Falcon Programmer

Manual

Simon O'Rorke

4th July 2024

Software Version 1.0

Contents

[Introduction 3](#_Toc171024785)

[Getting started 4](#_Toc171024786)

[More about running batch scripts 9](#_Toc171024787)

[Falcon Programmer is keyboard-friendly 10](#_Toc171024788)

[Script-based vs standard Info pages 11](#_Toc171024789)

[Why would you want to replace a script-based Info page with a standard Info page? 11](#_Toc171024790)

## Introduction

Falcon Programmer is an open source batch configuration application for the UVI Falcon software synthesizer. Multiple types of configuration change can be implemented in thousands of Falcon programs with a single batch run, taking seconds to minutes.

There is currently an installer only for Windows. However, the source code should run on macOS; and a macOS installer will be provided as soon as a collaborator can be found to create the installer and test the application on macOS.

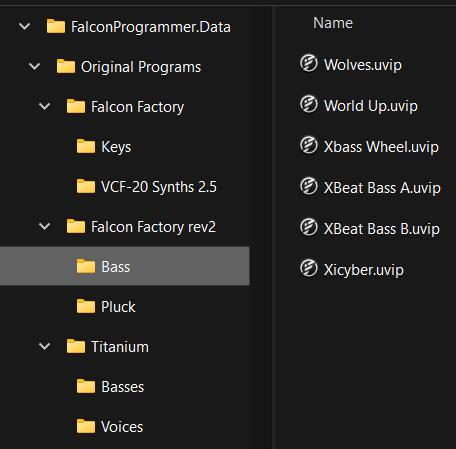
The following configuration tasks are available:

* Restore the program to an original version, ready for the configuration changes to be made.
* Initialise the program's Info page layout with many options, including converting a script-based layout to the standard layout.
* Assign MIDI CC numbers to macros and, provided the program uses the standard Info page layout, optionally append each macro's MIDI CC number to its display name.
* Bypass (disable) all known delay effects and then, provided the program uses the standard Info page layout, remove any macro that no longer modulates any enabled effects.
* If a Release macro is not part of a set of four ADSR macros and the macro is not modulated by the mod wheel, set its initial value to zero.
* Set the values of known reverb macros, with some exceptions, to zero.
* Move release and reverb macros that have zero values to the end of the standard Info page layout.
* Where feasible, replace all modulations by the modulation wheel with modulations by a new 'Wheel' macro on the standard Info page layout.
* If the modulation wheel's modulations have been reassigned to a Wheel macro (the previous task), reuse MIDI CC 1 (the mod wheel) for a subsequent macro, where feasible.
* Prepends a line indicating the program's path (sound bank\category\program name) to the program's description, which is viewable in Falcon when the Info page's ***i*** button is clicked.

Of these configuration tasks, assigning MIDI CC numbers to macros will be of use to many Falcon players. And restoring the program to an original version is just a safety feature to facilitate subsequent transformation. The remainder are merely what the developer has found useful as a Falcon player. Many more configuration tasks are surely possible. Users of the application are welcome to suggest some!

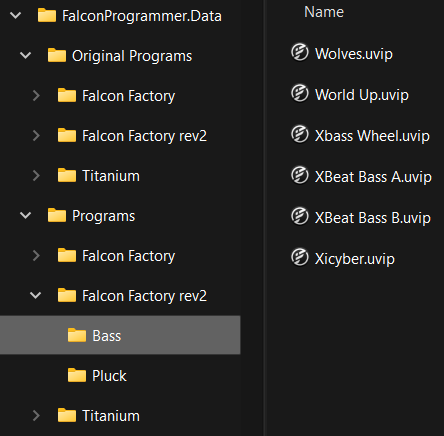
## Getting started

Save the Falcon programs you want to transform from Falcon sound banks into a folder hierarchy that will hold the original versions of the programs (perhaps with your own modifications). This tedious procedure is unfortunately necessary because Falcon Programmer cannot access the Falcon sound banks. The folder hierarchy must reflect the Falcon sound bank\category hierarchy, like this:



The names of the folders do not have to be exactly the same as the corresponding Falcon sound bank and category names. Falcon Programmer will get the original names from inside a program when applying sound bank-specific or category-specific rules. However, to avoid confusion, the folder names should at least be similar. For example, if you don't have the Falcon Factory (version 1) sound bank but do have the Falcon Factory rev2 sound bank, you could just call the latter's sound bank folder 'Factory'.

