

User's Guide

For the

Solo Instrument Performance Suite

V2

R. D. Villwock

7-15-08

Table Of Contents

i	Please Read This	4
ii	What's New In Version 2.0	5
iii	Acknowledgements	6
1.0	An Introduction to SIPS	7
1.1	About the Starter Script	7
1.2	The SIPS Articulation Script	8
1.3	The SIPS Legato Script	8
1.4	The SIPS Vibrato Script	8
2.0	SIPS Common Features	9
2.1	Setting Instrument Range.....	9
2.2	Assigning MIDI Controllers.....	9
2.3	SIPS Assignable CCs.....	10
2.4	Setting a MIDI Controller's Range	10
2.5	Setting User Preferences	11
2.6	SIPS Preset System.....	12
2.7	User Presets.....	12
2.8	Naming User Presets	13
2.9	Updating Your Custom Presets for SIPS 2	14
2.10	Auto Import Problems	15
2.11	Extending V105 User Presets.....	16
2.12	Instrument Range Not Included in Presets	16
2.13	Import/Export of User Presets	16
2.14	About the Built-In Presets.....	17
3.0	SIPS Articulation Script	19
3.1	Introduction.....	19
3.2	SIPS Articulations.....	19
3.3	Fixed Articulations	21
3.4	Chained Articulations	23
3.5	Layered Chains.....	24
3.6	Building Articulations	25
3.7	Beware of K2/K3 Quirks.....	25
3.8	The SAS Control Panels.....	26
3.9	Formatting a New Instrument.....	29
3.10	Kontakt's Group Editor	29
3.11	Using the Setup/Audition Panel	30
3.12	Auditioning Groups	30
3.13	Auditioning Articulations	30
3.14	The Play Monitor/Info Window.....	30
3.15	About Release Groups	31
3.16	About 'Inside' Groups.....	31
3.17	Assigning Groups	31
3.18	Making Fixed Articulations	32
3.19	Making Chained Articulations.....	32
3.20	Using the Multi Art Commands	33
3.21	Renaming Groups	34
3.22	Check All Groups and Assignment Errors.....	35
3.23	Keyswitch Setup.....	35
3.24	SIPS Variations.....	36
3.25	The TKT Variation Index.....	36

Table Of Contents

(Continued)

3.26	Using the Play Mode Panel	38
3.27	Articulation Control Choices	38
3.28	Selecting Articulations with MIDI Program Change Messages	38
3.29	Selecting Articulations with Key Switches	38
3.30	Triggering Release Groups	39
3.31	Using Silent (Ghost) Notes	39
3.32	Variation Selection Modes	39
3.33	Key-Switch Selection of Variations	40
3.34	Using the Automatic Variation Modes	40
3.35	Auto-Triggering Options	40
3.36	Auto-Variation Sequencing.....	41
3.37	Full-Cycle Random Mode	42
3.38	Random Repeatability	42
3.39	The Keep Settings Menu.....	43
3.40	MIDI Menu Control.....	43
3.41	Group-Start Programming.....	43
4.0	SIPS Legato Script	45
4.1	Introduction.....	45
4.2	Playing Legato.....	45
4.3	Use of the Sustain Pedal.....	45
4.4	Mode Carryover and Playing Chords	46
4.5	Control Panel Layout.....	46
4.6	Understanding the Crossfade Contouring Controls.....	49
4.7	MIDI Control of the Crossfade Function.....	53
4.8	Understanding the Bend Contouring Controls.....	53
4.9	MIDI Control of the Bend Function.....	54
4.10	Using DFD Mode.....	55
4.11	Using Portamento Mode	56
4.12	Release Sample Triggering	59
4.13	Guidelines for Making Legato Settings	60
4.14	The Super Bender Script.....	62
4.15	Configuring the SLS for use with the Super Bender.....	64
5.0	SIPS Vibrato Script	65
5.1	Introduction.....	65
5.2	Using the Vibrato Envelope.....	65
5.3	Using Both the Efx Amt Knob and the Envelope	66
5.4	Humanizing the Vibrato Effect.....	66
5.5	Setting Random Drift	66
5.6	Control Panel Layout.....	66
5.7	Guidelines For Making Vibrato Settings.....	69
6.0	SIPS Tips, Techniques, and Musings.....	70
6.1	Theo Krueger.....	70
6.2	Andrew Keresztes	71
7.0	File Types and Installing Scripts	72
7.1	Fresh Install vs Update	73
7.2	Installing and Removing KSP+.....	73
7.3	Get the KScript Editor	73
A1	About Big Bob.....	74
A2	The Best Things In Life Are Free.....	75

Please Read This

V2 of SIPS has some exciting new features but it also breaks some new ground in trying to overcome certain K2/KSP deficiencies. As a result there are some unusual things you need to know about **SIPS 2** before installing it. **Therefore please read the following carefully before trying to install and use SIPS 2.**

In order to provide **Aftertouch** control of various **SIPS** parameters and to enable the new **Articulation Script** to respond to **MIDI Program Change** messages, **SIPS 2** requires the services of a properly-configured **KSP+ Multiscript**. Therefore, any multi, that will contain instruments using **SIPS 2**, requires that **KSP+** be installed in that multi. You'll find a custom-configured version (see footnote ***) of **KSP+** in the **Precompiled** folder with the file name **SIPS_KSP+.nkm** and you'll find instructions for installing this multiscript in **Section 7.2**.

K2 does not handle big scripts very well and the **SIPS** scripts are starting to get rather large. Therefore some techniques have been employed with **SIPS 2** to reduce code size and, these techniques result in a few minor inconveniences that you need to be aware of. When you first load **SIPS 2**, the control panels cannot be displayed until the first **MIDI CC** event is received. Therefore, when **SIPS** is first loaded (or you reload or hit Apply), the panels display a 'prompting message' telling you to move a **CC**. As soon as you wiggle any **CC**, the full set of control panels will be displayed. It's a minor inconvenience, but it saves a lot of code. Note however, that most of this inconvenience can be circumvented if you install the special **SIPS Starter Script** in slot 1 (see **Section 1.1**).

The members of a suite of scripts like **SIPS** need to communicate with each other, but, such Interscript Communication is not generally supported by K2. To work around this limitation, **SIPS** has to exploit certain 'accidental features' of the KSP which provides less than perfect synchronization of the scripts under certain conditions. Therefore, it is important that you always 'play by the rules'. For example, don't change any suite-wide parameters such as the **Instrument Range** or **User Preferences** while any of the scripts have their **KSP Bypass** button depressed. If you want to disable the function of one or more of the scripts, set the script's mode menu to **Off** instead of using the **KSP Bypass** button. When a script's mode is set to **Off**, the effect is disabled but interscript communication still continues. However, when the **KSP Bypass** button is used, communication between the scripts is broken and things can get 'out of step' (and automatic correction of such things in K2 is difficult).

SIPS 2 provides some very powerful **Articulation** and **Variation** control features but in order to do this, **SIPS must have exclusive use of the group-start parameters**. So you must not alter or add to any of these settings. Moreover, before you can use **SIPS 2** with an instrument, the instrument must be properly '**Formatted**' for use with **SIPS**. Therefore, when you first add the **SIPS** scripts to one of your instruments, **you must Format the instrument** before you can begin to explore the operation of **SIPS**. Formatting is a simple, one-click operation that is performed by the **Articulation Script** via its **Setup/Audition** panel. You'll find the easy instructions for how to format a new instrument in **section 3.9**.

For certain operations (such as formatting), **SIPS 2** needs to alter the group-start engine parameters and Kontakt exhibits some very quirky behaviour if other things are going on while these parameters are being changed. Therefore, it is important that you avoid doing certain things while utilizing the **Group Assignment** functions. So, please be sure to read **section 3.7** and follow the recommendations given there to avoid unnecessary problems (at least until NI does something about this newly-discovered set of problems).

*** **SIPS_KSP+** is *functionally* identical with **KSP+CC47_V110** (one of the precompiled multiscripts supplied with the recently released **KSP+ V110** package). However, **SIPS_KSP+** has been simplified to remove all unnecessary code for this particular configuration. So, while you can use the **KSP+CC47_V110** multiscript, the **SIPS_KSP+** multiscript will run more efficiently and is therefore recommended.

What's New In Version 2

First off, you'll be glad to know that nothing has been done to *damage* the sound or musicality of either the **Legato** or **Vibrato** scripts. Moreover, your investment of time in crafting custom control panels and user presets will not be lost. You will be able to transfer all your instrument control panels and/or user presets from prior versions without the need to manually re-enter all of the parameters.

SIPS 2 now has an **Articulation Script**, the **SAS**. This new family member allows you to define and control up to 64 different articulations per instrument. Articulations can be selected in real time via a key-switch pair and/or with MIDI program change commands. Each articulation can contain multiple Groups and group types (such as Release-Groups). In addition, each articulation can have up to 8 alternate TKT variations as well as up to 16 sampled variations via group chaining; all controllable in a variety of flexible ways. The **SAS** works seamlessly with the **Legato Script** so that Articulations and Variations can be changed at any time, even within a legato phrase. And, working in conjunction with the **SAS**, the **Legato Script** now provides support for triggering special sample-start offset groups for the 'inside notes' of legato phrases with a new **DFD Offset Mode**. Also in conjunction with the **SAS**, the **SLS** can optionally trigger release groups at the end of a legato or portamento phrase.

In addition, **SIPS 2** adds a new **Portamento Mode** to the **Legato Script**. With this mode, you can glide or bend between any pair of notes (using any desired-pitch-versus time contour) with a very realistic, **formant-corrected** timbre. All the important parameters of this mode are MIDI controllable and will enable you to achieve just about any kind of glissando effect you might want. The **Vibrato Script** has also been re-designed to make it more flexible and easier to use.

MIDI-Controllable Knobs are now easier to set and their range can cover any desired portion of the knob's full range (including the zero position if desired). By reducing the total knob-range covered by the **MC**, you can have more resolution where you need it most. With **SIPS 2**, setting up such subranges is very easy and requires no additional controls other than the assigned **MC** and the **Knob** itself. The min and max settings can also be interchanged, allowing the **MC** to control the knob **inversely** where that might be appropriate.

MIDI Controller assignment retains the new scheme adopted with **V1.5** of **SIPS**. You simply double-click the **MC** assignment button to reveal a drop-down menu of choices. As soon as you click the desired **MC** in the menu, the assignment is made. To indicate this, the assignment button is illuminated and displays the **MC** that you assigned. For **SIPS 2**, controller choices now include the **CCs** from **0** to **119** plus the **Pitch Wheel**, **Aftertouch**, and **Velocity**. The ability to assign **Aftertouch** control as well as the ability of the **SAS** to respond to **Program Change** commands is provided by the **KSP+ Multiscript** included with the **SIPS** package.

V2 of **SIPS** includes a new **User Preferences Dialog** that allows you to customize your version of **SIPS** in a number of useful ways to suit your equipment setup and style of working. For example, starting with **V1.5**, **SIPS** has taken over handling of the sustain pedal function of **CC64**. However some users do not actually utilize the sustain function of this controller and would prefer to use **CC64** as just another assignable **MIDI CC**. With **SIPS 2**, there is a preference option switch, that allows you to de-assign **CC64** from its sustain function if desired. See **Section 2.5** for a current list of all the preference options that are available.

There have also been many small, and sometimes subtle improvements made to **SIPS 2** and also to this manual. Therefore, I encourage you to re-read and study this entire **User's Guide** in order to get the most benefit from **SIPS 2**. As a **minimum**, you should carefully read the sections titled '**Please Read This**' on **page 4** and '**An Introduction To SIPS**' in **sections 1.0 to 1.4**. However, if you want to get the **most musical** results from **SIPS**, reading and understanding **all** the material presented in this **User's Guide** may prove to be most helpful.

Acknowledgements

I want to take this opportunity to thank everyone in the K2 community who played some part in the development of these scripts, both the original release and various updates. Many of you offered useful ideas and suggestions and many others offered encouragement. Since I'm not involved in orchestral work, I've had to depend upon others to try the scripts with their libraries and provide feedback as to the musicality and usefulness of the scripts in an orchestral context.

My thanks to Gary Lionelli who provided much useful feedback about the script's musicality and participated in the early development of the human interface for the V105 release. I especially want to thank Theo Krueger and Andrew Keresztes for 'jumping in with both feet' in the eleventh hour when Gary's workload made it impossible for him to continue. Theo developed most of the Legato Script presets and Andrew developed the rest. The Vibrato Script presets were developed by Theo Krueger, Andrew Keresztes, and Martin Nadeau. I also want to thank Martin for doing a very meticulous job of proof-reading the original User's Guide and finding most of my mistakes (you'll have to find the rest, especially in the revised manuals). Theo, Andrew, and Martin also created some very nice mp3 demos for each preset that they developed. For **SIPS 2** beta testing, I again want to thank Andrew and Martin, this time along with Tod Stillman, Jörg Pfeil, and Adriano Cabreira. These guys deserve a real vote of thanks for their enthusiastic dedication to this project. They even had to re-read the whole manual several times during the course of beta testing (because I kept making major changes to it, with the dubious excuse of wanting to improve the scripts) ;-). And, if you find some good uses for the new Velocity control option (now included in the MC Assignment menus), you can thank Jörg for suggesting that I include that option.

