Falcon Programmer

Manual

Simon O'Rorke
9th July 2024
Software Version 1.0

Contents

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.

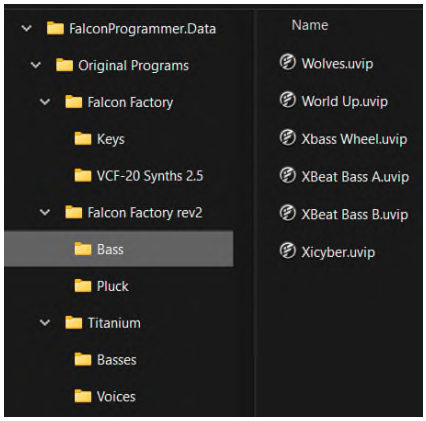
The following configuration tasks are available:

- 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!

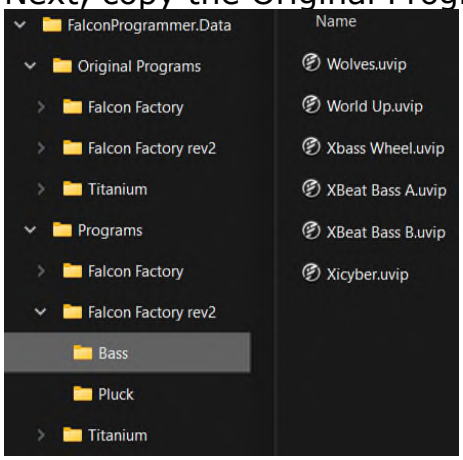
Getting started

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:



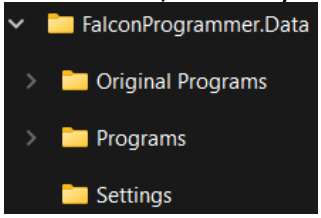
The names of the folders do not have to be exactly the same as the corresponding Falcon sound bank and category names. Falcon Programmer will get the original names from inside a program when applying sound bank-specific or category-specific rules. However, to avoid confusion, the folder names should 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:



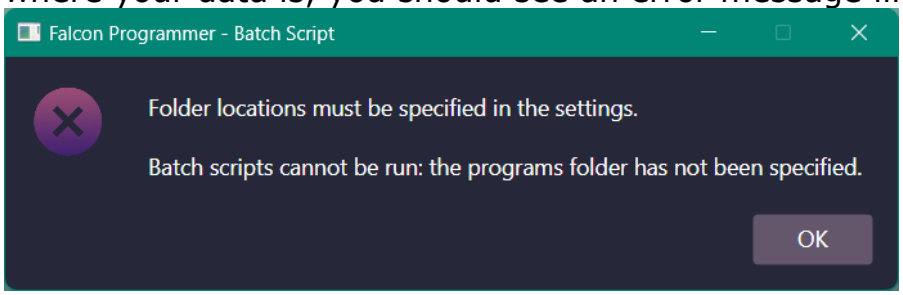
Falcon Programmer's configuration tasks will update the copies of the Falcon program files in the Programs folder.

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

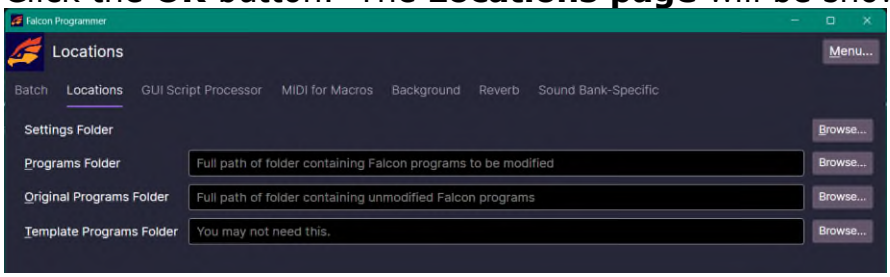


Falcon Programmer will save its settings to file Settings.xml, which it will create in the Settings folder.

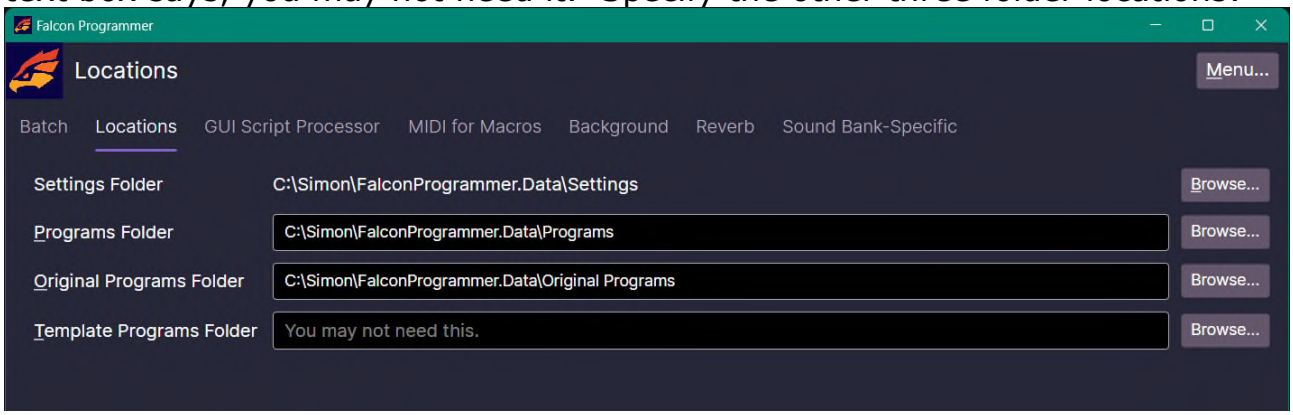
Now load Falcon Programmer. As you have not yet told Falcon Programmer where your data is, you should see an error message like this:



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

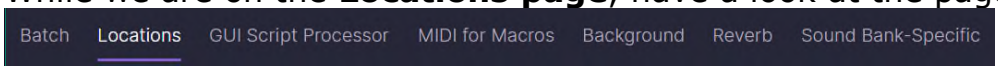


Leave the **Template Programs Folder** field empty for now: as the tip in its text box says, you may not need it. Specify the other three folder locations:

