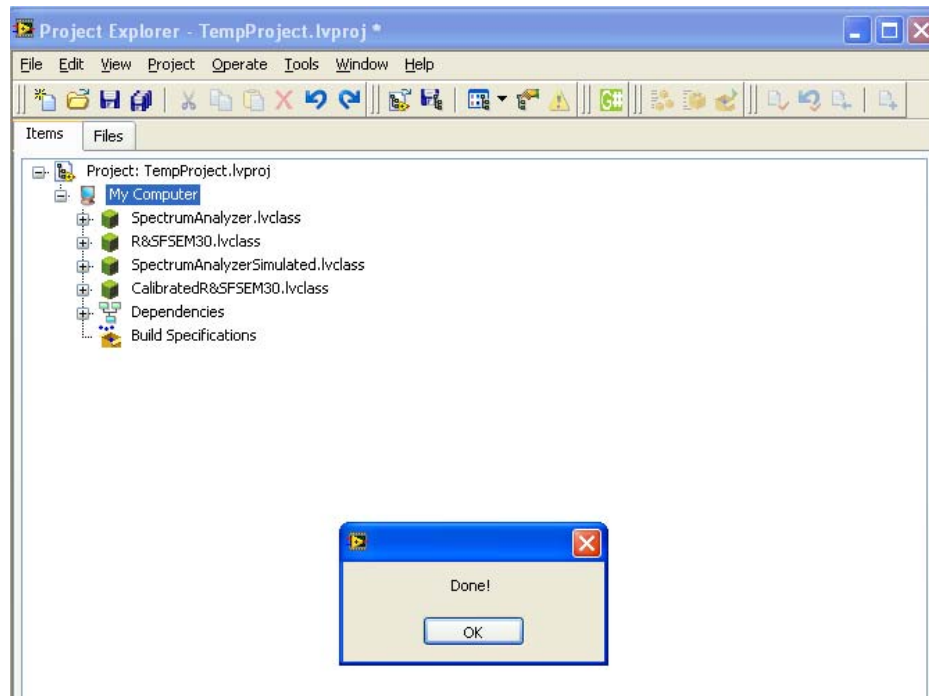
	Document Name	Document Class	Date	Version
	G1G2 Converter	Instruction	June 20 2011	1.3
	Author	Document Nr		Page
	Mattias Ericsson	4-100-12		5 (11)

- When finished, a *ClassStatus.txt* file is generated and pops up in your default text editor, showing the status of the class and which VIs that needs to be fixed. Make sure that there is **no missing sub-VIs** in the code structure to be converted. It is ok if it isn't executable. **Restart LabVIEW** when done!
- Start the *G# Converter – GOOP2...* (Or the GOOP1) tool from the LabVIEW Tools menu.
- Select a top folder containing the classes or select the classes one by one. It is possible to place a file call *ExcludeClasses.ini* with a list of class name (just one class name on one row) in the selected top folder path to ignore when converting. Press *Convert* to start converting.



- Classes will now be converted. A temporary project will now be created and the converted classes will be created in this project. After a while there will be some Save Dialogs asking you to save some VIs, press *Save* or *Save All* when these occur. (These VIs are about to leave memory).
- When the conversion tool finishes its execution, a pop of dialog will display the text "Done". The temporary project is left open. A *ClassStatus.txt* file is generated and pops up in your default text editor, showing the status of the class and which VIs that needs to be fixed. Fix the broken VIs by opening from the temporary project and correct the problem.

<i>Document Name</i>	<i>Document Class</i>	<i>Date</i>	<i>Version</i>
G1G2 Converter	Instruction	June 20 2011	1.3
<i>Author</i>	<i>Document Nr</i>	<i>Page</i>	
Mattias Ericsson	4-100-12	6 (11)	