Many thanks to Nils Liberg for developing and sharing his KScript Editor with us because it has made writing and maintaining the source code so much easier. I also want to thank both Nils Liberg and Frederick Russ for taking over hosting of the SIPS Download and Demo pages since Theo's web site became history. I also need to thank Nils for his willingness to take over maintenance of SIPS during my 'on again', 'off again' health problems. While my health still continues to be somewhat uncertain, the addition of a pacemaker plus a recent heart 'reboot' has 'recharged' me enough so that I was able to generate the **V1.5** series and now this **V2** update. The Good Lord willing, I may actually get this thing done and be able to start using it to do **something musical** for a change (instead of fiddling around with all these ones and zeros all the time). ;-)

Finally, my heartfelt thanks to all of you on the forums who have posted so many kind remarks and encouraging thoughts. I also appreciate the many emails and PMs expressing your concern for my health situation and of course I especially appreciate your prayers on my behalf. I'm sure that, in no small way, they were responsible for my being able to provide these new versions of SIPS and so it is my sincere hope that you will enjoy and benefit from this latest update.

May God Bless all of you,

Bob

1.0 An Introduction to SIPS

The Solo Instrument Performance Suite, **SIPS**, is a family of three K2/K3 Scripts that are designed to work together harmoniously (when chained) to provide a useful number of functions often referred to as performance effects. The three member scripts are referred to as the **SIPS Articulation Script**, the **SIPS Legato Script** and the **SIPS Vibrato Script** which will be referred to hereafter as the **SAS**, the **SLS** and the **SVS** respectively.

SIPS is intended to be used with solo, monophonic instruments and, as such, **SIPS** is designed with a **Solo-Mode Control Logic**, similar to that provided with many synthesizers. This doesn't mean that you can't use **SIPS** with sectional or layered samples (when that might be appropriate), it just means that *ordinarily*, you won't be playing chords. However, for playing an occasional chord, **SIPS** allows you (under MIDI control) to disable the Solo-Mode Logic at any time you wish (see **Section 4.4**).

The three scripts are designed to be installed in three consecutive KSP script slots, normally slots 2, 3, and 4 (leaving slot 1 for the **optional starter script**, see **Section 1.1**). The scripts should be installed in the order **SAS**, **SLS**, and then **SVS**. Some of the newest features of the **SLS**, such as the **DFD Offset Mode** and **Release Sample Triggering** arise from a cooperative effort between the **SLS** and **SAS**. In general, the three scripts have many inter-dependencies and they should be kept together as a set even if you don't need or want all the effects and features provided. **Rather than remove one of the scripts from the set, it is better to leave it installed and simply disable its function.**

To facilitate this, each of the three scripts has a 'mode menu' that allows you to disable its function (under MIDI control if desired). By using **SAS Off**, **SLS Off**, or **SVS Off**, you can disable the corresponding script's effect but **SIPS** will still be able to retain all the essential inter-communication needed for proper operation. **CAUTION: Don't use the KSP's Bypass button for this purpose.** Using the **KSP Bypass** button or removing one of the **SIPS** scripts can cause a breakdown in intercommunication and may cause unpredictable results.

Also be careful about using other scripts with **SIPS**. While all member scripts of the **SIPS** family are designed to work together when they are chained in the stipulated order, **this is not true of scripts in general**. Scripts often interfere with each other, sometimes in strange ways. So generally **you should not** put any scripts in front of or after **SIPS** member scripts unless you know for sure that they are compatible with **SIPS** (and each other).

1.1 About the Starter Script

When the **SIPS 2** scripts are first loaded, their control panels will not appear until the first MIDI CC event is received. This 'startup' situation will also occur anytime you reload or recompile one of the scripts. In addition, it will occur when you save the **SIPS Trio** with an instrument and later reload that instrument. When any of these actions occur, the affected script (or scripts) will display a prompting message asking you to move a CC.

However, if you install the special **SIPS Starter Script** in slot 1 (ahead of the **SIPS Trio**) and then save your instrument, when you reload the instrument, the **SIPS** scripts will be 'started' automatically without you having to wiggle a CC. The starter script will also 'auto-start' the trio whenever you recompile one of the scripts (by hitting the Apply button). The only situation where the starter script can't provide the needed CC event is when you reload one of the **SIPS** scripts as an **.nkp** file. For that situation, you will still have to wiggle a CC to restart the script.

Some users have written custom 'front-end' scripts for the **SIPS** family and if you have done this (or intend to do such a thing), you should include the starter script code in your front-end script. The starter script code is trivial (as can be seen by inspecting the supplied source code) but it will nicely eliminate most of the inconvenience of the K2 startup situation. Alternatively, you may want to use the new **Super Bender Script** in slot one. Besides providing the **Super Bender** function (described in **section 4.14**), this script also includes the starter code.

1.2 The SIPS Articulation Script

The **SAS** allows you to establish up to 64 articulations organized as 8 banks of 8 articulations each. Each articulation may utilize one or more Instrument Groups in various powerful and flexible configurations. Articulations can be selected in real time using two sets of keyswitches. One set of keyswitches (which can be positioned anywhere on your keyboard) selects the articulation's bank and a second set of keyswitches (which can also be positioned anywhere on your keyboard) selects one of the 8 articulations within the selected bank. Alternatively, or in addition to this, you can select any one of the 64 articulations using a MIDI Program Change command.

Up to three group types can be assigned to an articulation: **Normal**, **'Inside'**, and **Release**. Each of these group types can be treated differently by the **SLS**. Additionally, any of these group types can be layered and sub-selected by other means (such as with Mod-Wheel velocity layer crossfading). There are also two kinds of articulations: **Fixed** and **Chained**. The groups within a chained articulation can provide natural (ie sampled) variations. These variations can be selected manually via keyswitching or automatically with several kinds of triggering options. Auto sequencing of the variations can be simple round-robin or a special full-cycle random mode. The **SAS** can also provide up to **8 TKT** variations for each articulation and/or natural variation.. These variations are also under your complete control at all times with a number of manual and automatic TKT selection options similar to that of the **Ultra TKT** script. In addition, for Chained articulations, the TKT variations can be combined with the natural variations in a variety of ways that are bound to enhance the realism of your Virtual Instrument compositions.

The **SAS** works seamlessly with the **SLS** and **SVS**, allowing you to smoothly change articulations and/or variations even within a legato phrase (for each note of the phrase if desired). The **SAS** is also provided with a special **Audition/Setup** control panel that makes it easy for you to audition and prepare an instrument's Groups for use with the **SAS**.

1.3 The SIPS Legato Script

The **SLS** provides a very 'musical-sounding', simulated legato effect by 'connecting' the note transitions within a legato phrase. This is accomplished with a multi-stage amplitude crossfade and complementary pitch warping of each overlapping note pair. The **SLS** provides a lot of parameters that can be fine-tuned to tailor the legato effect for almost any library instrument and many of these parameters can be controlled in real time via MIDI for a very convincing legato sound.

SIPS 2 further extends the capabilities of the **SLS** with the addition of a new portamento mode. This *super bender* mode allows you to produce very realistic-sounding, wide-interval pitch glides and/or bends. The **SLS** accomplishes this feat by using equal-power crossfading of adjacent samples to provide smooth, formant-corrected bends, thus allowing the **SLS** to produce very realistic glissandos of almost any kind.

For **SIPS 2**, the **SLS** working in concert with the **SAS** can also optionally provide release sample triggering and a special new **DFD Mode** has been added to the **Offset** menu choices. While Kontakt still provides no convenient means for a script to control sample-start offset in **DFD** mode, **SIPS 2** can automatically play a different set of samples for the 'inside notes' of a legato or portamento phrase. Thus you can 'clone' your sustain sample groups, trim the attacks from the front and assign these groups in the **SAS** as an **'Inside'** type. Done properly, this will allow you to obtain the same quality legato sound in **DFD** mode as you can in **Sampler** mode.

1.4 The SIPS Vibrato Script

The **SVS** provides a flexible set of pitch and volume modulation functions that can produce a very convincing and musical-sounding vibrato effect. You can easily customize things like the ratio of volume to pitch modulation, the modulation rate and waveshape, etc. Many 'humanizing' parameters are available to 'break up' the usual mechanical monotony of synthesized vibrato. The time profile of vibrato intensity can be controlled directly with a MIDI Controller or via a customizable envelope. In addition, most of the key parameters can be MIDI controlled for even more convincing realtime variation.

2.0 SIPS Common Features

Details specific to each of the three **SIPS** scripts will be discussed subsequently in their own separate sections of this Guide. However, there are a number of features that are common to the **SIPS** family and these will be presented in this section.

2.1 Setting Instrument Range

For things like TKT offsets and keyswitches to be handled properly, each member script of **SIPS** needs to know the playable range of the associated instrument. And, since all three scripts are associated with the same instrument, all 3 scripts should be set to the same instrument range. Because the **SAS** is installed ahead of the **SLS** and **SVS**, common settings such as **Instrument Range** are set via the **SAS**'s control panel which then 'broadcasts' your settings to the rest of the suite. Thus, the **SLS** and **SVS** scripts will automatically be set to the same range.

To set the **Instrument Range**, use the following procedure. In the upper right-hand corner of the **SAS** control panel, click the button labeled **Set Inst Range** and follow the prompts (that appear on the button), ie hit the lowest instrument key followed by the highest instrument key. If you do this correctly, the display box beneath the button will show the keyboard range that you have set. These values will also be broadcast to the remaining scripts in the suite and they should now also show the same values for **Low Key** and **High Key**. The easiest way to enter this range data is to use the K2 keyboard display. After clicking **Set Inst Range**, simply click the lowest 'blue' note followed by the highest 'blue' note on the K2 keyboard. **NOTE: It is important that you set the Instrument Range before you attempt to setup your keyswitch ranges or use any TKT variations.** An incorrect setting for **Instrument Range** can result in several kinds of incorrect behavior, some of them rather subtle.

2.2 Assigning MIDI Controllers

Member scripts of **SIPS** allow many of their parameters to be controlled in real time via MIDI. When a panel parameter can be controlled by a **MC**, there will be an associated *assignment-button* nearby. When a **MC** is assigned to control a parameter, the *assignment-button* will be **lit** and will display the parameter name along with the **MC** assigned to control it. For example, when **CC#21** is assigned to control **XTime**, the 'lit' button will display '**XTime CC# = 21**'. When no **MC** is assigned, the button will be dark and will indicate '**XTime MC=None**'.

To change a **MC** assignment, slowly *double-click* the button to open a drop-down list of **MCs**. Scroll through the menu to find the **MC** you want and click on it. This will close the menu and re-display the assignment-button (with your new **MC** assignment now shown on it). **CAUTION: When the instrument editor 'wrench' is open, the KSP handles I/O rather sluggishly** (especially if the script editor's text window is open). **So instead of double-clicking, you may need to space the pair of clicks a little in time.** Actually, the **1st click** brings up a drop-down button labeled '**- MC Menu -**' and the **2nd click** opens the menu. The drop-down menu displays the standard **MIDI CC** numbers from **0 to 119** (along with a short description of their 'customary function' as assigned by the MMA). You are at liberty to assign any of these **CCs**, regardless of their 'customary' usage, but it is your responsibility to be sure there is no conflict of such assignments within your system.

Besides the standard **CCs**, **SIPS** also allows you to assign certain 'special controllers' to some parameters. The special controllers include the **Pitch Wheel**, **Velocity**, and **Aftertouch**. These are available in all drop-down **MC Menus**, however, they cannot be assigned to every **SIPS** parameter. Generally, the special controllers can be assigned to any continuous parameter such as **XTime** or **BTime** but cannot be assigned to things like Mode Control Menus. If you try to assign one of the special controllers to an incompatible function, **SIPS** will simply de-assign it and display a flashing error message for about 5 secs (on K2's Status Line). So, once again, to assign any **MC**, simply click on it in the drop-down menu. To de-assign MIDI control of a parameter, click on the menu line labeled '**Deassign Controller**'.

If you want to assign some physical controller on your keyboard but aren't sure what **CC#** it represents, simply move the slider or knob and then click on the menu line labeled '**Last CC Moved**'. Of course this will only work properly if there are no other **CC** events in the MIDI stream at the time. Note also that this feature only applies to **CCs** and not the special controllers. When you assign the **Pitch Wheel** for control in **SIPS**, it functions the same either side of its center rest position. If you raise the **PW** (from center) to its maximum position it behaves the same way as it does if you lower it (from center) to its minimum position. Either of these movements has the same control effect as moving one of the general **CCs** from min to max. Thus the **PW** behaves much like the general **CCs**, swinging from minimum to maximum but it does this either side of center and thus, like the other **CCs**, there is no negative region.