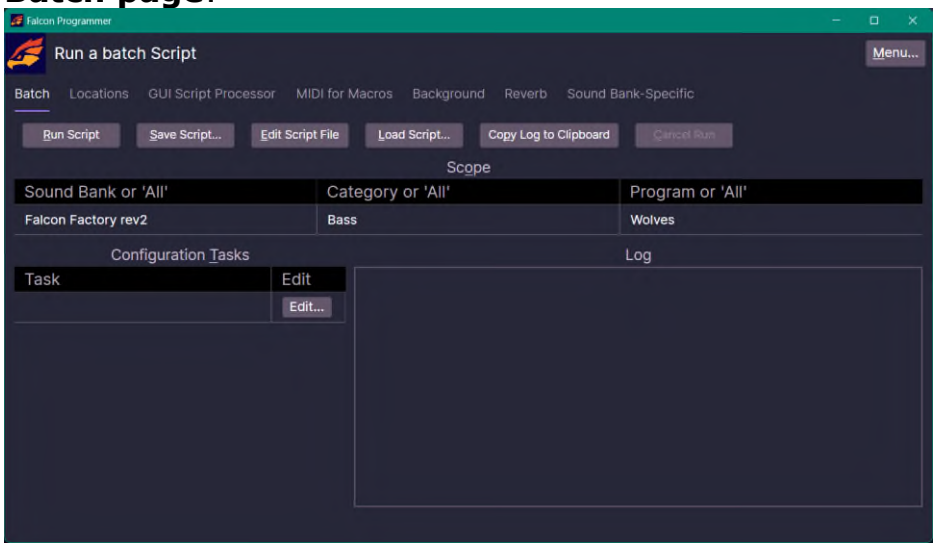


Note: Falcon Programmer will save its settings whenever you go to a different tabbed page or close the application.

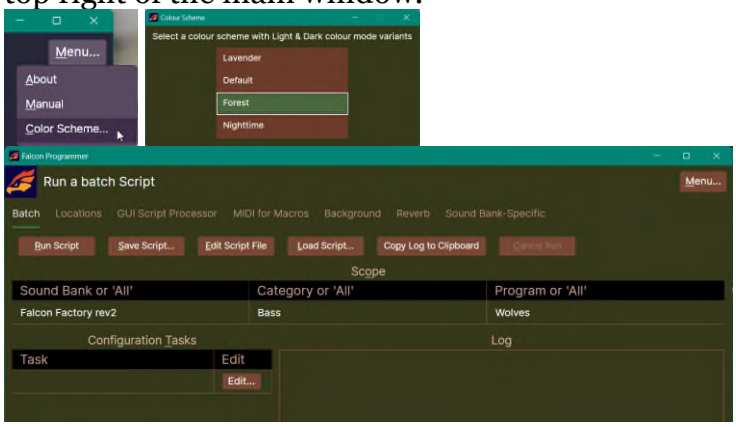
While we are on the **Locations page**, have a look at the page tabs:



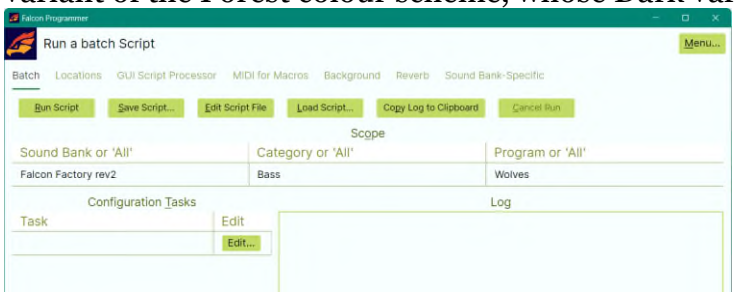
To the right of the **Locations page** are several more pages of settings. Many configuration tasks require some of these additional settings to be specified. But let's try running a batch script that requires no more settings. Go to the **Batch page**:



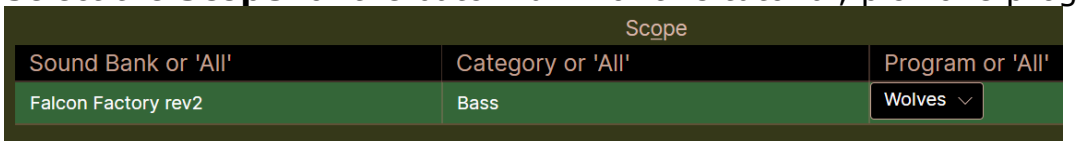
Tip: If you would prefer a different colour scheme, you can pick one from the menu at the top right of the main window.



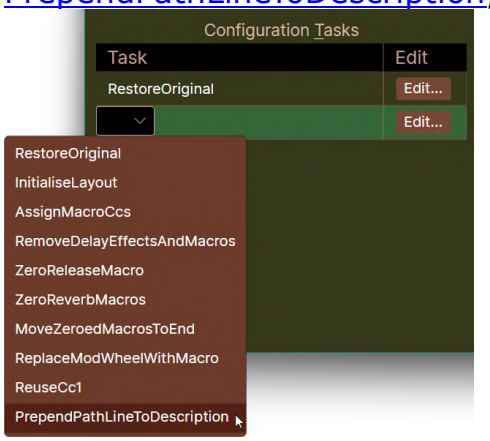
Colour schemes adapt to the operating system's light/dark colour mode. This is the Light variant of the Forest colour scheme, whose Dark variant is shown above:



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



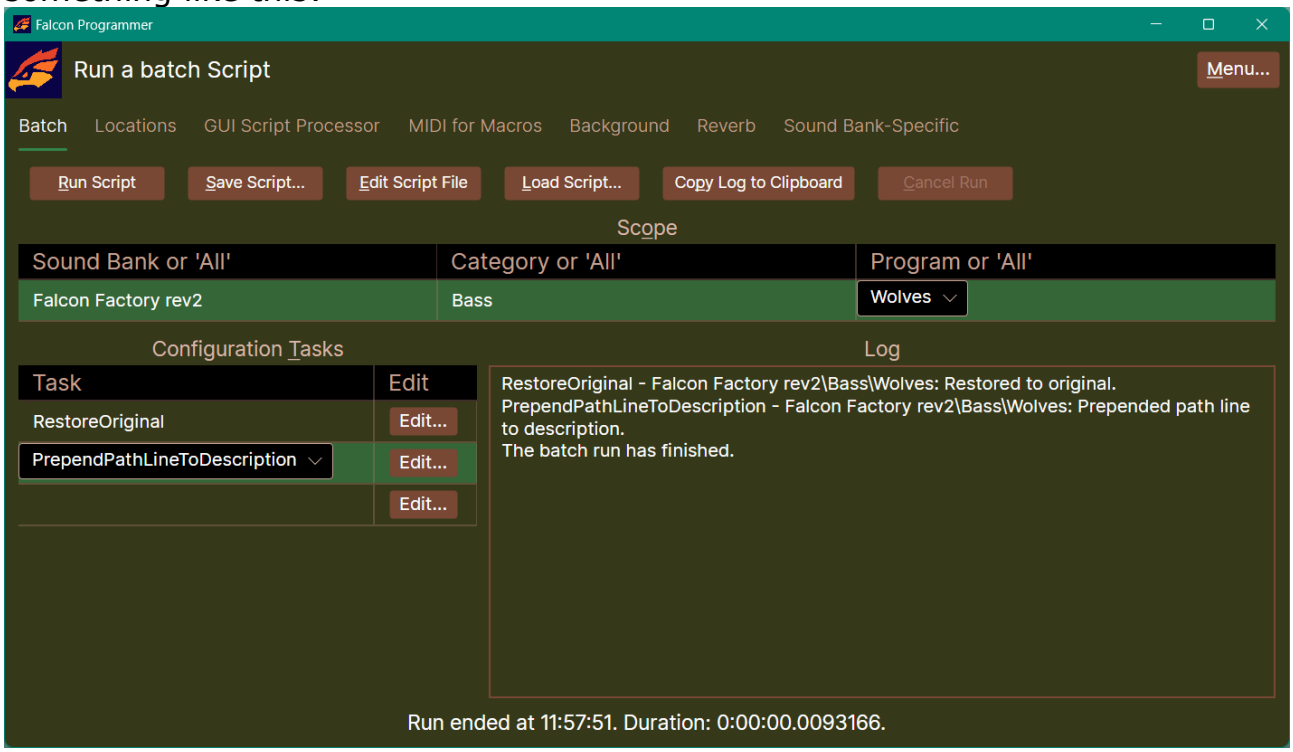
Now select **Configuration Tasks** RestoreOriginal and PrependPathLineToDescription, in that order.



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.

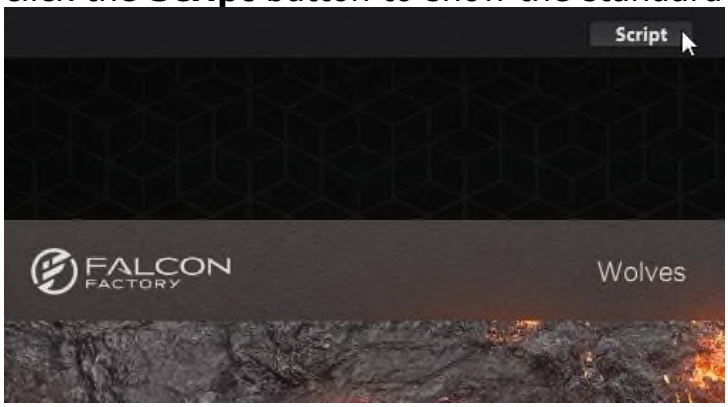
If you have not run any batch scripts yet, RestoreOriginal will make no difference to the content of program file. However, *it is good practice to always start with RestoreOriginal*, and follow it with tasks in a logical order to transform the program to the state you want it in.

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:



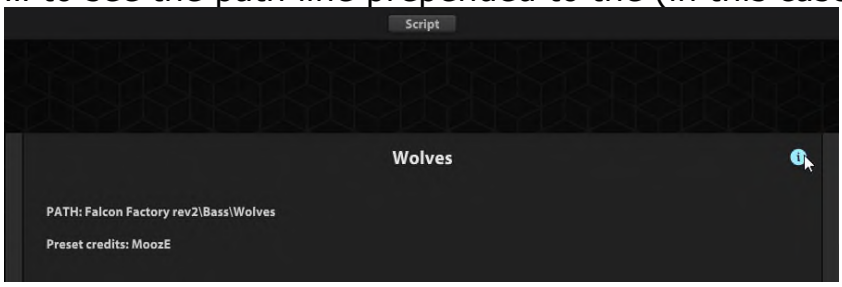
Have a look at the **Log**. It shows the task names, the paths of the programs run and specific actions taken (sometimes more than one per task for some tasks). *The log can also show tasks that could not be run or actions that could not be taken for a program, with reasons why. So it is very useful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

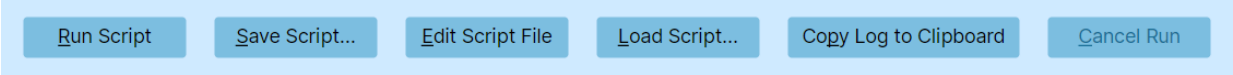
… to see the path line prepended to the (in this case very short) description.



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++:

```
<?xml version="1.0" encoding="utf-8"?>
<BatchScript xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema">
  <Scope>
    <SoundBank>All</SoundBank>
    <Category>All</Category>
    <Program>All</Program>
  </Scope>
  <Tasks>
    <Task>RestoreOriginal</Task>
    <Task>InitialiseLayout</Task>
    <Task>AssignMacroCcs</Task>
    <Task>RemoveDelayEffectsAndMacros</Task>
    <Task>ZeroReleaseMacro</Task>
    <Task>ZeroReverbMacros</Task>
    <Task>MoveZeroedMacrosToEnd</Task>
    <Task>ReplaceModWheelWithMacro</Task>
    <Task>ReuseCc1</Task>
    <Task>PrependPathLineToDescription</Task>
  </Tasks>
</BatchScript>
```

Tip: To conserve the scope already shown on the **Batch page** when loading a script, first edit the script file to omit the `Scope` element, like this:

```
<?xml version="1.0" encoding="utf-8"?>
<BatchScript xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema">
  <Tasks>
    <Task>RestoreOriginal</Task>
    <Task>InitialiseLayout</Task>
    <Task>AssignMacroCcs</Task>
    <Task>PrependPathLineToDescription</Task>
  </Tasks>
</BatchScript>
```

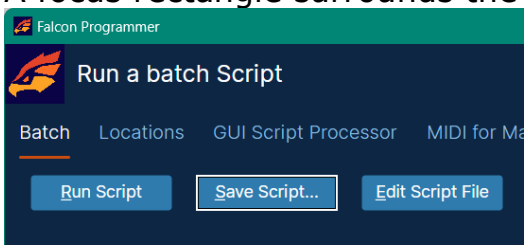
The **Log** display in Falcon Programmer is quite rudimentary: you can scroll it but not search or save it. To do more with the Log, click the **Copy Log to Clipboard** button and paste it into a text editor or word processor.

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

Falcon Programmer is keyboard-friendly

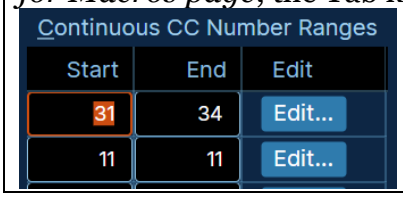
Falcon Programmer's user interface is designed to facilitate keyboard use, minimising scenarios where recourse to a mouse is required. If you are used to keyboard shortcuts, most of this should be intuitive, so we will just cover a few points here.