Next, copy the Original Programs folder to a Programs folder, like this:



Falcon Programmer's configuration tasks will update the copies of the Falcon program files in the Programs folder.

The last addition to the file system required before you run Falcon Programmer is a folder, initially empty, to contain Falcon Programmer's settings:

A screenshot of a computer screen

Description automatically generated

Falcon Programmer will save its settings to file Settings.xml, which it will create in the Settings folder.

Now load Falcon Programmer. As you have not yet told Falcon Programmer where your data is, you should see an error message like this:

Click the OK button. The Locations page will be shown:

Leave the Template Programs Folder field empty for now: as the tip in its text box says, you may not need it. Specify the other three folder locations:

**Note:** Falcon Programmer will save its settings whenever you go to a different tabbed page or close the application.

While we are on the Locations page, have a look at the page tabs:

To the right of the Locations page are several more pages of settings. Many configuration tasks require some of these additional settings to be specified.

But let's try running a batch script that requires no more settings. Go to the Batch page:

**Tip:** If you would prefer a different colour scheme, you can pick one from the menu at the top right of the main window.

Colour schemes adapt to the operating system's light/dark colour mode. This is the Light variant of the Forest colour scheme, whose Dark variant is shown above:

Select the scope for the batch run. For this tutorial, pick one program.

Now select configuration tasks RestoreOriginal and PrependPathLineToDescription, in that order.

**Tip:** The Task drop-down list shows the tasks in a logical order for running: a task is listed after any tasks that need to be run before it. *PrependPathLineToDescription always has to be run last*, due to a technical constraint on how the line breaks in the description are conserved.

If you have not run any batch scripts yet, RestoreOriginal will make no difference to the content of program file. However, *it is good practice to always start with RestoreOriginal*, and follow it with tasks in a logical order to transform the program to the state you want it in.

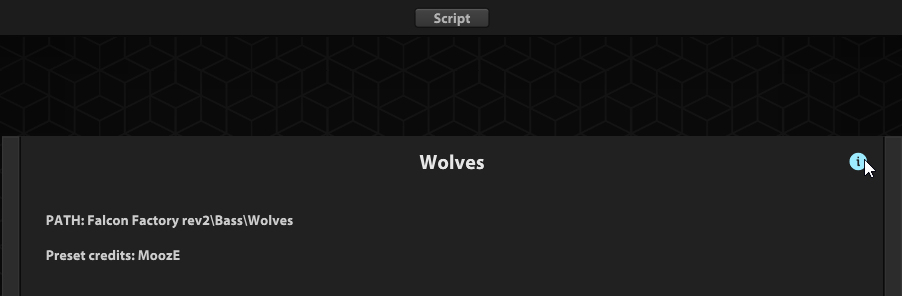
You have created a batch script, which consists of a scope and a list of configuration tasks. Click the Run Script button to run it. You should see something like this:

Have a look at the log. It shows the task names, the paths of the programs run and specific actions taken (sometimes more than one per task for some tasks). *The log can also show tasks that could not be run or actions that could not be taken for a program, with reasons why. So it is very useful for problem solving.*

Finally, have a look at the change you made. In their original forms, Falcon Factory rev2 programs all have script-based Info pages. So you first need to click the Script button to show the standard Info page.

And click the ***i*** button …

… to see the path line prepended to the (in this case very short) description.



## More about running batch scripts

The current batch script will be automatically be saved with the settings. So the next time you load Falcon Programmer, the last script you ran will still be on the Batch page.