2.3 SIPS Assignable CCs

You may have noticed that the **MC** menus in **SIPS** only cover the **CC** numbers from **0 to 119**. The MIDI specification reserves the remaining controller numbers from 120 to 127 as the 'so called' channel-mode messages which are as follows:

120	All Sound Off
121	Reset All Controllers
122	Local Control On/Off
123	All Notes Off
124	Omni Mode Off
125	Omni Mode On
126	Poly Mode Off
127	Poly Mode On

Of these, K2 only seems to respond to **CC120** and **CC123** (and these can be disabled in Instrument Options). But K2 allows you to assign any **CC** from 0 to 127 as modulators (or automators). However, use of the **CCs** above 119 for ordinary control purposes **should be avoided**. This allows these higher **CC** numbers to be used for 'special' purposes without conflict. The **KSP+ Multiscript** uses **CC122** for the timer 'interrupt' and **CC124** and **CC125** as proxies for **Aftertouch** and **Program Change** messages. **KSP+** also uses **CC127** for its configuration utility. Finally, the **SAS** uses **CC126/CC127** for Group-Start articulation control which leaves only **CC121** (which we may some day want to use for its intended purpose). So, the bottom line here is that you should **avoid using CC120 to CC127 for routine control purposes**. **SIPS 2** helps enforce this discipline by only including the **CCs** from **0 to 119** in its **MC Menus**.

2.4 Setting a MIDI Controller's Range

When you assign a **MC** to a panel knob, the **MC** can swing the knob over its full range if desired. However, oftentimes you may want the **MC** range to cover only a fraction of the knob's full range. When you set the **MC** to cover a smaller range, each of the 127 steps of the **MC** can effect a smaller change and thus can provide better resolution. To set such a subrange for an assigned **MC**, follow this simple procedure. Set the **MC** to minimum and then move the panel knob itself to the value you want associated with the **MC's** minimum. Next, set the **MC** to maximum and then move the panel knob to the value you want associated with the **MC's** maximum. That's all you need to do.

As an example, let's say you have assigned **CC21** to control the **XTime** parameter (whose full range is from 0 to 1000ms). Now, suppose you want **CC21** to only cover the **XTime** sub-range of **150ms to 250ms**. First, set **CC21** to its minimum position and then turn the **XTime** panel knob to **150ms**. That takes care of setting the minimum, so next set **CC21** to its maximum position and then turn the **XTime** knob to **250ms**. Now, whenever you move **CC21** from its min to max position, the **XTime** knob will rotate from **150 to 250ms**. If you like, you can even reverse the two knob settings (with the min above the max) and the **MC** will then provide **inverse control**.

While setting a **MC** subrange is generally easy, **Pitch Wheels** are spring loaded and keeping them solidly at max or min while setting the panel knob to the desired value can be a little more challenging. However, there is an alternate way you can use to set a subrange for the **PW** that may be physically easier. Temporarily, assign the panel knob to some other, easy-to-move controller. Use this temporary controller to set the min and max of the subrange and when you have it properly set, reassign the knob to the **PW**. The **PW** will then *inherit* the same subrange you set for the temporary controller. Similarly, if you want to assign **Velocity** or **Aftertouch** as a controller with a subrange setting, use another controller to set the subrange and then reassign control to **Velocity** or **Aftertouch**.

2.5 Setting User Preferences

Another new feature being introduced with **SIPS 2** is the ability for you to make certain optional choices about how **SIPS** behaves. These *preferences* are set from the **SAS** but, like **Instrument Range**, the settings are shared with all **SIPS** scripts as needed. To inspect or change **User Preferences**, open the **SAS Panel Menu** (the button just under the Instrument Range Display) and click on **Preferences**. This opens a dialog (see **Figure 2-1**) containing **16 DIP Switches** (1) and a pair of action buttons (2), (3). Each **DIP Switch** provides a 2-way choice of how the associated function will behave. You can easily move any of the switches up to their enabled position (4), or down to their disabled position (1), by dragging the switch up or down with your mouse.



User Preferences Dialog

Figure 2-1

Currently, only the first seven switches are in use but new preferences may be added from time to time with future updates of **SIPS**. The current functions for DIP switches 1 to 7 (as also displayed in the **Help Window**) are tabulated below.

Sw1	^ Disable CC64 Sustain Function	(4.3)
Sw2	^ Disable SAS MIDI Menu Control	(3.40)
Sw3	^ Increment MIDI Program Change Number from Keyboard	(3.28)
Sw4	^ Limit Playback of Release Samples to 3 seconds	(4.12)
Sw5	^ Reset Random Generator when Articulation or Var Mode is changed	(3.38)
Sw6	^ Display Articulation & KSw status on K2's Status Line	(3.29)
Sw7	^ Hide Keyswitch Display on K2's Keyboard (Pink Keys)	(3.23)

The ^ character (displayed to the right of the switch number) indicates that the described function applies when the switch is in the **Up** position (enabled). To see the above 'help' text, simply click on the switches one by one. Each time you click on a switch, a short summary of its function will be displayed in the **Help Window** (5). You'll find a more detailed description of these preferences in the sections indicated to the right in the above tabulation.

After you have set the switches the way you want them, simply click on the **Accept Settings** button (2). This will close the dialog box and the preferences you have just chosen will be in effect. If you change your mind before 'accepting' your new settings, you can restore the original settings (from when you first opened the dialog) by clicking the **Cancel Changes** button (3) followed by clicking on the **Accept Settings** button (2). This will restore the original configuration.

2.6 SIPS Preset System

The **SLS** and **SVS** both have a multi-purpose **Preset** menu which contains a number of **Built-In Presets** as well as space for you to store up to 10 **User Presets** of your own custom design. The preset menu also contains a number of action ‘**commands**’ (specifically **Save As**, **Rename**, **Export**, and **Import**). These commands will be discussed as we go along but for now we want to focus on the **Presets**. The **Built-In Presets** are usually named for the type of instrument or instrument family that the preset was designed to be used with. These presets were designed and installed by the developers of **SIPS**. However, since there is a good deal of variance from one sample library to another, as well as marked differences in the sound you may be trying to achieve, you should consider the **Built-In Presets** merely as ‘starting-point templates’ that you will want to customize for your particular library and application. Because of this, built-in presets will not generally contain values for every control-panel parameter. For example, the built-in presets do not contain values for the assigned **MCs**, or their subranges. Rather, built-in presets contain only a subset of the control panel parameters that are referred to as the ‘**Key**’ parameters. The remaining panel parameters are referred to as the ‘**Extended**’ parameters.

To recall a **Built-In Preset**, just click on the **Preset Menu** drop-down button and then click on the desired **Preset Name** in the menu. The **Key** parameter values for the selected preset will instantly be copied to the control panel. The **extended** panel parameters however **will be left unchanged**. Once you recall a preset, you can tweak the values and/or add MIDI control, etc to obtain the desired effect. If your customized panel is intended to be used only with one particular instrument, you can simply re-save the instrument. The scripts, along with the customized panel, will then be saved with it. The next time you load the instrument, it will be recalled just as you left it (including the extended parameters). However, if you then recall another preset, your customized panel will be overwritten and can be restored only by reloading the re-saved instrument.

2.7 User Presets

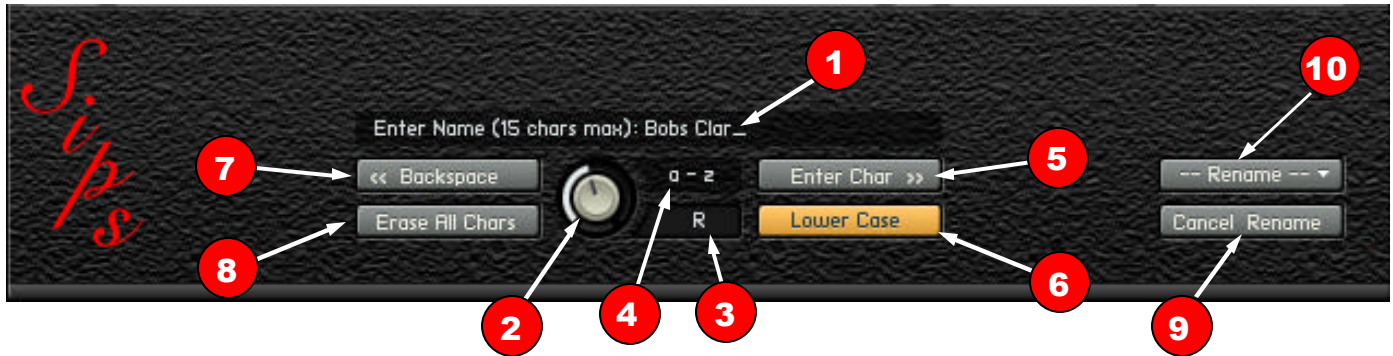
If you develop one or more customized presets or you would like to keep the control panel settings for several instruments together in one place, you can save your control-panel settings as **User Presets**. To save a **User Preset**, first set the control panel the way you want it and then open the **Preset Menu** and click on the – **Save As** – *command*. You’ll find this command between the **Built-In** and **User Preset** lists. After clicking on **Save As**, the menu will close and the button will read - **Save As** -. Now, open the menu again and click on one of the user preset locations (initially named **My Preset #1**, **My Preset #2**, etc). When you do this, the menu will close again with the user preset name on the button. The panel settings are now stored in the **User Preset** and this preset can be recalled just like any of the **Built-In Presets**. However, before you end your K2 session, **you must re-save the script** first, either as a **.nkp** file or with the instrument as a **.nki** file. If you don’t resave the script, your new **User Preset** will not be there the next time you load the script.

While **User Presets** can be recalled to the control panel just like any of the built-in presets, there is one notable difference. **User Presets** store **all** the panel parameters (including the **extended** parameters) whereas the built-in presets store only the **key** parameters. Thus, as already stated, when you recall a **built-in preset**, the **extended** panel parameters **are unaffected** and retain whatever values they were set to last. However, when you recall a **User Preset** the **extended** panel parameters **are also recalled** (as they were saved).

There are several things you should note about the **Save As** operation. First, the **Save As** command is only active until you select the following **User Preset** (the save destination). If you want to save another preset you will have to click the **Save As** command again. Second, if the **User Preset** slot you save to already has a preset (previously saved there), **it will be overwritten** and replaced by the new preset. Finally, if you click **Save As** and then change your mind, simply click on any **Built-In Preset** (or any other menu command) to cancel the pending **Save As**. As originally supplied, all the **User Presets** are ‘**empty**’. If you attempt to recall a **User Preset** that’s empty, the control panel will remain unchanged and K2’s status line will blink ‘**Preset Empty**’ for about 5 secs.

2.8 Naming User Presets

If you use the default names for the User Presets (ie My Preset #1, etc), you will have to keep track of what you have saved elsewhere. However, to avoid such separate record keeping, you can rename any **User Preset** (even an empty one). To rename a **User Preset**, open the **Preset Menu** and click on the '**Rename**' command (located just after the last **User Preset**). This will open the **Name Edit Dialog** as shown in **Figure 2-2**.



Name Edit Dialog

Figure 2-2

When renaming a user preset, the new name is entered in the **Name Edit Window** (1) character by character as selected by the **Alpha Knob** (2). As the **Alpha Knob** is rotated (by dragging up and down over it), you can choose one of 42 chars spread over 4 zones. The selected character is always displayed in the knob's **Data Window** (3) while the **Active Zone** (4) is displayed just above the **Data Window**.

Starting from minimum (full CCW position), the four zones are: **Space**, **Alphabet**, **Digits**, and **Symbol Set**. The **Space Zone** has only the one blank character. The **Alphabet Zone** contains the letters **A - Z**. The **Digits Zone** contains the numerals **0 - 9** and the **Symbol Set Zone** currently contains the five symbols **# < > + -**. It's very easy to find the desired character by dragging rapidly at first while watching the **Active Zone** display until you get to the appropriate zone. Then slow your dragging speed as you approach the desired character in the data window. When you are very near you can hold down the Shift key as you drag for more precision and it will be easy to 'zero in' on the desired character. With just a little practice, you can easily get to any desired character quickly.

When you have selected the first character that you want to enter, click on the **Enter Char** (5) button and the selected character will be entered into the **Name Edit Window** (1) and the blinking cursor will move to the right; ready for you to select and enter the next character. When selecting an alphabetic character, the **Data Window** (3) always displays the upper case form (because lower case letters are less readable). However, if the **Lower Case** (6) button is **active**, when you click **Enter Char** (5), the corresponding lower-case character will be entered. This is emphasized by the zone display (4) of '**a - z**' when the **Lower Case** (6) button is **active**.

If you change your mind about a character you have entered (or you have used the wrong case), you can click on the **Backspace** (7) button to delete the character to the left of the cursor so you can re-enter it. You can also use multiple presses of the **Backspace** button and each click will delete another character until you reach the left margin of the name-entry field. If you want to delete the entire name string that you have entered and start over, you can simply click the **Erase All Chars** (8) button.

You will most often want to start your name with an upper case letter and then continue with lower-case letters after that. Therefore, when you first open the **Name Edit Dialog** (or whenever you **Backspace** to the left margin), the **Lower Case** button will automatically be **deactivated** and after you enter the first character, the **Lower Case** button will automatically be **activated** for you. If this turns out to not be the correct choice for your situation, you can simply override it by manually clicking the **Lower Case** button to set it the way you want it.

When you have entered all the characters of your new name, simply click the **Rename (10)** button and then click on the **User Preset** name you want to replace. **Unfortunately, your new name will not actually appear in the menu immediately because of KSP limitations.** The easiest way to force the name to update (without having to save and reload the script) is to open the KSP text editor and click the **‘Apply’** button. Once this is done, you will see your new preset name in the **Preset Menu**. However, before you end your K2 session, you will still have to resave the script (either as a **.nkp** or with the instrument as a **.nki** file). Otherwise, the next time you load the script your new names (as well as any other changes you make) will be lost. If you are going to rename several presets, you need not hit **‘Apply’** nor resave the script until you have made all the changes (unless you want to see the changes right away). Finally, if you change your mind about renaming a **User Preset**, you can simply click the **Cancel Rename (9)** button and the Dialog will close without renaming anything.

