# Updating Legacy Experiments to work with SpinsolveExpert V1.4x

Contents

[Required changes due to new Experiment Macro version numbers 1](#_Toc52787985)

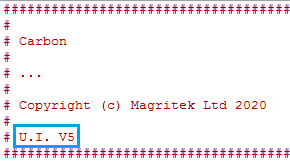
[Recommended changes 3](#_Toc52787986)

[Nice to haves 7](#_Toc52787987)

The SpinsolveExpert interface has gone through a number of changes since its inception. This has resulted in modifications to the format for the various macros used to describe an experiment. This document describes how to update these experiments to work with the new software. Some of these changes are required while others are optional.

## Required changes due to new Experiment Macro version numbers

The main change has been to the Experiment macro. This has a version number in the title



The version number can be:

**V3** These are interfaces designed before Expert was released. They include the user interface description in the file and are designed to work with the standard Prospa interface.

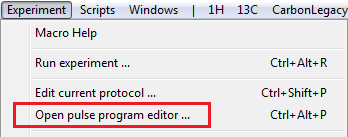
**V4** These use the interfaces designed for SpinsolveExert before the introduction of classes in V1.40. This still uses significant amounts of code from the old Prospa interface.

**V5** This is the latest version and is designed to work with the new class based macros introduced in V1.40.

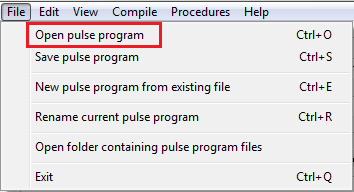
The aim of this progression is to simplify the exposed code and make it easier to extend in the future.

Converting a legacy macro from the old Prospa style interface (V3) or older Expert interface (V4) into V5 format for SpinsolveExpert/Tester is straight forward:

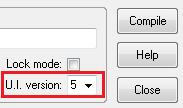
1. Select the Pulse program editor from the main Experiment menu



1. In the Pulse program editor and compiler window select the menu option File->Open pulse program. Note that you probably won’t be able to load it from the main Expert menu if it is already visible there. When selecting the legacy experiment select the folder not the macro.



1. Once the experiment has been successfully loaded select the UI version (5) Lock mode and press compile.



1. Additional changes required to V3 interfaces.
   1. All experiments require a ‘nucleus’ parameter. This should be the first parameter in the pulses sequence list. This allows the program to know what frequencies to expect.

...

* 1. The nucleus for the B1 frequency should also be explicitly defined in the variable name i.e. b1Freq1H or b1Freq13C *not* b1Freq. b1Freq should only be used if you are detecting on a nucleus other than proton. In this case it should be defined in the relationships list in the pulse program like this:

...

“b1Freq = b1Freq13C”,

...

if the detection is for carbon. This is the only place this variable should be used. If you don’t define this variable then the default transmit and receive frequency will be for proton, assuming the variable b1Freq1H is defined. Make sure b1Freq is not defined in the default parameter file as this will also set the default transmit and receive frequency.

* 1. In the experiment macro all file names which start with *kea* should be replaced with *ucs* (ultra compact spectrometer – an early name for the Spinsolve). eg. keaRun:getData becomes ucsRun:getData. This should only be necessary if updating a V3 interface.

Recompile after making these changes

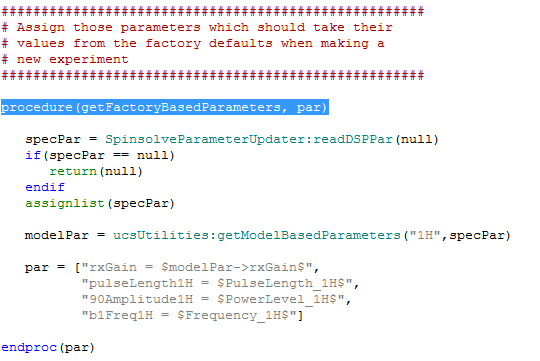
The experiment should now run as before.

## Recommended changes

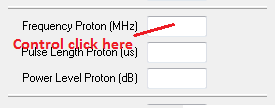
The following changes are not mandatory, but they will make the experiment easier to use

1. In the Experiment or pulse program macro add a procedure called getFactoryBasedParameters. This can be copied from a similar experiment. This procedure replaces certain variables with factory defaults when loading an experiment from the menu. In this way we can ensure that these parameters have sensible values.

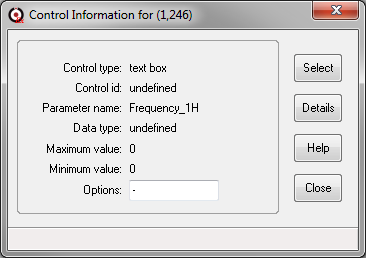
Typically this will replace the B1 frequency, the 90 pulse length and amplitude and the receiver gain.



The procedure getModelBasedParameters returns a structure called modelPar which contains the recommended receiver gain and dwelltime for this experiment. In this case we choose to ignore the dwelltime parameter. Note that all parameters come directly from the Spinsolve. You will need to know the names of these hardware variables. You can find these by opening the Spinsolve Parameter dialog from the main Files menu. Click on the parameter text box while holding down the control key to see the parameter name.



e.g. when the Frequency Proton textbox is clicked the following small dialog appears.



From this we see that the parameter name is Frequency\_1H. Note that this method works on all textboxes and other controls throughout Expert.