At the top of the Batch page is a row of buttons whose use should be mostly self-explanatory.

You can **Save** the current script to file and **Load** a saved script to modify or run. Batch script files are plain text files in XML format. The **Edit Script File** button opens a batch script file to edit or view the text in whichever application is associated by the operating system with the .xml file extension. It is recommended to use an editor that shows the XML text colour-coded. Free editors that do this include Visual Studio Code (macOS and Windows), Notepad++ (Windows) and BBEdit (macOS). Here is a batch script opened in Notepad++:

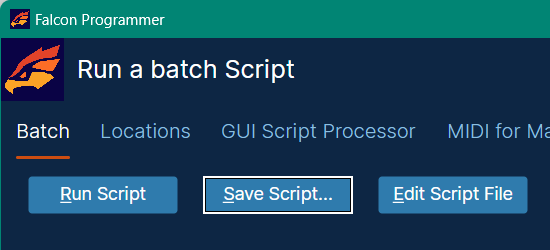
The Log display in Falcon Programmer is quite rudimentary: you can scroll it but not search or save it. To do more with the Log, click the **Copy Log to Clipboard** button and paste it into a text editor or word processor.

When a batch script is running, you can cancel the run with the **Cancel Run** button.

## Falcon Programmer is keyboard-friendly

Falcon Programmer's user interface is designed to facilitate keyboard use, minimising scenarios where recourse to a mouse is required. If you are used to keyboard shortcuts, most of this should be intuitive, so we will just cover a few points here.

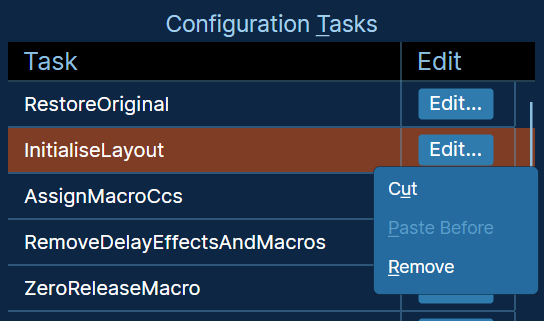
A focus rectangle surrounds the button, grid cell etc. that has the focus:



The underlined access key in a label or control caption, when pressed with the Alt key down, activates the required control without focusing it.

Grid keyboard shortcuts are similar to Excel. However, there is one that may not be obvious. Emulating a click on a button embedded in a grid cell is a two-step procedure. First, navigate to the button cell with tab or arrow keys:

Then press the F2 key, which is the standard key for putting a grid cell into edit mode. If it's a Browse button, the Open dialog will be shown. If it's an Edit button, the Edit context menu will be shown:



## Script-based vs standard Info pages

To make the best use of Falcon Programmer, a key Falcon concept of that needs to be understood is the choice between two ways of defining the Info page's GUI/layout. **A standard Info page** contains just two types of control, and they always look the same: continuous macros and toggle macros. Visual variety is provided only by a background image. Here's a standard Info page:

The appearance of **a script-based Info page** is defined in a special **GUI script processor**. Here's the script-based Info page of Falcon Factory rev2\Bass\Big Sleep:

Configuration task **InitialiseLayout** will remove the GUI script processor, if found, from any program whose sound bank or category is not listed on the **GUI Script Processor page**, so that the standard Info page will be shown.

### Why would you want to replace a script-based Info page with a standard Info page?

Here some possible reasons:

* You may find some script-based Info pages less easy to look at or less ergonomic to use than the equivalent standard info pages.
* Some script-based Info pages make the program slow to load. Modular Noise programs with script-based Info pages take around 10 seconds to load on a fast computer.
* Some Falcon Programmer configuration tasks can reposition macros on the standard Info page layout, provided any GUI script processor has been removed. See the documentation for specific configuration tasks later in this manual.