2.9 Updating Your Custom Presets for SIPS 2

You may have saved one or more **.nki** Instrument files (or **.nkp** script files) with customized control panel settings and/or a set of custom **User Presets**. If so, when you update to **SIPS 2**, you’ll most likely want to transfer all your custom presets and control panel settings from your prior version of **SIPS** (preferably without having to painstakingly write down all the settings on paper and then re-enter them all over again into the new version). **.nki** and **.nkp** files with custom presets can easily be updated to the current version of **SIPS** by using the **Auto-Import** feature built into **SIPS**. However, please note that the **Auto-Import** function for **SIPS 2** works a little differently than it has in the prior **V1x** series. For **V2** the process requires an additional operation (when updating from **V1x**) with a special **V2 Updater Script**.

Using the **Auto-Import** feature requires that you know how to place a copy of various **K2-Ready** source files into your computer’s clipboard. If you are unfamiliar with what this means, before proceeding with this discussion, you should read **sections 7.0** and **7.1**.

In general there are two kinds of files that you will want to upgrade, **.nkp** files and **.nki** files. **.nkp** files contain only a single script, either the **SLS** or the **SVS** (the **SAS** is new with **V2**) having a control panel with a full set of parameters (**key** and **extended**) and possibly a set of up to 20 **User Presets**. On the other hand, **.nki** files contain both an instrument, and possibly both the **SLS** and the **SVS** that were saved with the instrument. For such **.nki** files, both the **SLS** and **SVS** will have a full control panel saved with them and in addition the scripts may also contain up to 20 **User Presets** each. For **V105** (and **V1051**), **User Presets** contain only **key** parameters (just like the built-in presets) but starting with **V110**, **User Presets** contain the full set of parameters (**key** and **extended**).

Starting with **V2** the number of **User Presets** has been reduced from 20 to 10 (to make more room for future enhancements). Therefore, if you want to transfer more than the first 10 **User Presets** from a **V1x** series script to **V2**, you will first need to use the **Export** feature of your **V1x** script to transfer the **upper 10 presets** to the **lower 10 presets** of a second copy of the script. The process required to do this is outlined in **section 2.13**.

As an example of how to use the **V2** update process, I’ll only describe the procedure you would follow for updating an instrument that contains the **V1.05 Legato Script**. However, you should easily be able to translate this example to updating **.nkp** files, other versions, the **Vibrato** script, etc. Begin by loading the instrument’s **.nki** file. Then, launch the **KScript Editor** (V1.25.6 or higher) and load the two **KR** files named: **SIPS-Legato V205_KR.txt** and **SIPS-Legato-V2U_KR.txt**. Where **V205** is assumed to be the current version of **SIPS 2** and **V2U** is the **V2 Updater** script. Select the **V2 Updater** tab in the **KScript Editor** and then use **Ctl-A, Ctl-C** (Windows) to select and copy the **Updater KR** source to the Windows clipboard. Now open the KSP text editor and use **Ctl-A, Ctl-V** to select the **V1.05** source text and replace it with the **V2U** source text. Now hit **Apply** and you should see a prompting message indicating that you have successfully completed the first step by converting **V1.05** to the interim **V2U** format.

Now, in the **KScript Editor**, select the **V205** script and use **Ctrl-A**, **Ctrl-C** to select and copy the **V205** source into the windows clipboard. Then, click on the KSP text editor to give the focus back to it and use **Ctrl-A**, **Ctrl-V** to replace the **V2U** text with the **V205** text from the clipboard. Now hit **Apply** again and the new **SIPS 2** control panel should come up with all the **V1.05** panel settings intact. Note also that the first **10 User Presets** from your **V1.05** instrument will also have been transferred to the **V2** script. Now simply resave the instrument as a **.nki** (under the same or a different name if you wish to keep the old version also).

Please note that the extra step involving the **V2U** file will only be necessary when updating from the **V1x** series to the **V2x series**. For future **V2** updates (within the **V2x series**), **this extra step will not be needed**. The **Auto-Import** feature of **SIPS** involves the use of a number of format converters (one for each prior version of **SIPS**) and the amount of code space devoted to these converters was beginning to get excessive. Instead of carrying all this code forward throughout all subsequent revisions of **SIPS**, it was decided to break clean with **V2**. Therefore, **V2** (and all subsequent releases in the **V2x series**) will only be able to update from another script from the **V2x series** itself. This greatly reduces code size and eliminates a lot of seldom-used code from being carried forward. However, I wanted to provide a way for you to transfer panels and presets from the **V1x** series without having to manually re-enter all the settings. This was accomplished by writing the **V2 Updater Script**. The **V2U** script can update any of the **V1x** series scripts to an interim format that is then acceptable to any **V2x series** script. So, while the **V2 Updater** scheme involves a little extra work, it's a one-time operation and it frees a lot of code space that can be put to better use. Also, as stated above, subsequent **V2x series** updates will not require the extra step with the updater script. So, hopefully, this interim format scheme provides the best overall compromise between backward compatibility and future code savings.

2.10 Auto-Import Problems

When you 'replace' the source code for a **SIPS** member script and then hit '**Apply**', the code that executes tries to verify that what you are trying to do is valid. To do this, the **Auto-Import** algorithm checks certain attributes of the 'old' source code and the 'new' source code. These attributes are then compared to determine if the import will be valid. Using the **V2 Updater**, valid upgrades can be made from any earlier version of **SIPS** including **V105/1051**, **V110**, or **V150/151**, however you must also be sure that the old and new scripts belong to the same member. In other words you can't import from the **SVS** to the **SLS** or vice versa. If **V2U** is the importing script and the 'old' script is any older or newer **SIPS** script, all illegal combinations **are detectable** and will be reported by the updater (an error message box appears and indicates that an **Auto-Import** error has occurred. But, if you try to update to **SIPS 2** from some other script which is not part of the **SIPS** family, the problem **may or may not be detected** and, the outcome may or may not be OK. So, play by the rules. ;-)

Generally, if the 'old' script has no persistent variables named the same as in **SIPS 2**, trying to update should be relatively harmless. However, if one or more persistent variables in the 'old' script should happen to have the same name as some of the persistent variables in **SIPS 2**, trouble may occur. If a name match happens in certain key areas, the **Auto-Import** logic may detect it and warn you but chances are that such will not be the case. Also, a recently discovered flaw in K2's implementation of persistent arrays reveals that if the new script tries to downsize a persistent array that contains non-zero data in the excess elements, very serious crashes can occur due to illegal memory-access errors.

Therefore, to be on the safe side, whenever you are updating to **V2** of the **SLS** or **SVS** (via **V2U**), be absolutely sure that the 'old' source file **is a SIPS file**. If you do this, the results will either be harmless or you will be warned with a suitable error message. On the other hand, if you want to install **SIPS 2** from a source file and you don't want to transfer your custom presets, use the **Fresh Install** procedure (see **section 7.1**). This will clear K2's persistence buffers and you will get an install where all settings will be initialized to their initial **default** values. A fresh install like this is the same as loading the original **.nkp** file for **V2** (the one included in the original **SIPS 2** package).

2.11 Extending V105 User Presets

User Presets in **V105** only contain the **key** parameters and not the **extended** parameters. For **V110** on up however, whenever you save a **User Preset**, **all panel parameters** are saved (including the **extended** parameters). So, after an **Auto-Import** from **V105**, **User Presets** do **not** contain the **extended** parameters (since they didn't exist in **V105**) and until you **resave** them in **SIPS 2**, these extended parameters are set to a special 'token' value to indicate that they are **undefined**. Thus when you recall a **User Preset** imported from **V105**, the **extended** parameters of the control panel are left unchanged (just as they would have been in **V105**).

However, after updating to **SIPS 2** you can update your imported **User Presets** (if you wish) to include the **extended** parameters. To do this, all you have to do is the following. First, recall an imported **User Preset** to the control panel. Then, set the **extended** parameters on the control panel to the desired values and finally, use the '**Save As**' command to resave the **User Preset** (and of course resave the .nkp or .nki). The next time you recall this **User Preset**, all panel parameters will be recalled (including the **extended** parameters).

2.12 Instrument Range Not Included In Presets

Instrument Range, **IR**, is no longer included with the extended preset parameters. Rather the **SLS** and **SVS** always get their **IR** setting from the **SAS**. Of course the **SAS** remembers your last setting whenever you save the script (either as a .nkp or as part of a .nki file). And, whenever you reload or recompile the **SAS**, it will broadcast its **IR** setting to the **SLS** and **SVS**. The **SLS** and **SVS** also 'remember' their last known **IR** when saved and reloaded. However, if these scripts are loaded apart from the **SAS**, their **IR** may not agree with the **SAS**. If you ever get into that kind of situation, merely reload or recompile the **SAS** (or overtly reset the **IR**) and the **SAS** will resync the **SLS** and **SVS IR** settings.

2.13 Import/Export of User Presets

When **V105** of **SIPS** was written, it was before double-buffering of persistent variables had been added to K2. Therefore an **Import/Export, I/E** feature was added to **V105** of **SIPS** to assist in the process of transferring presets when updating. Since this function is now provided by the **Auto-Import** feature added to **SIPS** (starting with **V110**), we no longer need the **I/E** function for its original purpose. However, starting with **V110**, **SIPS** has retained a slimmed-down version of the **I/E** function because it facilitates the creation of **User Preset Libraries** by allowing you to re-group and/or re-arrange panel settings from several different .nkp or .nki files. Since **User Presets** now 'remember' **all the panel parameters** and can be renamed, this ability can be very useful.

However, the **I/E** function is now intended to be used only to transfer **User Presets** between scripts of the same exact kind (both member type and version). Moreover, the **Export** command only transfers one **User Preset** at a time. There is no longer any **bulk-export** of all **User Presets** (as there was in **V105**) because that is most easily done via the new **Auto-Import** feature. To transfer **User Presets** (one at a time) from one script to another, position the receiving script in the next slot to the right of the sending script. Open the **Preset Menu** of the receiving script and click on the '**Import**' command (you'll find the **Import** command near the bottom of the **Preset Menu**). The menu will close and the button will display **Import** indicating that the script is ready to receive **User Presets**.

Next, open the **Preset Menu** of the sending script and click on the **Export** command. The menu will close and the button will display **Export**. Now, open the menu again and click on the **User Preset** you want to export. If the operation is successful, the Importing script will blink the message '**Import Done**' in the K2 status area. The exported **User Preset** is now available in the receiving script (including the preset's name) but you will have to hit **Apply** before you will see the new name. Of course you will also have to save the script if you want to retain the new name and imported preset for the next time you load the script. After you have exported a preset, the sending script will remain in the **Export** mode in case you want to send another preset (the receiving script will also remain in the **Import** mode). To send another preset, simply open the **Preset Menu** again and click on the next **User Preset** that you want to send.

When you have finished sending presets you can end the **Export** mode by selecting any built in preset. Note that imported presets are positioned the same as in the sending script (ie if you export the **5th User Preset**, it will be imported as the **5th User Preset**). Also note that if you try to export an empty preset, **nothing will be sent** and the K2 status area will blink **'Preset Empty'**. Needless to say, if you try to do an **I/E** operation between two non-compatible scripts, the data will not be transferred.

If you want to transfer a control panel, first save the panel to one of the **User Presets** using the **'Save As'** command, then **Export** that **User Preset**. With this general idea, you can use the **I/E** facility to load up to 10 instruments (one by one) and 'collect' their control panels as **User Presets** into one single **.nkp** file. In addition, while the **Export** command always transfers a **User Preset** to the same location in the import script as it was in the export script, you **can rearrange the order** in the new script (at least with some degree of latitude) because, until the importing script is completely filled, you can always move a preset by first recalling it to the panel and then using **Save As** to write it to a different user location (effectively copying it). You can then re-use the slot it came from.

Therefore, by using a combination of the **I/E** facility and the **Save As** command to jockey panels around, you can easily rearrange or subdivide your **User Presets** into two or more separate collections. If you have a **V1x** series script with all 20 user presets filled, you can use the process just outlined to move the highest 10 presets to the first 10 presets of another script. This will be necessary if you wish to transfer all 20 presets to **SIPS 2** because the **V2x** series only holds **10 User Presets** per script.

2.14 About the Built-In Presets

The current version of the **SLS** contains a total of 17 built-in presets and the **SVS** contains 10. The 'fresh-install' default presets are **Clarinet 1** for the **SLS** and **Basic Setup** for the **SVS**. **Basic Setup** makes a good starting-point for constructing new **Vibrato** presets. The remaining **Built-In** presets were developed by **Theo Krueger**, **Andrew Keresztes**, and **Martin Nadeau**. Their presets can be identified by their initials in the right margin of each preset in the menu. Thus, the legend **TK#**, **AK#**, and **MN#** are used for **Theo's**, **Andrew's**, and **Martin's** presets, respectively, with the numbers after their initials used to tie the presets to their corresponding **mp3** demos (available for listening by following the links on the **SIPS** download site). Each demo is prefixed with a tag that corresponds with the Preset annotation.