10. Done! Continue with the next set of classes and repeat from step 2.

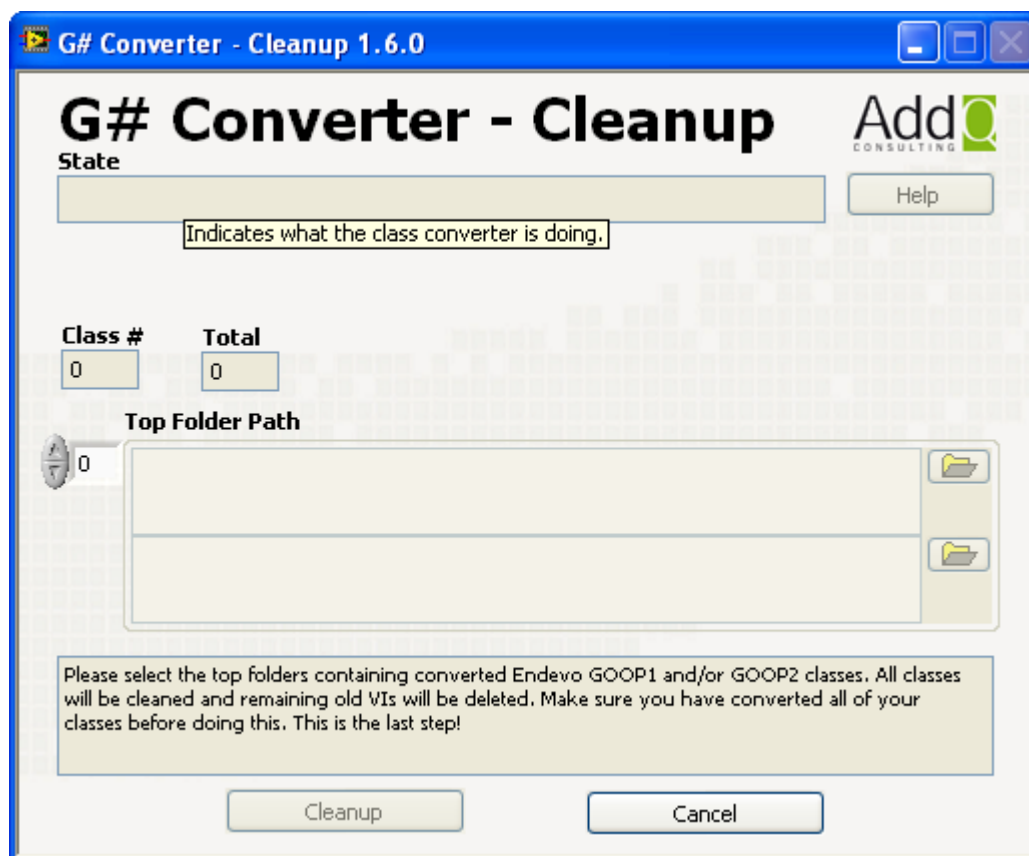
	Document Name	Document Class	Date	Version
	G1G2 Converter	Instruction	June 20 2011	1.3
Author	Document Nr		Page	
Mattias Ericsson	4-100-12		7 (11)	

6 Finishing Conversion and Cleaning Up

When the conversion is completed, the code needs to be cleaned up by removing the last remains of the GOOP1 and GOOP2 classes. When a class is converted, old files like *Object Reference.ctl*, *GOOPKernel.vi* and the prefixed *ObjectAttributes.ctl* and *ClassAttributes.ctl* will still be in the class, but not used by the actual class. However, if there are other classes that have not yet been converted and they use this class by aggregation or inheritance, these classes will still be linked to these old VIs and controls. Therefore these could not be removed until all classes have been converted.

When done with all class conversions, there are two things remaining to do:

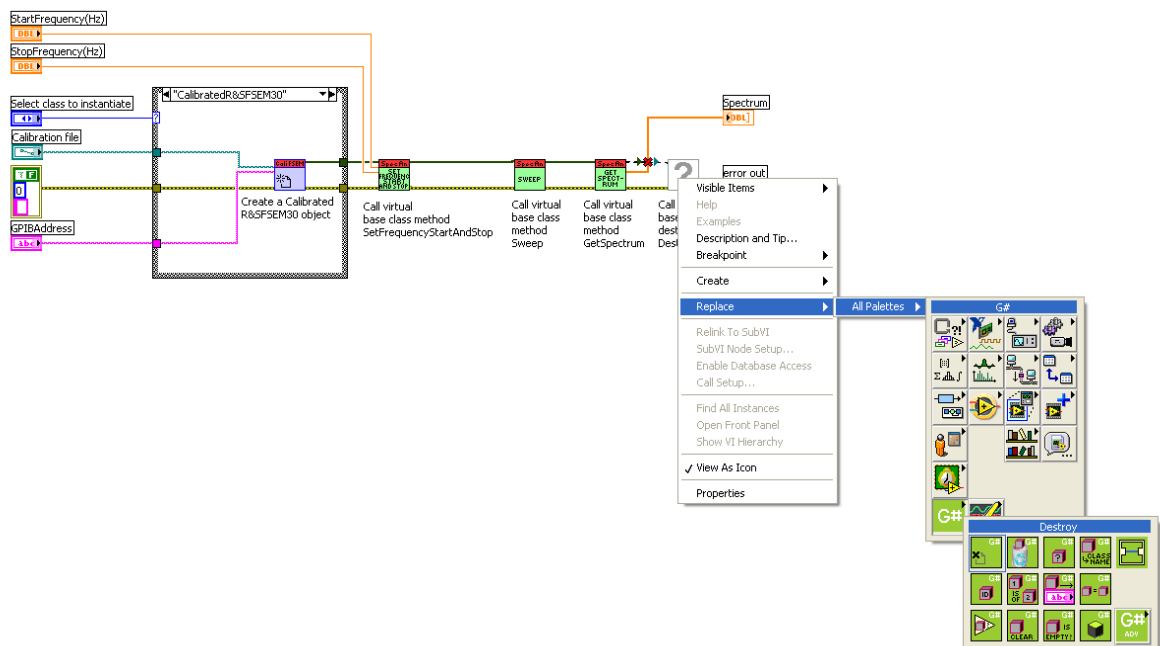
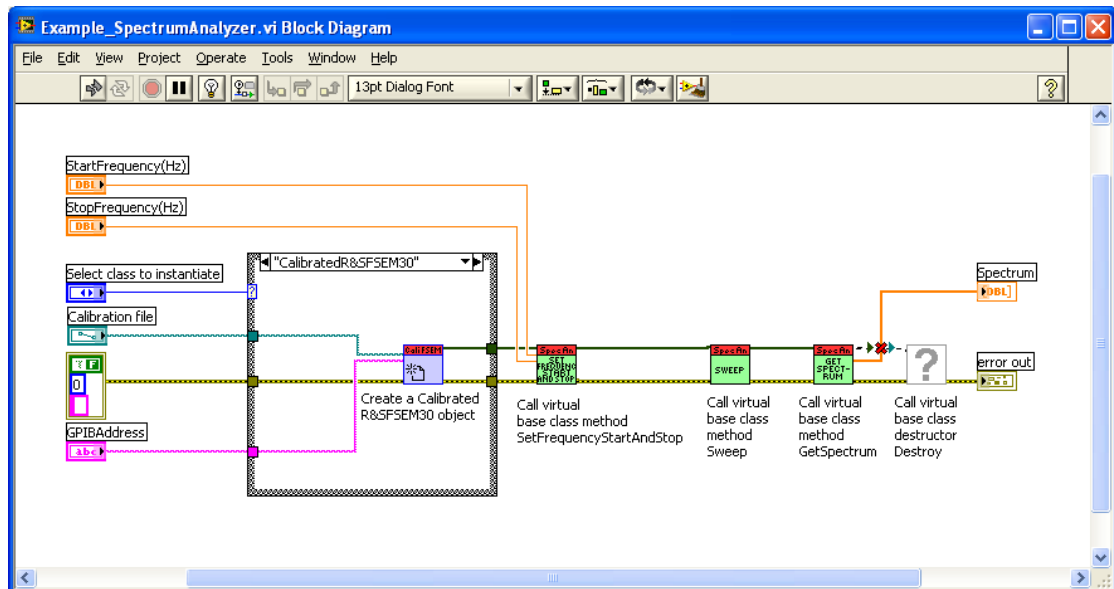
1. Use the tool *G# Converter - Cleanup*, from the Tools menu in LabVIEW; it will remove all unused old GOOP1 and GOOP2 utility VIs and controls.



2. Update and fix all users of the classes, like top-level applications. These are users of the class and needs to be fixed. Typically there are three things to do:
 - a) Check that all constructors (the *Create* method) are correct. If you used `<space>` and not `"_"` as a delimiter in your old classes, this will occur. G# only allows `"_"` as delimiter.

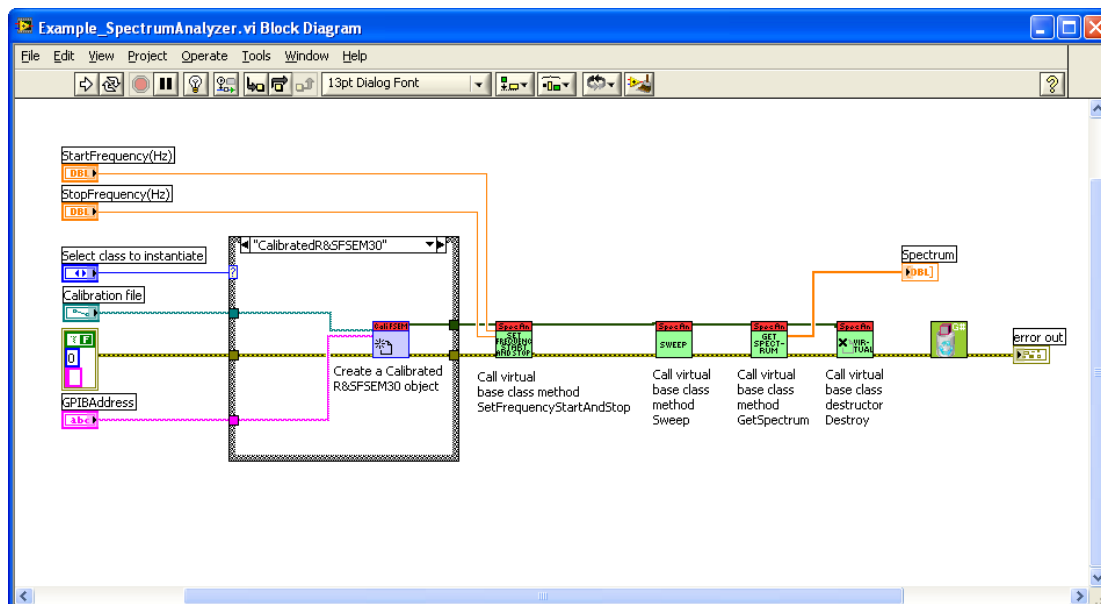
Document Name	Document Class	Date	Version
G1G2 Converter	Instruction	June 20 2011	1.3
Author	Document Nr	Page	
Mattias Ericsson	4-100-12	8 (11)	