Other reasons relate specifically to configuration task **AssignMacroCcs**, which assigns MIDI CC numbers to macros:

* **AssignMacroCcs** can optionally append each macro's MIDI CC number to its display name, provided a program has a standard Info page.
* *For the Factory (version 1) sound bank*, **AssignMacroCcs** only supports GUI script processors for a few specific categories. See the **AssignMacroCcs** documentation below for details.
* Probably most importantly, *for the Factory rev2 sound bank and possibly other sound banks*, configuration task **AssignMacroCcs** cannot always correctly distinguish between continuous macros and toggle macros when allocating MIDI CC numbers to the macros.

This last point requires explanation. Hardware continuous controllers, such as knobs and expression pedals, send the whole range of MIDI values from 0 to 127. Hardware toggle controllers, such as buttons and footswitches, send only 0 or 127. They need to be assigned to modulate continuous and toggle macros respectively on the Falcon program's Info page. For this reason, Falcon Programmer's **MIDI for Macros page** allows for separate lists of MIDI CC numbers for **AssignMacroCcs** to assign to continuous and toggle macros.

For a standard Info page, Falcon Programmer can tell whether each macro is continuous or a toggle. Therefore a MIDI CC number from the corresponding list can be assigned.

For a script-based Info page, the process is less straightforward. Consider the information provided about the GUI script processor on Falcon's Events page:

This is essentially just the name of the top-level script, 'stub.lua' in this example, and a picture of the Info page. In sound banks licenced from UVI, the top-level script, whose contents are embedded into the program file, is just a stub, whether or not it's actually called 'stub.lua': it contains a reference to what we can call the main script, which does all the work. You cannot look at the main script. Read access is not provided to main scripts referenced by script processors in programs belonging to the Falcon factory sound banks or other sound banks licenced from UVI.

If Falcon Programmer were able to read a program's GUI script, it might be possible to definitively recognise which macros are continuous and which are toggles. As it is, Falcon Programmer has two types of workaround it can attempt.

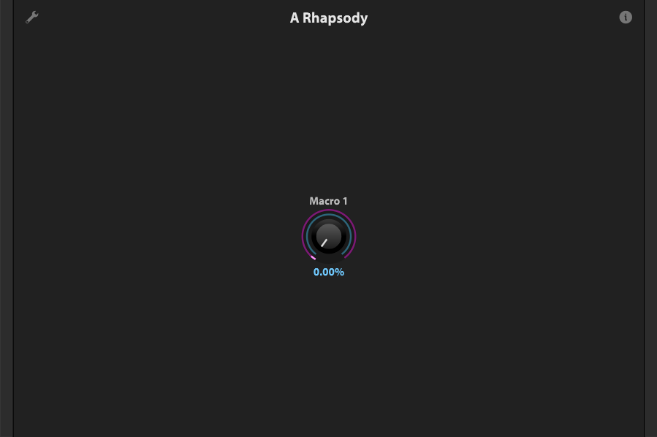
**Option 1:** Base the GUI script processor's MIDI CC assignments on **a built-in or user-defined template.** This is possible where all programs in a sound bank or category have consistent layouts.

**Option 2**: Base the GUI script processor's MIDI CC assignments **on the macros in the standard layout**. A simple example is the Titanium sound bank.

All Titanium programs have four macros, all continuous, on both the script-based GUI and the standard GUI. Notice that the order in which the macros are laid out on the standard GUI bears no relation to the order of the knobs on script-based GUI. That's fine, *as Falcon Programmer still uses a built-in script processor template for Titanium to determine the order in which MIDI CC numbers are to be assigned to the knobs on the script-based GUI, but not whether each is continuous or a toggle.* Because the standard GUI shows that *all* macros are continuous, their order does not matter.

Where continuous macros cannot be distinguished from toggle macros by examining the macros in the standard layout

However, it is only *necessary* where, in addition, continuous macros cannot be distinguished from toggle macros by examining the macros in the standard layout, which you can view by clicking the Script button near the top of the info page. An example is the Organic Keys sound bank:



As you can see, the standard GUI only contains one macro. It corresponds to the Expression knob on the script-based GUI.