As mentioned previously, not all of the control panel parameters are stored with the **Built-In** presets. This doesn't mean that the remaining parameters (the **extended** parameters) are unimportant, just that they tend to be more Instrument-specific and/or dependent on individual preferences. For this reason, **Extended parameters ARE stored with your custom User Presets**. Of course all panel parameters are also saved with a **.nkp** or **.nki** file so when you reload the file, their last settings will be recalled. Also, when you upgrade to a newer version of **SIPS** and you use the **Auto-Import** feature, all the panel parameters of the old version will be transferred to the panel settings of the newer version (if appropriate).

Again, it must be emphasized that the **Built-In** presets should only be viewed as a reasonable starting point for several reasons. First of all, many of the presets were developed with a certain amount of generality in mind. As an example, **Theo** describes his presets this way: **"These presets are a little bit 'contained', meaning that I didn't overuse the bending or try to make them overly-expressive. I did that considering the various libraries and trying to make settings that would work well for most."** Theo's preset **demos** (mp3s) were basically done using just the raw preset (without CC riding) so that you can hear what the basic preset itself sounds like. However, to get the most from **SIPS**, you will want to customize these presets on an instrument by instrument basis and, you will want to add **MIDI** control to enable you to add your own personal touch of realism to the sound. **MIDI CC** assignments are not included with the built-in presets because everyone has a different set of favorites and will tend to use CCs in a somewhat different way. Furthermore, just assigning CCs doesn't 'play them' for you. Yet, **assigning CCs and 'riding them' during the performance is almost essential to producing a really convincing effect.**

If you use the built-in presets as is, along with a ‘set it and forget it approach’, you surely won’t get the most from **SIPS**. While the effect can still be surprisingly good, don’t expect miracles. On the other hand, if you are willing to add some **MIDI** controls and put some effort into using them musically, the results can be **extremely convincing**. The artful use of CCs can be likened to viewing **SIPS** as you would a musical instrument. If you want a quality, musical sound, don’t expect the scripts to ‘play themselves’. **Andrew** did his preset demos (mp3s) with CC riding added and I think you will find his demos are quite convincing. Here’s what Andrew said about how he used **SIPS** to make his demos: **“I use SIPS as an instrument.— it rarely stays static. I’m always riding the CCs. So as time goes on, I’ll get better at it. Plus, for strings (for me), it’s imperative to have varying gliss times and amounts”**.

Be sure you study the rest of this **User’s Guide** so that your future knob twiddling can be carried out with some solid technical knowledge backing it up. You will especially want to read the sections titled **‘Guidelines for Making Legato Settings’** and **‘Guidelines For Making Vibrato Settings’**. These sections offer a solid recipe for ‘cooking up’ presets. In addition, I asked Theo and Andrew to write about their experience with **SIPS** and to share with us their **tips and techniques for creating presets** and using **SIPS** under fire. So, be sure that you read what these **SIPS Pioneers** have to say, you’ll find their narratives in **sections 6.1 and 6.2** of this **User’s Guide**.

3.0 SIPS Articulation Script

3.1 Introduction

Using Kontakt's Group architecture, the **SAS** allows you to arrange and configure flexible combinations of instrument groups that can then be referenced in numerous ways as **Articulations** and **Variations**. Three types of groups: **Normal**, **'Inside'**, and **Release** can be assigned to **Articulations** which can contain any number and combination of these group types. Group types can be automatically enabled by various phases of the **SLS** and multiple groups of the same type can be layered to support such things as velocity-selected groups or Mod-Wheel Velocity Crossfading. Besides **Fixed** articulations, multiple groups can be assigned to an articulation in a **Chained** configuration. **Chained** articulations can be sequenced to provide 'natural' variations and the **SAS** can also generate up to eight **TKT** (synthesized) variations which can be used alone or in combination with natural variations.

Up to 64 articulations can be defined for each instrument and these articulations can be selected in realtime with **MIDI Program Change** messages, **Keyswitches**, or both. **Variations** can be chosen automatically or manually with a plethora of options. **Articulation** and/or **Variation** change requests can be made at any time (including within a legato phrase or portamento bend). The **SAS** includes a powerful **Setup/Audition** panel that allows you to audition **K2 Groups** or **SIPS Articulations** and provides everything you will need to easily configure any instrument for use with **SIPS 2**.

NOTE: Before you can use **SIPS 2** with an instrument, all the instrument's group-start options must be properly prepared. This process, called **Formatting**, can be done easily from the **Setup/Audition** panel. But, please note that formatting will completely remove all former group-start settings that may have existed. So, if your instrument has such settings and you want to keep them, first make a copy of the instrument for safekeeping. The procedure for formatting an instrument is covered in **section 3.9**.

3.2 SIPS Articulations

In **SIPS 2**, **Articulations** are essentially 'containers' for Kontakt instrument groups. For **SIPS**, there are three different **types** of groups and any number of such groups can be assigned to an articulation in various combinations. When an articulation is **active**, any number of its groups may be 'sounded' depending on various factors which will be discussed. Up to 64 different articulations can be defined for each instrument and you can easily switch between these articulations at any time. Only one articulation can be **active** at the same time but multiple groups, within the active articulation, **can** sound at the same time.

Articulations are referenced by a two-digit number, using only the digits from 1 to 8. For clarity, articulation numbers will be written with a hyphen between the two digits. You can view the two digits as representing a **Bank** and **Index** number. The first digit specifies 1 of 8 banks and the second digit specifies 1 of 8 articulations within that bank. For example, **Articulation 2-3** would be interpreted as the 3rd articulation in bank 2.

Before we discuss how to set up articulations and assign groups to them, it is important to understand what kind of group configurations are possible and how they function. As indicated in **section 3.1**, the 3 group **types** that can be assigned to an articulation are called **Normal**, **Inside**, and **Release** (the meaning of this terminology will become clear soon). Any number of each group **type** can be assigned to a given articulation but, it is not necessary that all three group types be included. However, regardless of how many or how few groups might be assigned to an articulation, **there must always be at least one Normal group**. Therefore, the simplest possible articulation contains only one **Normal** group as depicted in **Figure 3-1**. Note that the articulation itself is represented by the outer box while groups assigned to it are shown as smaller boxes within the articulation's 'container'. Thus, **Figure 3-1** indicates that **Articulation 1-1** consists of one **Normal** group whose K2 index number is **5**. Remember that K2 references its groups with index numbers from zero to one less than the total number of groups.

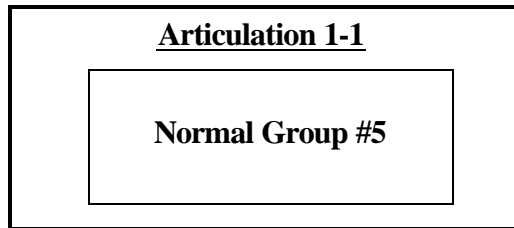


Figure 3-1

When **Articulation 1-1** is active, all played notes will be sounded from K2's **Group #5** since there are no other groups assigned to this articulation. Now, let's take a look at an articulation that contains one of each **type** of group as depicted in **Figure 3-2**.

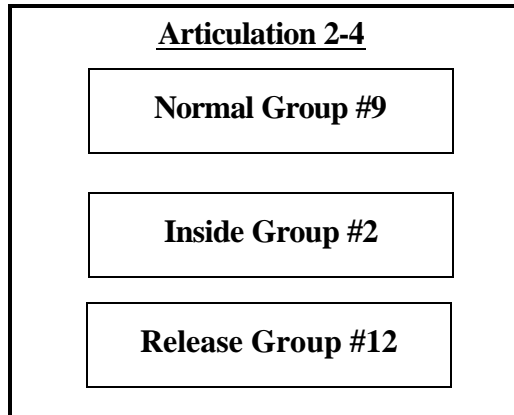


Figure 3-2

Thus, **Articulation 2-4** contains K2 groups 2, 9, and 12 with **Group #9** being assigned by **SIPS** as a **Normal** group, **Group #2** being assigned as an **Inside** group, and **Group #12** being assigned as a **Release** group. When **Articulation 2-4** is active and a note is played, the group that will be enabled to 'sound' the note is determined by the current state of the **SLS**.

First let's discuss the legato and portamento modes. The first note of any legato or portamento **phrase** will always be sounded from **Group #9**, the **Normal** group. If the **SLS** offset mode menu is set to the new **DFD Mode**, all the 'inside' notes of a legato or portamento phrase will be sounded from **Group #2**, the **Inside** group. However, if the offset mode menu is set to any mode **other than DFD Mode**, the inside notes will simply be sounded from **Group #9**, the **Normal** group. Of course the idea here is that **Group #2** will be an edited version of **Group #9** with the attack portion of the sample zones removed. Whether or not such **Inside** groups actually contain samples without their attack portion is up to you. So if you want to use some other kind of samples for **Inside** groups, you are at liberty to do so. However, since such a group type will be sounded only when playing the inside notes of a legato or portamento phrase, this **type** of group is called an **Inside** group. Please note however that **Inside** groups (regardless of what kind of samples they contain) will only be sounded when the **SLS** is in **Legato** or **Portamento** mode **and** the offset menu is in **DFD Mode**. In **Solo Mode** or **SLS OFF** mode, notes are sounded only from the **Normal** group.

If the active articulation contains a **Release** group, it will be triggered by the **SLS** at the appropriate time. For **Legato**, **Portamento**, or **Solo Mode**, release groups are only triggered when the **last note of a connected phrase ends**. However, if the **SLS OFF** mode is selected, the **SLS** merely passes all notes through without adding any effect other than the sustain pedal function. **Each** such 'thru-note' will always be played from the **Normal** group but each thru-note will also trigger the release group when it ends. For example, if you play a chord and release all the keys at once, the release samples for all the notes of the chord will be triggered simultaneously.

As with **Inside** groups, you can populate **Release** groups with any kind of sample you wish, however, it is assumed that such groups contain actual release samples. What the **SLS** does is to ‘trigger’ release groups by issuing a **play-note(n,v,0,0)** function call to the KSP. Note that this call uses the ‘play for ever’ option. When this option is used, K2 will play such samples until the sample itself runs out. Since **SIPS** has no idea of how long your release samples might be, the ‘play forever’ approach will guarantee that the entire release sample will sound, without being cutoff. However, if your release samples are looped and depend on a envelope to decay to silence, then the ‘play forever’ trigger will result in leaving release samples playing (silently, but still consuming resources). Looped release samples are a rarity and really should be converted into non-looped samples but, if you encounter such a situation, **SIPS 2** has a **User Preference** option, **Sw4**, which you can enable so that release samples will only be triggered for 3 seconds instead of forever.

Up until now, we have been looking at very simple group configurations but more groups of each type can be included in a **SIPS Articulation** and a general representation of this is depicted in **Figure 3-3**. When making up articulations, there are only **two rules** that you need to be aware of. **Rule #1: All articulations must have at least one Normal group** (all other groups and group types are optional). To emphasize that, **Normal Group #19** is shown in **Figure 3-3** with a thicker outline. **Rule #2: A group may be assigned to only one articulation**. However, if you have a situation in which you would like to include the same group in more than one articulation, you can accomplish this by using K2 to clone the group (several times if necessary), and then assign the clones to the other articulations in which you want to include that group.

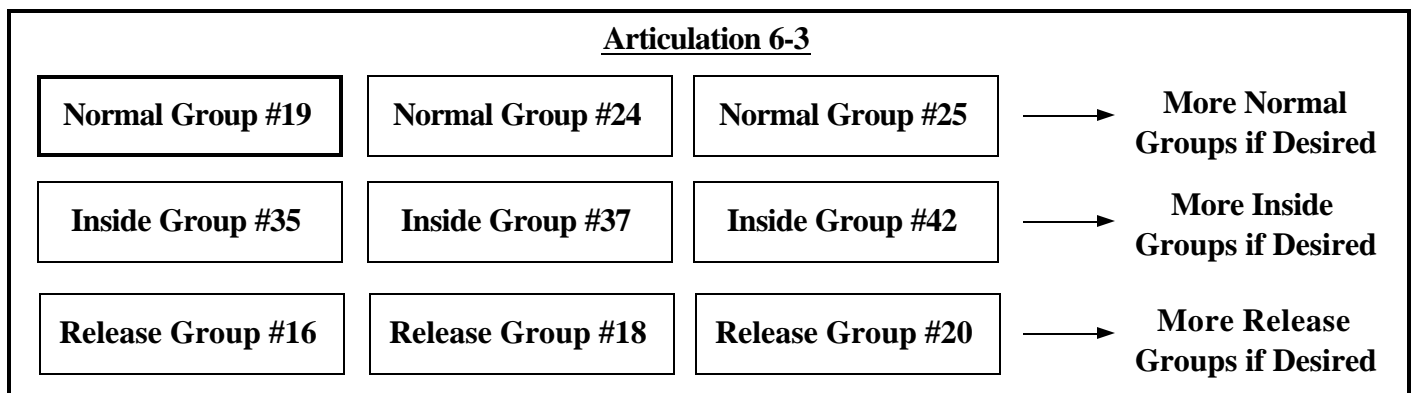


Figure 3-3

3.3 Fixed Articulations

Before we can discuss how **SIPS** deals with multiple groups of the **same type**, we need to introduce the **two kinds of SIPS Articulations**. And, before we can do that, we need to talk about **Variations**. Besides enabling groups by virtue of the active articulation and the current state of the **SLS**, the **SAS** also provides **Variation** control. Some libraries provide multiple sample sets (for the same articulation) that have subtle differences in attack and/or tonality, etc. Sequencing through such variants can help eliminate things like the dreaded ‘machine gun effect’ and generally add to the realism that can be achieved with a Virtual Instrument. In **SIPS** this kind of alternation is referred to as **natural** or sampled variation (as opposed to the synthesized TKT form of variation).