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

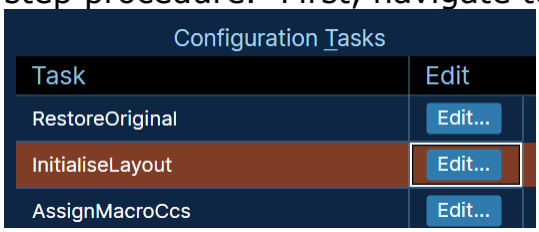


The underlined access key in a label or control caption, when pressed with the Alt key down, activates the required control without focusing it.

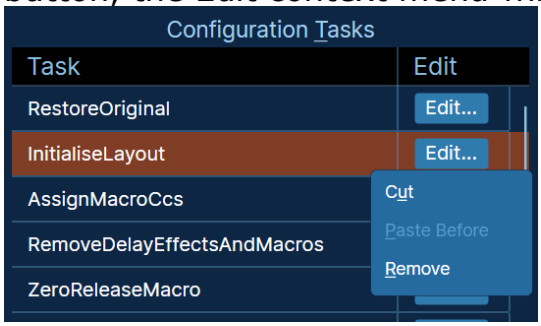
Tip: To focus the first cell of a grid after pressing Alt + the access key in the grid header label, press the Down arrow key or, *for either of the CC Number Ranges grids on the Midi for Macros page*, the Tab key.



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:



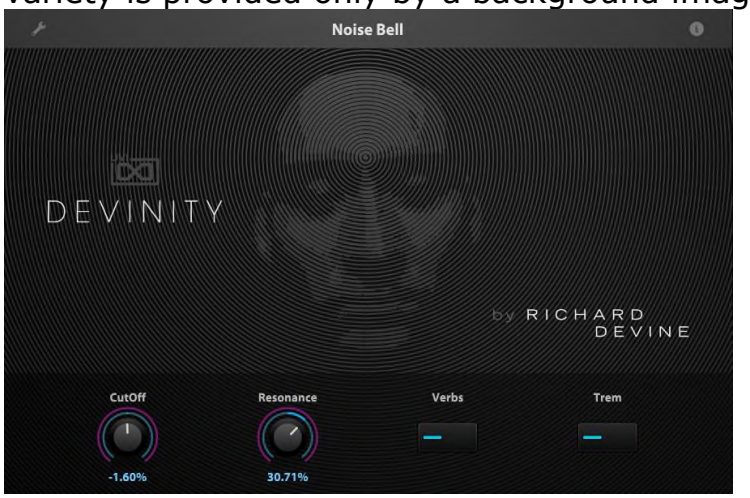
Then press the F2 key, which is the standard key for putting a grid cell into edit mode. If it's a **Browse** button, the Open dialog will be shown. If it's an **Edit** button, the Edit context menu will be shown:



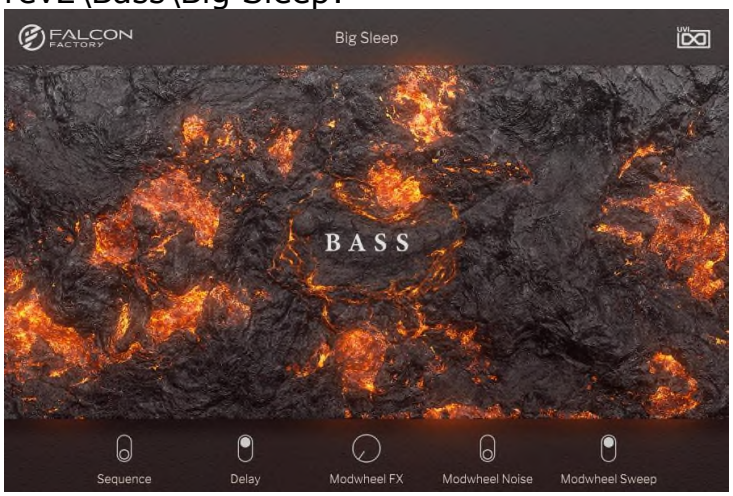
For either of the CC Number Ranges grids on the Midi for Macros page, this works differently. Just tab to an **Edit** button and then press the Enter key to show the Edit context menu.

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 might you want to replace a script-based Info page with a standard Info page?

Here some possible reasons:

- 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.

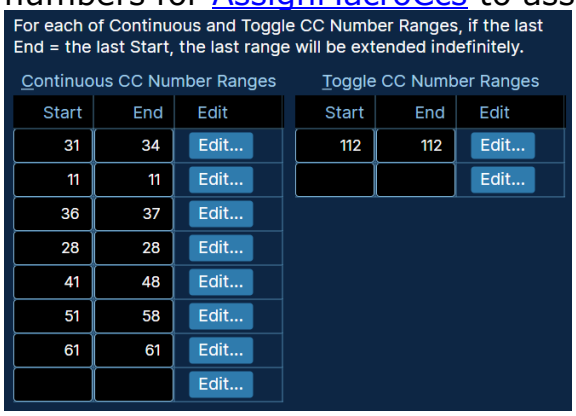
- You may find some script-based Info pages less easy to look at or less ergonomic to use than the equivalent standard info pages.
- Some script-based Info pages make the program slow to load. Modular Noise programs with script-based Info pages take around 10 seconds to load on a fast computer.

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.

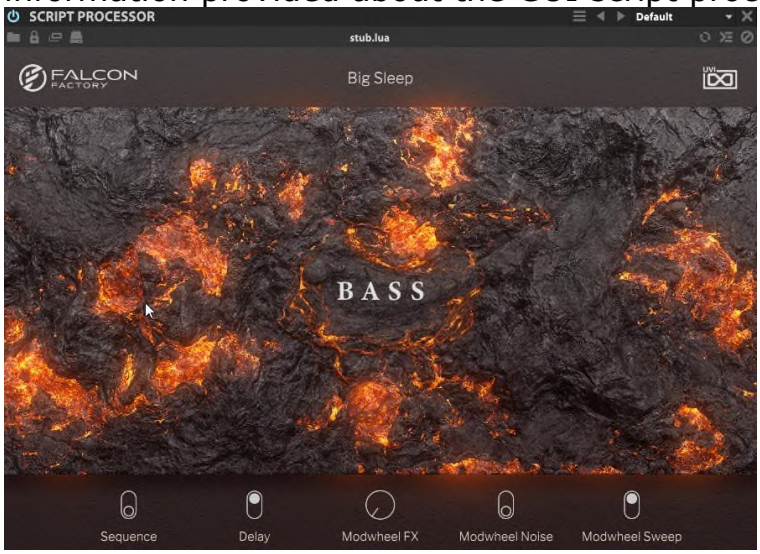
- AssignMacroCcs does not support GUI script processors for categories where some original programs have GUI script processors and others do not. Consequently, *for the Factory (version 1) sound bank*, AssignMacroCcs only supports GUI script processors for a few specific categories. See the Organic Pads sound bank documentation below for details.
- *For the Factory rev2 sound bank*, AssignMacroCcs does not support GUI script processors.