You can also type

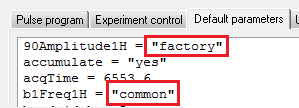
> pr gData->dspParameters

This will print a complete list of these parameters in the command line interface. (Assuming you are connected to a running Spinsolve).

1. Ensure that all default parameters are set and are relevant.

This one is quite important. By giving the default parameters sensible values you can ensure that the experiment will work first time. This is important when the experiment has a lot of carefully chosen delays or amplitudes and it is a while since you chose these. Also any parameter defined here which is not exposed in the user interface will also be available throughout the experiment so take care not to add variables which might cause problems (like b1Freq as mentioned above).

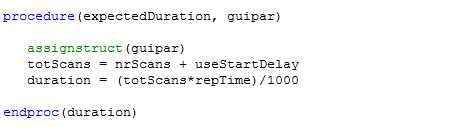
Default parameters which take factory defaults (see above) should have the value “factory” to make this clear while common parameters – i.e. those parameters which vary with calibration, such as frequencies, should be labelled as “common”



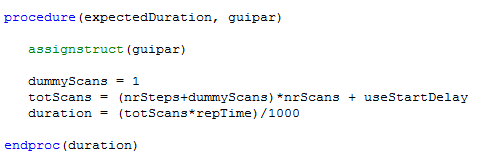
Again this is not essential, but it is less confusing that putting in some random and perhaps incorrect values.

1. Provide timing information.

Previous versions of Expert needed a potentially complex procedure called updateProgress in the main Experiment macro. This kept track of the experiment progress updating the progress bar after each scan. For 2D experiments the calculations here could become quite complex. This latest version of Expert replaces this with a simpler procedure called expectedDuration. This should return the expected duration of the experiment in seconds assuming that each scan takes 1 repetition time. An example from the Proton experiment is shown here:



And a 2D experiment like T2 requires only minor additions – in this case the nrSteps and dummyScans



The only non-user interface parameter here is useStartDelay. This is either 1 or 0. In history mode it is 0 and in batch mode 1. This means that in history mode the experiment will start immediately while in batch mode you need to wait one repetition time before the experiment starts.

If expectedDuration is present then the progress bar will update during the experiment automatically.

However some other changes are necessary to support these options. In the main experiment loop you should remove references to updateprogress and the organisation of commands should be like this:

Scan loop

Set phase

Write DSP parameters

Call checkTiming

Run the experiment using getData

Accumulate

Transform and plot

Check for finish

End of scan loop

It is not necessary to set any timers now and the timing check is done before the experiment starts, not afterwards. Notice the new timing command

check = ucsRun:checkTiming(guipar,scan,pcList)

rather than

check = ucsRun:checkTimeAndAbort(guipar,t1,scan,pcList,"ignoreLastScan")

Please refer to the Carbon experiment for an example of this. Proton uses an even simpler structure combining the first 4 commands in the scan loop into a single command called runSequence.

(data,pAcq,status) = ucsRun:runSequence(guipar,ppList,pcList,pcIndex,scan)

Both methods are equivalent.

## Nice to haves

The following changes will make the code less confusing to other users and easier to read, but are not essential. Please refer to examples of these changes in the Proton and Carbon experiment menus.

**The pulse program macro**

1. Add a sensible comment at the top of the file. Incorrect comments can be very confusing.
2. Remove the x, y coordinates of the controls and in the comment – this is no longer used in V4 and 5 of the interface
3. Add semicolons at the end of each line (except the last) in the interface description. This converts it to a 2D list and makes it easier to catch errors such as missing commas.
4. Line up columns in the interface and relationship tables. Again this makes it easier to spot errors.
5. Rename the ‘tabs’ variable ‘groups’ and also change this name in the returns variable list at the end of the pulse\_program procedure (tabs were used in the V3 interface)
6. Remove the dim variable definition (not used) and set dim=0 in the return variables.
7. Be careful not to have mismatched quotes anywhere in the file.
8. Recompile.

**The experiment control macro**

1. Use external procedures to simplify the code as much as possible. See the latest Proton and Carbon sequences for examples.

**The defaults parameters macro**

1. Remove the following default parameters which are no longer used in V4 and V5 of the interface: expNr, incExpNr, position, windowSize.
2. Be careful not to have mismatched quotes anywhere in the file.

**The user interface macro**

The processing\_controls procedure can now use a 2D structure instead of a 2D list. This results in a more readable syntax. Here is an example from the Proton experiment. Each button definition is defined by a series of comma delimited assignments and finishes with a semicolon.

layout = struct(buttonLabel = "SNR", plotName = "pt1", macroToRun = "snrSpectrum()";

buttonLabel = "FT", plotName = "pt1", macroToRun = "apodizeNTransform(\"pt1\",\"pt2\")";

buttonLabel = "Phase", plotName = "pt2", macroToRun = "manualPhase1DSpecial()";

buttonLabel = "Apodize", plotName = "pt2", macroToRun = "apodizeFreq()";

buttonLabel = "SNR", plotName = "pt2", macroToRun = "snrSpectrum()";

buttonLabel = "ppm/Hz", plotName = "pt2", macroToRun = "togglePPM\_Hz(1)";

buttonLabel = "Calib.", plotName = "pt2", macroToRun = "calibrateXAxis()";

buttonLabel = "MNova", plotName = "pt1", macroToRun = "exportMNova(\"pt1\")";

buttonLabel = "MNova", plotName = "pt2", macroToRun = "exportMNova(\"pt1\")";

buttonLabel = "Integ.", plotName = "pt2", macroToRun = "PeakIntegration(\"pt2\")")