When a **SIPS Articulation** has no natural variation groups to cycle through, it is referred to as a **Fixed** articulation. The name comes from the fact that the groups that sound are determined solely by the current **SLS** state and not affected by any variation control conditions. On the other hand, when a **SIPS** articulation does contain **natural variation groups**, the groups that sound are **not Fixed** but also depend on variation control conditions. Since natural variations are ‘hooked up’ in a chained configuration that can be sequenced, such articulations are referred to as **Chained** articulations and will be discussed in **section 3.4**.

Let's return now to **Figure 3-3**. For **Fixed** articulations, additional groups of the **same type** are merely **layered**. So, if you assign three **Normal** groups to a **Fixed** articulation then all three **Normal** groups will sound at the same time. While this might be useful in some situations where you actually want to hear a layered sound from two or more groups, the primary reason that **SIPS** allows you to assign multiple groups of the **same type** is to support special velocity control techniques used by many libraries and/or that you might wish to implement yourself. A few examples should help to clarify this situation.

Suppose that the sustain samples for an instrument have been recorded at the three volume levels of **mf**, **f**, and **ff**. If these samples are all put in one group they will be stacked vertically. For example, the **ff** samples might be mapped to the upper velocity range of **109 to 127** while the **f** samples might be mapped as **89 to 108** and the **mf** samples would then be mapped to the velocity range from **1 to 88**. However, many library developers will instead use three groups and put all the **mf** samples in one group, the **f** samples in another group, and the **ff** samples in the third group. The **ff** group will still have the samples mapped vertically in the **109 to 127** velocity range (the remaining velocity range from 1 to 108 will be empty), the **f** group will still have the samples mapped vertically in the **89 to 108** velocity range, and the **mf** group will still have its samples mapped vertically in the **1 to 88** velocity range. For such a library, you need to 'enable' all three groups when this articulation is selected. But, even though all three groups are enabled, for any given note, only the group containing the matching velocity range will actually sound. Basically then, velocity acts as a sub-selector among the three groups that are enabled.

If **SIPS** only allowed one of each group type to be assigned to a given articulation, in order to accommodate the library configuration just described, you would have to merge the 3 velocity groups into one composite group. However, by simply assigning all three groups to the same group type, **SIPS** will **group-start** enable all 3 of these groups and a played note's velocity will sub-select only one of the three groups to actually sound.

Now, let's consider the way many libraries implement Mod-Wheel-controlled, **Velocity Crossfade**, **VXF**. Consider the same set of sustain samples recorded at **mf**, **f**, and **ff** as described in the prior example. Again, we will put the samples for each volume level in their own groups. However, we will now 'stretch' the mapping in each group to cover the full velocity range from **1 to 127**. If we then enable all 3 groups, any note played will sound all three volume samples at once. However, if we now assign the Mod Wheel as a volume modulator for each of these three groups, we can construct a set of Rescaling curves such that the Mod Wheel can be used to crossfade the three sets of volume samples as desired. For an actual example of such an instrument, take a look at the Kontakt 2 Library/01-VSL Kontakt Orchestra/01 Violin ensemble/02 Modwheel X-Fades/Violin ens 14 (sustain X).

To use this form of **VXF**, we must again enable all three groups when we want to use this articulation. However, in this case (as opposed to the velocity-selected example above), all three groups are always enabled no matter what the played note's velocity. But, the Mod-Wheel acts as a crossfade volume controller of the groups so that only the right amount of each group is actually heard. Of course this kind of articulation can easily be accommodated by virtue of **SIPS** ability to layer groups of the same type.

As mentioned earlier, besides easily accommodating existing library programming, you may want to use similar techniques to customize your own libraries. Furthermore, since V200 of the **KSP Math Library** now includes the special functions needed to do **equal-power crossfading of groups**, the group-layering feature of **SIPS Articulations** paves the way to add a **VXF** script to the **SIPS** family. Such a script would follow the **SVS** and would only have to deal with group volume control rather than the complications of individual note volume control. Handling the **VXF** function at the group level will allow for much looser coupling between the scripts.

You might also want to apply different group effects to different notes. In this case you can clone the group and then divide the notes between them so that some notes sound in one group and the remaining notes sound in the other. In this case group sub-selection is made by note rather than velocity. Of course you can also use a combination of note and velocity to sub-select a group from a layered set. Thus group layering can be quite useful.

3.4 Chained Articulations

Chained Articulations can be set up in either a non-layered or a layered configuration. The simpler, non-layered configuration can be diagrammed the same way as a **Fixed Articulation** (such as that shown in **Figure 3-3**). When building an articulation, you always have to start with the **Normal** group(s). When you assign these **Normal** groups, you need to specify if the articulation will be **Fixed** or **Chained**. Once defined as **Fixed**, you will not be able to add any **Chained** groups. Similarly, once an articulation is defined as **Chained**, you cannot add any **Fixed** groups after that. The only way to convert an articulation's classification is to first de-assign it and then re-assign it. So, if **Figure 3-3** represents a **Chained** articulation, it would mean that the **Normal** groups **19, 24, and 25** form a **Normal Chain** with a length of 3. Similarly, the **Inside** groups **35, 37, and 42** are an **Inside Chain** of 3, while **Release** groups **16, 18 and 20** form a **Release Chain** of 3.

To explain the difference in behavior between such a **Chained Articulation** and its **Fixed Articulation** counterpart, imagine that we enabled a round-robin sequencer that is triggered by every note. Next, suppose we set the **SLS** so its offset menu is in **DFD Mode** and then we play a five-note scale as a legato phrase. For example, let's say we play C, D, E, F, G with overlapping notes. Now if **Figure 3-3** represents a **Fixed Articulation**, we would expect the following. The C would be sounded from groups 19, 24, and 25 at the same time. The D, E, F, and G would each be sounded from groups 35, 37, and 42 at the same time. And, finally, when the G key is released, the release trigger would sound groups 16, 18, and 20 simultaneously.

Now, if **Figure 3-3** represents a **Chained Articulation** and we played the same five-note sequence, we would expect the following. The C would be sounded from group 19 only. The D would be sounded from group 37, the E would be sounded from group 42, the F would be sounded from group 35, and the G would be sounded from group 37. Finally, when the G key was lifted, the release sample would be sounded from group 18. Note that if we set the **SAS** for round-robin 'always', each played note advances through the chain but the first note of a legato phrase always comes from the **Normal** groups, while all the 'inside' notes of a legato phrase come from the **Inside** groups (provided that **DFD Mode** is enabled). However, please also observe that a note release does not advance the Round-Robin sequencer so the release sample is sounded from the same chain position that the last down-key was sounded from.

Therefore, the primary difference between a **Fixed Articulation** and a **Chained Articulation** is that the **Chained** articulation only sounds **one** of the chained groups per note. So, the chained groups are not layered as they would be for a **Fixed Articulation**. Also, we cannot state which group or groups will sound based only on the **SLS** state but rather, we must also know the current state of the **Variation** sequencer. I should also mention that **Variation Sequencing** in **SIPS** is very flexible, both as to when or whether to advance in the chain, and/or where in the chain to go next. However, for simplicity in trying to understand how **Chained** articulations behave, we will stick to the simple 'round-robin, advance on every note' sequence that we just described.

The next thing we need to discuss is how chain length affects things. First off, the length of the **Normal-Group Chain**, determines the cycle length for the sequencer. If the articulation has other group types, their chains can be longer or shorter but the cycle length of the sequencer is not affected by them. For example, suppose that the **Inside** chain for **Figure 3-3** was 5 groups long while the **Normal** chain length remains at 3 groups. In this case, the behavior would be identical to that just described. The extra 2 groups at the end of the **Inside Chain** would never be enabled. Now suppose that the **Inside Chain** had only groups 35 and 37 in it (deleting group 42 from the chain). Now when we play our 5-note legato scale, we could expect the following. The C would be sounded from group 19, the D would be sounded from group 37, the **E would be sounded from group 25**, the F would be sounded from group 35, the G would be sounded from group 37 and the release would still be sounded from group 18. Note the difference is that the E is played from **Normal** group 25 instead of the **Inside** group 42 (which no longer exists). Thus, when the sequencer 'runs off the end' of the **Inside Chain**, the corresponding **Normal** group will be used instead.

Now, while it is possible to construct **Chained** articulations with chain lengths that are different for each group type, it will seldom be of much use. For example, if the **Inside or Release Chains** are longer than the **Normal Chain**, the excess groups will never ‘sound’ so why have them? Also, consider something like a **Normal Chain** length of 3 and a **Release Chain** length of only one. If you play a legato phrase with such an articulation, no release sample will be played when the phrase ends, unless the phrase happens to end in the first position (of the round-robin). In most libraries, natural variations are for short samples and therefore don’t need to use release samples. If the samples are short staccato notes, it’s not too likely that you will be playing legato phrases with them either. However, if the variation samples can be played legato and/or do have release samples to go with them, then you will likely want each variation to be a complete set. Therefore, the most typical situation for **Chained Articulations** will either be only a **Normal Chain** or, equal chain lengths for each of the group types. However, the flexibility exists if you come up with a situation where you want to use unequal chain lengths.

Note also that a chain can be as short as one group (and as long as 16 groups). However, if you construct a **Chained Articulation** with only one **Normal** group, it would make little sense since it would be the only group that could be ‘sequenced’ to provide natural variations. In such a situation, using a **Fixed Articulation** will be functionally the same but it will be easier on CPU resources (chained groups have a little more overhead connected with them compared to **Fixed** groups). Nevertheless, the purpose of this introductory material is to discuss what can and can’t be done when configuring articulations (without passing judgement on just how useful some of these configurations might be).

3.5 Layered Chains

It is also possible to construct **Chained Articulations** with multiple chains of the same type. To illustrate this kind of structure, refer to **Figure 3-4**. In this figure, Groups 1, 2, and 4 form a **Normal Chain** of length 3. Similarly, groups 13, 14, and 15 form an **Inside Chain** of length 3, while groups 26, 27, and 28 form a **Release Chain** of length 3. However, we have added a second **layer** for each of these three group types. Groups 6, 7, and 8 form a second **Normal Chain**, groups 21, 22, and 23 form a second **Inside Chain** and groups 29, 30, and 31 form a second **Release Chain**. Now let’s return to our 5-note, legato phrase example (again using a simple round robin variation sequencer), and we’ll see how **Articulation 4-6** will perform. As we play our C, D, E, F, G legato phrase, the C will cause groups 1 and 6 to sound together. The D will cause groups 14 and 22 to sound, the E will cause groups 15 and 23 to sound. The F will cause groups 13 and 21 to sound and the G will cause groups 14 and 22 to sound. Finally, when the G key is lifted, release groups 27 and 30 will be triggered.

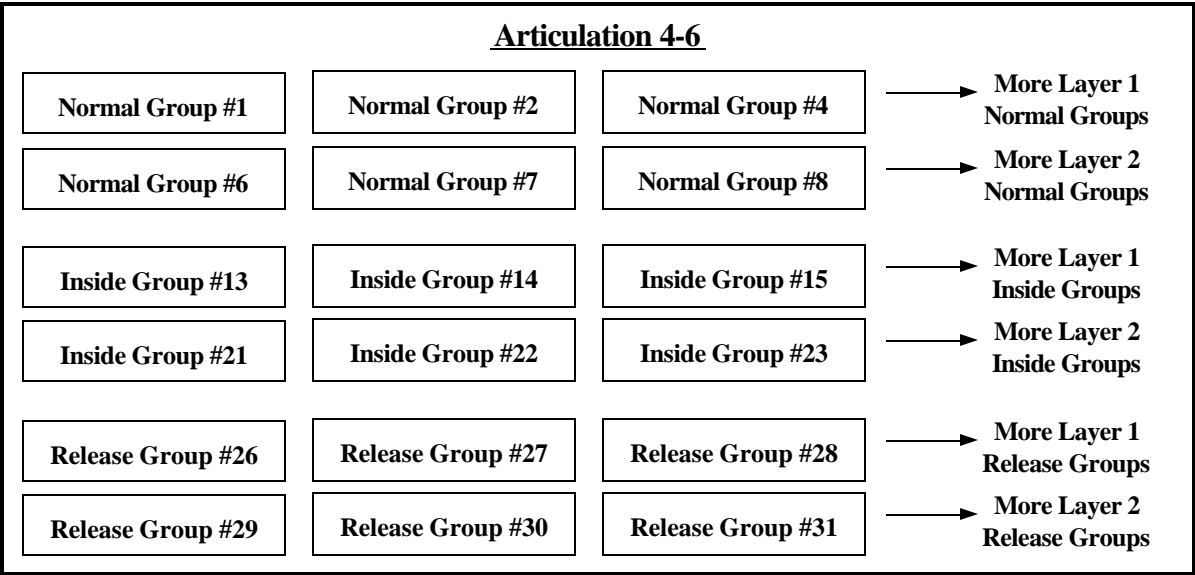


Figure 3-4

Thus, layered chains behave just as their name suggests. For each position of the variation sequencer, multiple groups of the same type (at that position) will each sound at the same time. Note that the configuration of **Figure 3-4** can be expanded both vertically and horizontally. That is the number of layers can be extended by adding more chains of the same type, and, the chain lengths can also be extended (up to a maximum of 16 groups per chain). As with the single layer situation, the chain lengths need not be the same length. When multiple **Normal Chains** are included in an articulation, the **variation cycle length** is governed by the **longest Normal Chain** layer. The excess groups of the longer, non-normal chains are never sounded and shorter non-normal layers simply don't sound. When there are no **Inside** groups to sound, the inside notes will be played from the **Normal** group and when there are no **Release** groups to sound, no release sample will be triggered.