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:

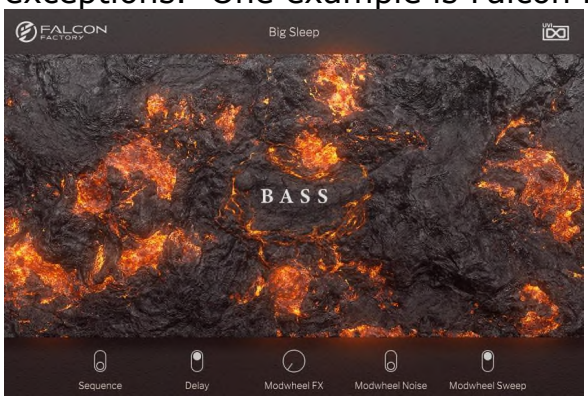


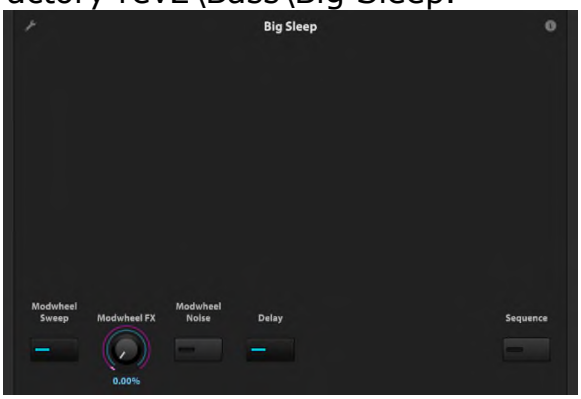
This is essentially just the name of the top-level script, 'stub.lua' in this example, and a picture of the Info page. In sound banks licenced from UVI, the top-level script, whose contents are embedded into the program file, is just a stub, whether or not it's actually called 'stub.lua': it contains a reference to what we can call the main script, which is embedded in the sound bank file and 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 controls (technically they are not macros) are continuous and which are toggles.

Instead, Falcon Programmer bases the GUI script processor's MIDI CC assignments on **a built-in or user-defined template.** However, this is only possible where all programs in a sound bank or category have consistent combinations of continuous and toggle macros in the same order on the Info page. For many if not all Falcon Factory rev2 categories, that is not the case.

An alternative that unfortunately proved not to be feasible would be to base the GUI script processor's MIDI CC assignments **on the macros in the standard GUI**, which you can view by clicking the Script button near the top of the info page. That would only work if all the controls in the script-based GUI corresponded to macros in the standard GUI that were in the same order. For most Falcon Factory rev2 programs, that is the case. But there are many exceptions. One example is Falcon Factory rev2\Bass\Big Sleep.

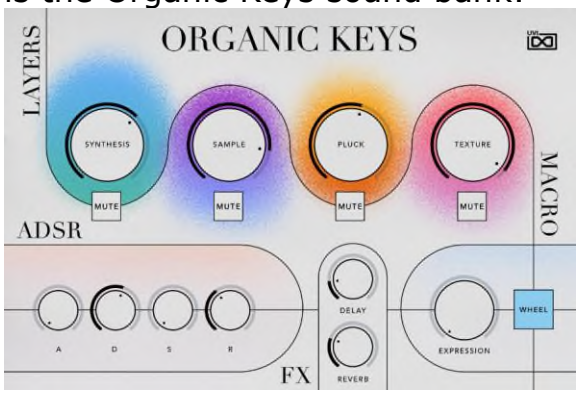


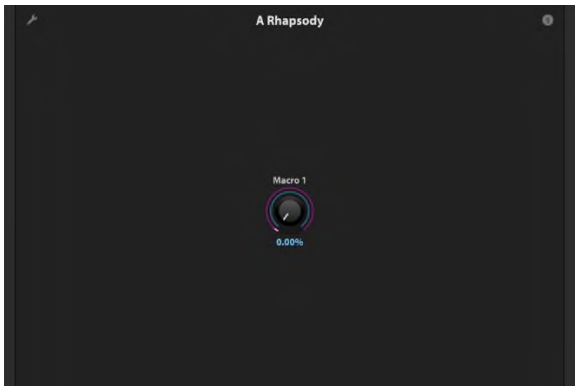


For category Falcon Factory\Brutal Bass 2.1, AssignMacroCcs does use a combination of those two methods to distinguish continuous controls from toggle controls on the script-based Info page. The last control can be either continuous or a toggle. Fortunately, the last macro on the standard Info page consistently indicates which it is.

Mandatory script-based Info pages

A few sound banks and categories do not have macros on the standard Info page GUI corresponding to every control on the script-based GUI. An example is the Organic Keys sound bank:



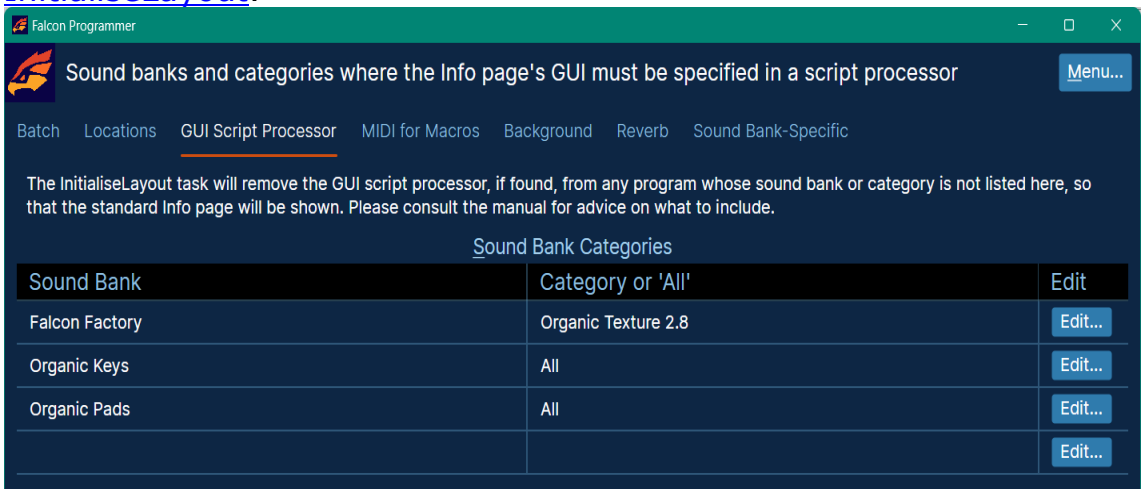


As you can see, the standard GUI only contains one macro. It corresponds to the Expression knob on the script-based GUI.