- b) Check the destructor. In G# all destructors are called *Destroy.vi* and inherit from the generic *G#Object* destructor. Updated and replace all destructor by right-clicking on a broken destructor (or missing as shown below) and select the G# menu and choose *Destroy*. LabVIEW will automatically find the correct destructor by dynamic dispatch.



<i>Document Name</i>	<i>Document Class</i>	<i>Date</i>	<i>Version</i>
G1G2 Converter	Instruction	June 20 2011	1.3
<i>Author</i>	<i>Document Nr</i>	<i>Page</i>	
Mattias Ericsson	4-100-12	9 (11)	

- c) Add a G# *Garbage Collector* as the last VI in the top-level application (located next to the *Destroy* in the G# menu). This is not absolutely necessary, but a good G# coding principle.




	<i>Document Name</i>	<i>Document Class</i>	<i>Date</i>	<i>Version</i>
	G1G2 Converter	Instruction	June 20 2011	1.3
	<i>Author</i>	<i>Document Nr</i>		<i>Page</i>
	Mattias Ericsson	4-100-12		10 (11)

7 Troubleshooting

If the tool fails to convert or reports an error there are a few things to consider. There is a log file generated in the same folder as the temporary project.

- If using llb, sometimes Windows prevents LabVIEW to rename the llb and then create a folder with the same name (including the .llb). This could occur if restoring a backup etc. Try the following steps:
 1. Restart LabVIEW
 2. In Windows explorer, rename the llb to a dummy name and then back to the original llb name.
 3. Retry conversion.
- LabVIEW crashes (disappears) – make sure the code is mass compiled before trying to convert. Look in the log file generated in the *<Temporary Directory>\TempProject\TempProject.txt*. Check where the tool crashed. It will give you a hint of which class (or VI) that caused the crash. You could also try to convert more classes (or less) at the same time.
- LabVIEW hangs – probably converting too many classes at the same time. Try to convert smaller pieces of code at one time.
- After conversion there are missing *GetAttributes*, *GetAttributesToModify* or *SetModifyAttributes* in some methods. Probably there have been forbidden usage of private methods outside the class. Just replace these with the correct one from the converted class. (Found in the *utils* folder). You could use the *G# Converter – Prepare* tool to fix these problems before conversion.
- If you have used the same name of a method in a base class and a subclass, and the base class method was not set as a virtual method in GOOP2, you might end up with non-executable code in G#, since LabVIEW classes *doesn't allow static override*. Either set the base class method as dynamic dispatch (reference in and out, set as dynamic dispatch on the connector pane and assure identical connector panes for both methods) or rename one of the methods.
- Don't use folder names *private* or *protected* in your structure, except in folder based classes where all private methods are stored in this folder, e.g. don't put a class llb in a folder called *private*. Rename these if used in any other way than as a private folder in folder based class.
- For aggregated classes, you might sometimes end up with a broken wire using a method from a subclass, which is not present in the base class (where it would

	Document Name	Document Class	Date	Version
	G1G2 Converter	Instruction	June 20 2011	1.3
Author	Document Nr		Page	
Mattias Ericsson	4-100-12		11 (11)	

have been declared virtual). In GOOP2 all classes within a class hierarchy used the same object reference control (from the root base class), but for LabVIEW classes this type checking is more strict and in order to do this, you need to insert a *cast to more specific class* node as shown on the pictures below. You could also add a base class method declared dynamic dispatch (virtual), and then you don't need the *cast to more specific class* node.