3.6 Building Articulations

In **sections 3.2 to 3.5**, we have examined some of the ways that groups can be arranged within articulations and we discussed how these configurations perform. In the following sections, we will discuss the tools provided with the **SAS** for creating your own articulations. At the most elementary level, **SIPS Articulations** are constructed by 'programming' the group-start parameters for each group in your instrument. However, you do not need to actually set up or edit these parameters yourself, in fact, **it is strongly recommended that you do not directly manipulate the group-start parameters**. Rather, always use the **SAS** commands provided for this purpose.

All the tools needed for building articulations can be found on the **Setup/Audition** control panel, see **Figure 3-6**. The **SAS** has three panel views, the **Play Mode Panel**, the **Setup/Audition Panel**, and the **User Preferences Panel**. These panels can be selected via the **Panel Mode Menu** button (16). On the **Setup/Audition** panel, the commands needed to prepare a new instrument and to configure your articulations can be found in the **Assign Groups** menu (12). Each of these commands will be discussed in detail as we come to them.

As its name implies, the **Setup/Audition** panel can also be used to audition how things will sound. You will be able to listen to any group in isolation with or without TKT variations added. With the **Assign** commands you will be able to construct an articulation with any kind of group configuration you might want and, once you have constructed one or more articulations, you will be able to 'preview' how they will sound when 'played'. The **Setup/Audition** panel also provides the facility to set the **Instrument Range** and position the dual-group keyswitches in the most convenient keyboard location(s).

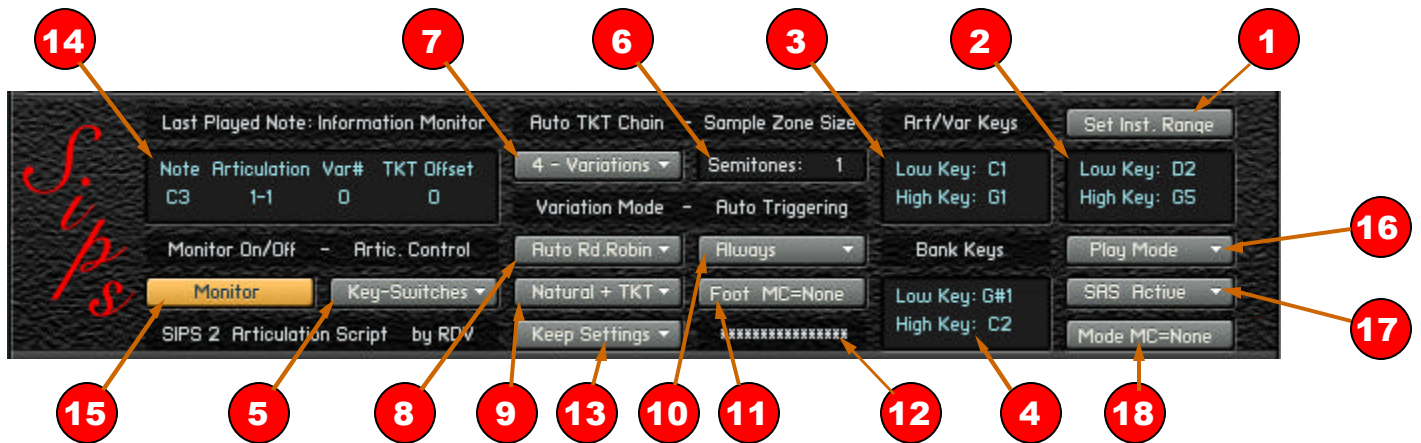
3.7 Beware of K2/K3 Quirks

SIPS 2 uses some of the engine parameters (introduced with V2.2) for manipulating the group-start settings. Apparently altering these parameters conflicts with certain kinds of activity and NI has not done a very thorough job of keeping these conflicting processes mutually exclusive. For example, if one or more notes are sounding when the KSP writes to these parameters, K2/K3 can crash violently! In other areas, evidently some protective code has been included but it's 'half-baked' and often causes strange problems of its own, including things like exiting callbacks at any 'wait' statement, 'losing' the values of polyphonic variables, etc. **SIPS 2** includes some extra code to protect you from the worst of these situations but not everything can be 'fixed' with script code. Therefore, the following practices are recommended whenever you are executing any of the **Assignment** commands.

Before executing any of the **Assignment** commands, stop all notes from sounding and then refrain from playing any more notes while these commands are executing. Notes played during command execution are collected in some 'secret' place by NI and will then come blasting out when the command finishes. These note events have no IDs and cannot be stopped by the KSP. If you fall into this pothole, you might be able to kill these notes by clicking the instrument's MUTE button and then releasing it. But, better yet, don't play notes while you are assigning groups. Please also refrain from clicking on and/or changing any of the group-start parameters during command execution (you shouldn't find it necessary to fiddle with these anyway since the **Assign** commands will handle all this for you). Fortunately, none of these kinds of problems occur during normal use, only during the configuration process.

3.8 The SAS Control Panels

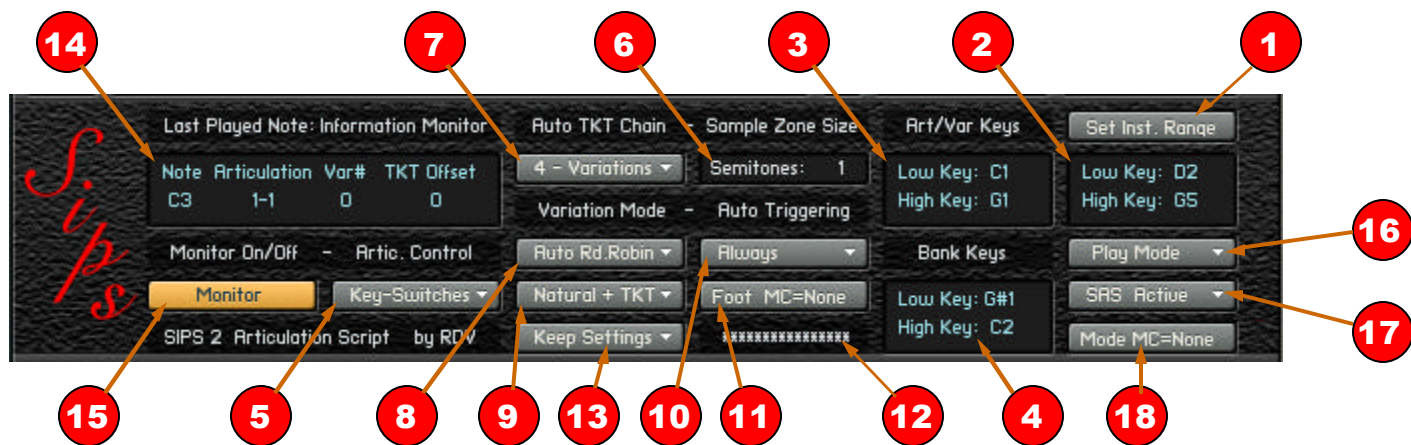
The main **Play Mode** panel and legend is shown in **Figure 3-5** and the **Setup/Audition** panel and legend is shown in **Figure 3-6**. These panels, as well as the **Preferences Panel** (Figure 2-1) are accessed from the **Panel Mode Menu**, designated as (16) in Figure 3-5 or Figure 3-6. The main **Play Mode** panel is intended to be used during normal ‘playing’ while the **Setup/Audition** panel is intended to be used only during the initial instrument configuration process. However, you will notice that the main panel and the setup panel have several controls in common. Generally these ‘common’ controls are useful both during normal ‘playing’ and during instrument setup.



Articulation Script Play Mode Panel

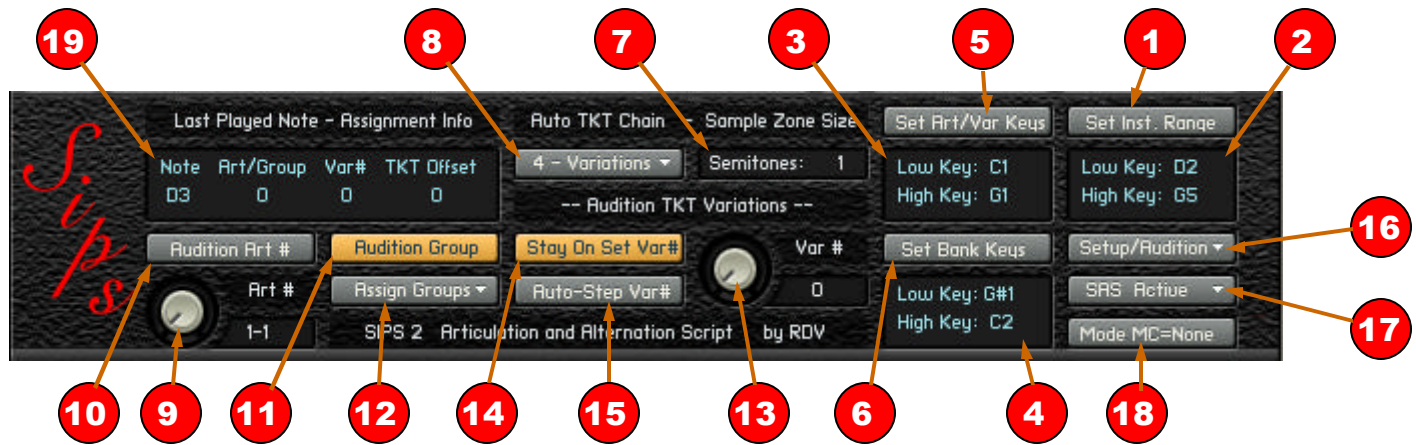
Figure 3-5

- (1) **Set Inst. Range** button. Accepts the next two notes played on a MIDI keyboard (or K2 keyboard) as the lowest and highest keys for the instrument. These keys are then displayed in (2). **Note:** Setting the Instrument Range for the **SAS** will also set the Instrument Range for all other **SIPS** member scripts.
- (2) **Instrument Range Box.** Displays the Low/High Key of the instrument range currently set for **SIPS**.
- (3) **Articulation/Variation Keyswitch Group Display** A set of eight consecutive keys used to select the articulation/variation index. This keyswitch group can be positioned anywhere on your keyboard (outside of the Instrument Range) from the **Setup/Audition Panel**.
- (4) **Bank Keyswitch Group Display** From 1 to 10 consecutive keys. The lowest key is used to activate the Variation Bank and the remaining keys (if any) are used to select the articulation bank. This key group can be positioned (separately from (3) if desired) and sized from the **Setup/Audition Panel**.
- (5) **Articulation Control Menu** You can choose to select articulations with either **Key-Switches**, **Program Change** messages, or **Both**.
- (6) **Sample Zone Size** Set this edit box to the average number of semitones covered by each sample. This value is used as a multiplier for the TKT offset calculation. See **section 3.25** for details.
- (7) **Auto-Cycle-Length Menu** When (8) is in one of the **Automatic** modes, this menu is used to set the maximum number of **TKT Variations** (besides the normal sample) that will be available.
- (8) **Variation Mode Menu** This menu allows you to choose how **Variations** are selected or sequenced. The choices are **Auto FC Rand** (ie Full-Cycle Random), **Auto Rd Robin**, and **KeySw-Select**.



Articulation Script Play Mode Panel

- (9) **Variation Mix Menu** Determines how Natural and TKT variations are combined. The choices are **All Natural**, **Natural + TKT**, and **TKT + Natural**. For **Fixed Articulations** this menu is ignored.
- (10) **Auto Trigger Mode Menu** When (8) is set to one of the **Auto** modes, this menu is used to choose the triggering method that will be used. The choices are **Never**, **Match in 2**, **Match in 3**, and **Always**. See **Section 3.35** for details of how these modes work.
- (11) **MIDI Foot Switch Assignment Menu** Click this *assignment-button* **twice** to see a drop-down menu of MIDI CCs that can be assigned to be logically 'ANDed' with the Triggering Menu (10). See **Section 3.35** for how this can be used.
- (12) **Foot Switch Status** If (11) is assigned to a **CC**, this box displays the **ON** or **OFF** status of the CC. If no **CC** is assigned, a row of asterisks is displayed.
- (13) **Keep Settings Menu.** Variation setups, that is the settings of (6), (7), (8), (9), and (10), can be linked to an Articulation using this menu. Opening this menu and selecting **Attach to Articulation**, will link the current Variation setup with the current Articulation. Thereafter, whenever this articulation is chosen, the same variation setup will be recalled. You can also disassociate the current articulation from any variation setup by opening this menu and selecting **Sever from Articulation** (see section 3.39).
- (14) **Play Monitor Window** When turned on via button (15), this window displays pertinent information about the last note played.
- (15) **Play Monitor On/Off** Enables the **Play Monitor Window** display (**On**) or blanks it out (**Off**). This turn-off option is provided so you can reduce CPU demand if you are pushing the limits.
- (16) **Panel Mode Menu** Use this menu to switch between the **Play Mode** panel, the **Setup/Audition** panel, or the User **Preferences** panel.
- (17) **SAS Mode Select Menu** Enables (**SAS Active**) or Disables (**SAS Off**) the **SAS**. Can also be MIDI controlled with (18).
- (18) **SAS Mode Assignment Menu** Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control (17). A **CC** value of **0** to **63** will select **SAS Off** while a value of **64** to **127** will select **SAS Active**.