Clearly, simply removing the GUI script processor is not going to work for such sound banks and categories. So for two of the three known cases, the InitialiseLayout configuration task does not support GUI script processor removal. These are *the Falcon Factory (version 1)\Organic Texture 2.8 category and the Organic Keys sound bank*. For the *Organic Pads sound bank*, a new standard GUI is instead created: see the InitialiseLayout documentation for details.

GUI script processor page defaults

Falcon Programmer's **GUI Script Processor page** initially includes the sound bank and category for which GUI script processor removal is not supported, along with the Organic Pads sound bank, in the list of sound banks and categories for which the GUI script processor must not be removed by InitialiseLayout.



Organic Pads is included because the sounds of its programs will be modified in the standard GUI version.

GUI script processor templates

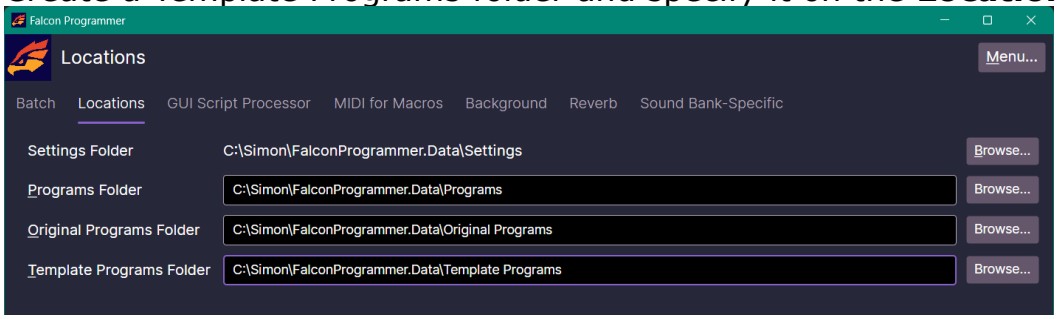
A sound bank-specific or category-specific GUI script processor template is required in order for the AssignMacroCcs configuration task to assign MIDI CC numbers to macros on a script-based Info page. Falcon Programmer has built-in templates for the following sound banks and categories:

- Falcon Factory (version 1) (category-specific)
  - Brutal Bass 2.1
  - Lo-Fi 2.5
  - Organic Texture 2.8
  - RetroWave 2.5
  - VCF-20 Synths 2.5
- Fluidity
- Hypnotic Drive
- Inner Dimensions
- Modular Noise
- Organic Keys
- Organic Pads
- Pulsar (category-specific)
  - Bass
  - Leads
  - Pads
  - Plucks
- Savage
- Titanium
- Voklm

User-defined templates

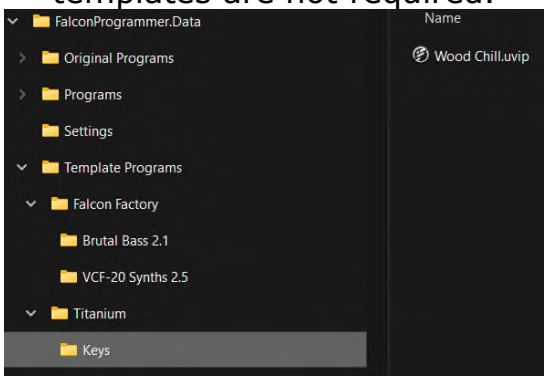
When Falcon Programmer does not provide a template or the provided template does not assign CC numbers to the macros in your preferred order, you can create your own template.

Create a Template Programs folder and specify it on the **Locations page**:

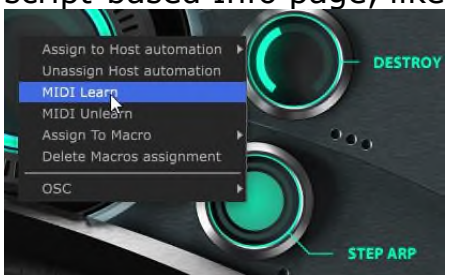


The folder hierarchy for Template Programs has the following differences from the folder hierarchies of Original Programs and Programs:

- Only one program file per category folder.
- Only one category folder in a sound bank folder where category-specific templates are not required.



In Falcon, assign MIDI CC numbers to the controls on the template program's script-based Info page, like this:



Then save the template program file and you are done. There is, however an optional additional step that you may find useful.

User-defined template creation: optional additional step

With the MIDI CC numbers saved to the template files, you would have to change them all if you had to play in a new hardware environment with different MIDI CC numbers. To avoid that, you can amend the template files so that AssignMacroCcs will assign MIDI CC numbers from the lists of continuous and toggle CC numbers specified on the **MIDI for Macros page** (which is what happens with the built-in templates).

To do so, you will have to manually edit each template program file.

Tip: Falcon program files, like Falcon Programmer batch script files, are text files in XML format. 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).

However, while a batch script file name has a .xml extension, a Falcon program file name has a .uvip extension. So your text editor may not initially recognise a Falcon program file as an XML file and in that case will not show the XML text colour-coded. To fix that, add an XML header line to the top of the file and then save the file. That does not stop Falcon from loading the program. Here is the required XML header line:

```
<?xml version="1.0" encoding="utf-8"?>
```

First, you have to find the GUI `ScriptProcessor` element. A program may have several `ScriptProcessor` elements but only one GUI `ScriptProcessor` element. The GUI `ScriptProcessor` element is the only one that contains modulations, which are `SignalConnection` elements in the XML. The modulations\`SignalConnection`s specify the MIDI CC numbers. So the GUI `ScriptProcessor` element will look something like this:

```
<ScriptProcessor Name="EventProcessor9" Bypass="0" API_version="17">
      <Connections>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC 31" Destination="Macro1" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC 32" Destination="Macro2" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC 33" Destination="Macro3" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC 112" Destination="Macro4" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
      </Connections>
      <Properties OriginalProgramPath="$Falcon Factory.ufs/Presets/Brutal
Bass 2.1/808 Line.uvip" ScriptPath="$Falcon
Factory.ufs/Scripts/Facotry2_1Stub.lua"/>
      <script><![CDATA[require("Factory2_1")]]></script>
      <ScriptData/>
</ScriptProcessor>
```

Replace the CC numbers, shown in red in the above example, with placeholders in format `Cn` and `Tn`. AssignMacroCcs will replace `C1` with the first continuous CC number on the **MIDI for Macros page**, `T1` with the first toggle CC number and so on. So you amended GUI `ScriptProcessor` element should look something like this:

```
<ScriptProcessor Name="EventProcessor9" Bypass="0" API_version="17">
      <Connections>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC C1" Destination="Macro1" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC C2" Destination="Macro2" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC C3" Destination="Macro3" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
            <SignalConnection Name="SignalConnection 0" Ratio="1"
Source="@MIDI CC T1" Destination="Macro4" Mapper="" ConnectionMode="1"
Bypass="0" Inverted="0" Offset="0" SignalConnectionVersion="1"/>
      </Connections>
      <Properties OriginalProgramPath="$Falcon Factory.ufs/Presets/Brutal
Bass 2.1/808 Line.uvip" ScriptPath="$Falcon
Factory.ufs/Scripts/Facotry2_1Stub.lua"/>
      <script><![CDATA[require("Factory2_1")]]></script>
      <ScriptData/>
</ScriptProcessor>
```

Configuration Tasks Reference

AssignMacroCcs

Summary: Assigns MIDI CC numbers to macros. The CC number can optionally be appended to each macro's display name, provided the program uses the standard Info page layout (the sound bank\category is not listed on the **GUI Script Processor page**, and the GUI script processor, if there was one, has been removed by InitialiseLayout). See Script-based vs standard Info pages.

The ranges of continuous and toggle CC numbers to be assigned must be specified on the **MIDI for Macros page**, which also includes the setting for whether the CC number is to be appended to each macro's display name.

If the sound bank\category is listed on the **GUI Script Processor page**, which indicates that the Info page layout must be defined in a script, the MIDI CC number ranges must be mapped to the GUI script processor's parameters in a built-in or user-defined template. See GUI script processor templates.

InitialiseLayout

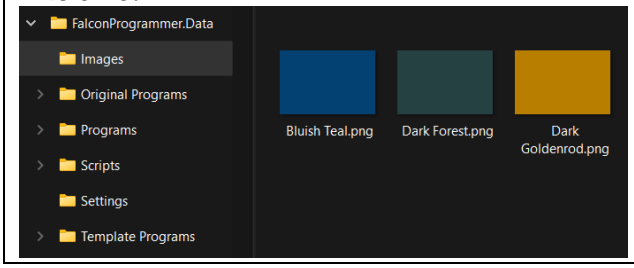
Summary: Initialises the program's Info page, with options specified on the **GUI Script Processor**, **Background** and **Sound Bank-Specific pages**.

- First, unless the sound bank or category is listed on the **GUI Script Processor page**, removes the GUI script processor, if there is one, so that a standard Info Page will be shown, with the macros arranged into a standard layout. See Script-based vs standard Info pages.

The remaining procedures are executed only if the standard Info page layout is shown, with the GUI script processor, if there was one, having just been removed.

- Sets the background image for the Info page, if one had been specified for the sound bank on the **Background page**. Background images should be PNG files with 720x480 resolution.

Tip: There are some background image files in the Background Images subfolder of the application folder. If you want to use them, it is recommended that you copy them to your own images folder, especially if you add new images of your own. Something like this:



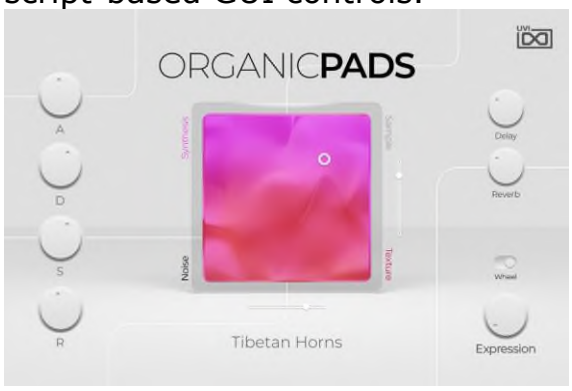
- If specified by options on the **Sound Bank-Specific page**, *for sound banks Ether Fields and Spectre*, rearranges the macros into a standard layout.

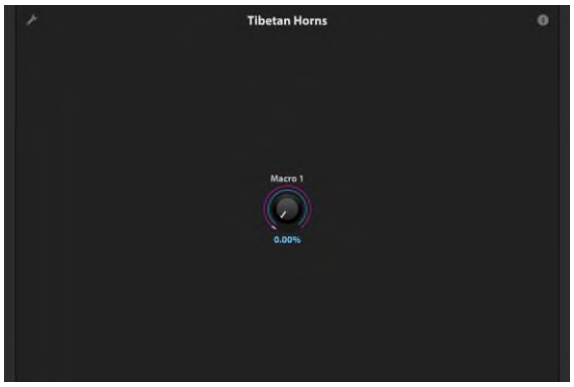
- If specified by an option on the **Sound Bank-Specific page**, *for sound bank Fluidity*, moves the Attack macro, if any, to be the last macro in the Info page layout.

- *For sound bank Organic Pads*, creates an entirely new standard Info page layout, as follows.

The Organic Pads sound bank

The Organic Pads sound bank lacks standard GUI macros corresponding to all script-based GUI controls.





As you can see, the standard GUI only contains one macro. It corresponds to the Expression knob on the script-based GUI. For the *Organic Keys* sound bank, which has the same problem, GUI script processor removal is consequently not supported: see Mandatory script-based Info pages.

However, InitialiseLayout *does support removal of the Organic Pads script processor*. To get round the problem with the original standard GUI, InitialiseLayout rebuilds the standard GUI to include macros corresponding to the script-based GUI controls, with some exceptions.

Instead of two macros to emulate the X-Y control on the script-based GUI, a macro is added for each of the four Layer gains. This makes variation those four parameters mutually independent. *However, you may consider this change to be a disadvantage, especially if you have a hardware X-Y controller.*

All known delay and reverb effects are bypassed. This is out of necessity. For Delay and Reverb, instead of defined modulations, the Organic Pads GUI script processor has delay and reverb parameters, controllable by the script. As we do not have read access to the script, there is no way to tell specifically what

the parameters do. Consequently it is impossible to replace the Delay and Reverb controls on the script-based Info page with macros on a standard Info page.

- **Known reverb effects** are DelayedReverb (PreDelay Verb), Diffusion, FilteredReverb, GateReverb, PlainReverb, SampledReverb (IReverb) SimpleReverb, SparkVerb, TapeEcho.
- For **known delay effects**, see RemoveDelayEffectsAndMacros.

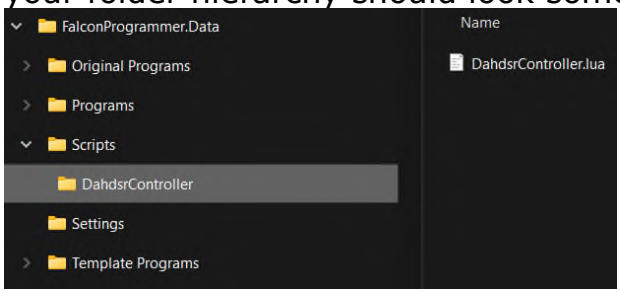
Maximum attack seconds, maximum delay seconds and maximum release seconds are initialised to values specified on the **Sound Bank-Specific page**.

Attack seconds and release seconds are *optionally* initialised to values specified on the **Sound Bank-Specific page**.

Script setup for Organic Pads

In order for the envelope macros Attack, Decay, Sustain and Release to control the DAHDSR modulator in the absence of the GUI script processor, a dedicated non-GUI script processor is added to each Organic Pads program. You need to make the new script processor's script accessible to it.

- Create a folder called 'Scripts' within the same parent folder as contains the Programs folder.
- From the Scripts subfolder of the application folder, copy file 'DAHDSR Controller.lua' and folder DahdsrController to the new Scripts folder. So your folder hierarchy should look something like this:



More about the DAHDSR Controller script

'DAHDSR Controller.lua' is the stub script embedded in the Falcon program file, while 'DahdsrController\DahdsrController.lua' is the main script that the stub script references.

You can use the DAHDSR Controller script to allow macros to control a DAHDSR in other Falcon programs than just those in the Organic Pads sound bank that are modified by InitialiseLayout. But in that case you need to follow the instructions at https://github.com/SimonORorke/UVIScript.

MoveZeroedMacrosToEnd

Summary: Moves release and reverb macros that have zero values to the end of the standard Info page layout. A non-zero macro will also be moved to the end if it is modulated by the wheel: in this case it is assumed that the player will use the wheel, and therefore does not require easy access to the macro on the screen.

Requirements:

- The program uses the standard Info page layout (so the sound bank\category is not listed on the **GUI Script Processor page**, and the GUI script processor, if there was one, has been removed by InitialiseLayout).

- **ZeroReleaseMacro** and/or **ZeroReverbMacros** have already been run, if required.

PrependPathLineToDescription

Summary: Prepends a line indicating the program's short path to the program's description, which is viewable in Falcon when the Info page's ***i*** button is clicked. For screenshots of the path line in the description, see Getting Started.

Requirements:

This task must be done last! Due to a fix required to conserve paragraph breaks in the description, updates by subsequent tasks can corrupt the program file.

RemoveDelayEffectsAndMacros

Summary: Bypasses (disables) all known delay effects. Then removes any macro that no longer modulates any enabled effects, provided the program uses a standard Info page.

Known delay effects are Buzz (Analog Tape Delay!) DiffuseDelay, DualDelay, DualDelayX, FatDelay, FxDelay, PingPongDelay, SimpleDelay, StereoDelay, TapeEcho, TrackDelay, VelvetDelay.

ReplaceModWheelWithMacro

Summary: If feasible, replaces all modulations by the modulation wheel with modulations by a new 'Wheel' macro.

Requirements:

- The program uses the standard Info page layout (so the sound bank\category is not listed on the **GUI Script Processor page**, and the GUI script processor, if there was one, has been removed by InitialiseLayout).

Exclusions:

- Sound banks on the "Do not run ReplaceModWheelWithMacro or ReuseCc1" list on the **MIDI for Macros page**.
- Programs that already have a macro with display name 'Wheel'.
- The mod wheel only 100% modulates a single macro. In this case, there's no point replacing the mod wheel modulations with a Wheel macro.

RestoreOriginal

Summary: Restores the program file in a sound bank\category subfolder of the Program Files folder from the corresponding file in the Original Program Files folder.

ReuseCc1

Summary: If the modulation wheel's modulations have been reassigned to a Wheel macro by ReplaceModWheelWithMacro, reuses MIDI CC 1 (the mod wheel) for a subsequent macro, where feasible.

MIDI CC 1 is assigned to the macro after the macro whose MIDI CC number is as specified by the Modulation Wheel Replacement CC No setting on the **MIDI for Macros page**. The MIDI CCs of any subsequent macros are incremented accordingly.

Requirements:

- The program uses the standard Info page layout (so the sound bank\category is not listed on the **GUI Script Processor page**, and the GUI script processor, if there was one, has been removed by InitialiseLayout).

- Any modulations by the wheel macro have already been reassigned to a wheel macro by ReplaceModWheelWithMacro.

- Modulation Wheel Replacement CC No > 1 has been specified on the **MIDI for Macros page**.

- There is a macro whose MIDI CC number is as specified by the Modulation Wheel Replacement CC No setting and at least one more macro after it.

Exclusions:

- Sound banks on the "Do not run ReplaceModWheelWithMacro or ReuseCc1" list on the **MIDI for Macros page**.

- Programs with macro modulations whose MIDI CC number is 1 but that have not been assigned to a wheel macro by ReplaceModWheelWithMacro. If anything, that should be done instead of assigning MIDI CC 1 to a different macro.

It would be possible but probably not desirable for ReuseCc1 to support programs with GUI script processors. For example, some Pulsar programs don't use the mod wheel and so could reuse CC 1. Pulsar programs have 30 macros, all continuous. Reusing CC 1 and incrementing the CC numbers of the following numbers for some Pulsar programs but not others would lead to two inconsistent CC numbering schemas. And it would not be easy to tell them apart, as it is not possible to append the CC number to the displayed name of a macro when the Info page's GUI is provided by a script.

ZeroReleaseMacro

Summary: If a Release macro is not part of a set of four ADSR macros and the macro is not modulated by the mod wheel, sets its initial value to zero.

ZeroReverbMacros

Summary: Sets the values of known reverb macros, with some exclusions, to zero.

Known reverb macros are those whose display names contain 'Reverb', 'Room', 'Sparkverb' or 'Verb' (not necessarily as whole words).

Exclusions:

- Programs listed on the **Reverb page**.
- Macros modulated by the mod wheel.