Articulation Script Setup/Audition Panel

Figure 3-6

- (1) **Set Inst Range Button** Click this button and follow the prompts to set or reset Instrument Range.
- (2) **Instrument Range Display** This is the same as item (2) in **Figure 3-5**.
- (3) **Articulation/Variation Keyswitch Group** This is the same as item (3) in **Figure 3-5**.
- (4) **Bank Keyswitch Group** This is the same as item (4) in **Figure 3-5**.
- (5) **Set Art/Var Keyswitch Position** Click this button followed by the **Low Key** desired for this group.
- (6) **Set Bank Keyswitch Range** Click this button followed by the **Low** and **High Keys** desired.
- (7) **Sample Zone Size** This is the same as item (6) in **Figure 3-5**.
- (8) **Auto-Cycle-Length Menu** This is the same as item (7) in **Figure 3-5**.
- (9) **Articulation Index Selector Knob** This knob specifies an articulation for auditioning or assignment.
- (10) **Audition Mode Buttons** These are mutually-exclusive (ie 'radio') buttons. If you click on (10), it will activate and (11) will de-activate and vice versa. With (10) active, any notes that you play will sound the articulation set by (9). With (11) active, any notes you play will sound the group currently **highlighted** in K2's Group Editor.
- (11) **Audition Mode Buttons** These are mutually-exclusive (ie 'radio') buttons. If you click on (11), it will activate and (10) will de-activate and vice versa.
- (12) **Assign Groups Menu** This drop-down menu contains commands to **Assign** Instrument Groups to **Articulations** plus a number of Utility commands. See **Section 3.17** for details.
- (13) **TKT Variation Selector Knob** With (14) active, any note you play will be played with the variation index set by (13). With (15) active, when you play the same note repeatedly, the variation index will automatically sequence from 0 to 8. Each time you play a different note, the variation index will fall back to zero (ie the normal sample with no TKT offset).
- (14) **TKT Audition Mode Buttons** These are mutually-exclusive (ie 'radio') buttons. If you click on (14), it will activate and (15) will de-activate and vice versa.
- (15) **TKT Audition Mode Buttons** These are mutually-exclusive (ie 'radio') buttons. If you click on (15), it will activate and (14) will de-activate and vice versa.
- (16) **Panel Mode Menu** Allows selection of the **Play Mode**, **Setup/Audition**, or **Preferences** panels.
- (17) **SAS Mode Select Menu** This is the same as item (17) in **Figure 3-5**.
- (18) **SAS Mode Assignment Menu** This is the same as item (18) in **Figure 3-5**.
- (19) **Play Monitor Window** This window displays pertinent data associated with the last note auditioned. This window is similar to its counterpart (14) in **Figure 3-5** except that the 2nd column can display either a Group index or an Articulation index. To avoid ambiguity, articulations are always displayed with a hyphen between the Bank and Index. This window also displays **Assignment** Information.

3.9 Formatting a New Instrument

Before you can use an instrument with **SIPS 2**, all groups must have their group-start parameters properly setup. Until this is done, many things will not function properly, including the group auditioning features of the **SAS**. This group-start initialization process is called formatting and it is very easily performed. To format a new instrument for use with **SIPS 2**, open Kontakt's Group Editor and enable the **Edit All Groups** button. Then, in the **SAS**, bring up the **Setup/Audition** panel and open the **Assign Groups** menu. In the **Utility** section (near the bottom of the menu) click on **Format & Verify**. If your instrument has a lot of groups, this process could take a little while but you will see the progress displayed in the **Assignment Info** window (19).

CAUTION: If your instrument has existing group-start programming, **it will be deleted when you format**. It is important that **SIPS 2** have **exclusive** use of these parameters, which means that you must not try to use the group-start parameters for any other purpose. If you want to preserve your instrument's group-start settings for **non-SIPS** usage, please make a backup copy of your instrument before you format it. If you are interested in knowing how **SIPS** sets up and utilizes the group-start parameters, you'll find information about this in **section 3.41**.

3.10 Kontakt's Group Editor

When using the **Setup/Audition** panel to configure an instrument you will want to have **Kontakt's Group Editor** open because you will need to have access to Kontakt's Group display window at the same time. Many operations such as auditioning a group or assigning groups to an articulation will require that you select one or more groups in **Kontakt's Group Editor**, hereafter referred to as the **KGE**.

For example, before executing the **Format & Verify** command, you need to specify which groups you want to format. You specify multiple groups to the **SAS** by checking their boxes in the **KGE**. Of course when you want to format all the groups for a given instrument, you will need to check **all** the group boxes and this is most easily done by using the **Edit All Groups** button (as was described in **section 3.9**). However, as you are building up an instrument, you may occasionally need to add some new groups. Whenever you do that, you will need to format the new groups before you can do anything else with them. But, formatting not only initializes the groups, it also de-assigns them from all articulations. Therefore, when you format some newly added groups, generally you will not want to re-format your existing groups (unless of course you want to clear all articulations and start over).

Therefore, before formatting new groups, be very careful (when checking the new group boxes), that you don't accidentally check some of the previously formatted groups. If you accidentally include a previously formatted group that hasn't yet been assigned to an articulation, re-formatting will do no harm. However, if you accidentally re-format a group that was already assigned to an articulation, you will delete that group and most likely will experience future problems with that articulation. Similarly, when executing any of the assignment commands that require you to select one or more groups, you need to be very careful not to check too many (or the wrong) groups.

Since it is always important to be precise with group checkbox selections, there are some practices you can adopt that will greatly reduce the possibility of error. One thing you can do is to keep your eye on Kontakt's **Group(s) Edited** display in the lower right hand corner. This display shows the number of groups checked (as the numerator) and the total number of groups in your instrument (as the denominator). A good safety precaution to use before checking a number of groups (especially if you can't see all the groups at once without scrolling) is to **highlight** the first group you want to check and then enable and disable the **Edit All Groups** button. This will **set** and then **clear** all checkboxes **except for the group that is highlighted**. Once you have established this condition, you can then check the remaining boxes. But, before executing the assignment command, take one last look at the **Group(s) Edited** display and make sure the numerator agrees with the total boxes you think you have checked.

3.11 Using the Setup/Audition Panel

The following sections will focus on how to use the **Setup/Audition Panel** to construct the various kinds of articulations that were described in **sections 3.2 to 3.5**. We will also discuss what each control does and how you can use it. Then, beginning with **Section 3.26**, we will discuss how the features of the main **Play Mode Panel** are used during ‘performance’.

3.12 Auditioning Groups

When configuring a new instrument for use with the **SAS**, the process will usually consist of listening to the various groups available and then assigning the groups of interest to a convenient series of **Articulations**. The first step in this process (if you haven’t already done so) is to format all of the instrument groups using the procedure given in **Section 3.9**. This will ‘clear out’ any leftover group-start programming and will also enable all the **SAS** tools to function properly. Once the instrument has been formatted, bring up the **Setup/Audition Panel** and be sure that the **Audition Group** button (11) is active. Then open the **KGE** so you can see all the instrument groups.

You can now hear how any group sounds by first **highlighting** it in the **KGE** and then playing some notes on your keyboard. You can choose any group, including those that contain release samples. Only the **highlighted** group will sound when you hit a key (even release groups will sound on key down). If you want to hear the normal samples (without any TKT effect), be sure that **Stay On Set Var#** (14) is engaged and that the **Var #** knob (13) is set to zero. Alternatively, if you want to preview how a group will sound when various TKT offsets are used, you can select any variation number with the **Var #** knob (13). As you audition the groups, you can decide which ones you want to assign to various articulations but any group can be auditioned by itself, regardless of whether or not it is already assigned to an articulation. If you want to hear a group without **any SIPS effects** added, you may want to also disable both the **SLS** and **SVS** while you are auditioning groups.

3.13 Auditioning Articulations

In the **Setup/Audition** panel, you can also audition articulations at any time. First enable the **Audition Art #** button (10) and then set the **Art #** knob (9) to the desired articulation. Now, when you play your keyboard, the selected articulation will play and sound the same way it will when using the **Play Mode** panel. It is important to understand the distinction between auditioning a group and auditioning an articulation. A simple way to illustrate the difference is as follows. Suppose you have created an articulation that has group 7 assigned as a normal group and group 8 assigned as a release group. If you audition that articulation, when you press a key on your keyboard, group 7 will sound and when you release the key, group 8 will be triggered (as a release sample). This is of course the same thing that will happen in the normal **Play Mode** when this articulation is active. However, if you use the **Audition Group** feature instead and **highlight** group 7 in the **KGE**, when you hit a key on your keyboard, group 7 will still sound but when you release the key it will stop sounding **without triggering any release group**. If you **highlight** group 8 in the **KGE**, when you hit a key on your keyboard, release group 8 will be played on key down (but not key up). Thus, in **Audition Group**, only the highlighted group sounds and only on key down.

3.14 The Play Monitor/Info Window

When you audition either a group or an articulation, pertinent information about the last note played is displayed in the **Play Monitor Window** (19). This window displays the note played and the active group or articulation. When auditioning a group, the group number is displayed and when auditioning an articulation, the articulation **Bank-Index** is displayed. To distinguish between these, articulations are always shown with a **hyphen** between the bank and index whereas groups are displayed with just the digits. In addition to the note played and the active group or articulation, the current TKT variation index and offset are displayed. These will be discussed later in **section 3.25**. This window is also used to display progress information and error messages when using the group assignment commands (12).

3.15 About Release Groups

It is important to realize that the only thing that tells **SIPS** that a group is a release group is how you assign it to an articulation. **SIPS 2** disables Kontakt's normal processing of release groups and instead determines when and how to trigger release groups based on its own logic and on the group type designation made when assigned to an articulation. Thus both **SIPS** and **Kontakt** ignore the setting of any group's **Release Trigger** button. Since enabling or disabling these buttons has no effect on how the associated groups are utilized, you can use these buttons as additional annotation (over and above the group name) if you like. For example, if you enable the **Release Trigger** buttons for each group that contains release samples (and disable the buttons for all other groups), the lit button can serve as a reminder that the associated group is a release group. Of course you can also include some text in the group name for the same purpose. The important point is that the setting of the **Release Trigger** button doesn't matter, either to **SIPS** or **Kontakt**.

3.16 About 'Inside' Groups

The only thing that tells **SIPS** that a group is an '**Inside**' group is how you assign it to an articulation. While it is intended that such groups will contain edited samples that have their attack portion removed (to support the new **DFD Mode**), **SIPS** has no way of knowing whether that is actually the case. If inside groups are included in an articulation, they will be used for the 'inside' notes of both legato and portamento phrases (if the **SLS** offset menu is set to **DFD Mode**). If the active articulation contains no inside groups, then the **SLS** will play inside notes from the normal groups, regardless of whether the **DFD Mode** is active or not. This behavior will occur regardless of what kind of samples are actually contained in any groups you declare to be '**Inside**' Groups.

3.17 Assigning Groups

The **Assign Groups** menu is shown in **Figure 3-7** and you can see that it is divided into four sections. The first 3 sections contain the actual group assignment commands while the 4th group contains a number of **Utility** commands. Each of the first 2 sections contain an identical set of commands for assigning the three group types. The first set is for creating **Fixed Articulations** while the second set is for creating **Chained Articulations**. The **Multi-Articulation** section allows for creation of a series of fixed articulations.

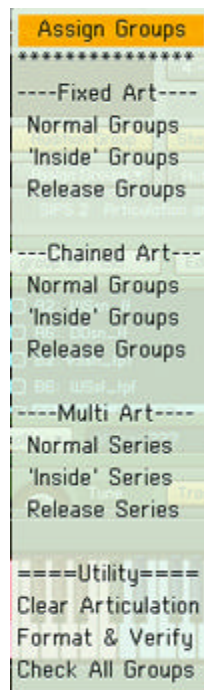


Figure 3-7

In the next few sections, we will illustrate how these commands work by building a few articulations as examples. In all of these examples we will need to alternate between using the **Kontakt Group Editor** and the **Assign Groups Menu** which, for brevity, we will refer to as the **KGE** and the **AGM** respectively. When we refer to the assignment commands in the **AGM**, we will prefix them with the articulation class (so you will know which command set is used). For example, to assign a set of **Normal Groups** to a **Fixed Articulation**, we will refer to the **AGM** command as **Fixed: Normal Groups**. When we talk about checking groups in the **KGE**, it should be understood that you need to insure that **only the specified groups are checked** (by using some safeguards, such as those suggested in **section 3.10**).

3.18 Making Fixed Articulations

To illustrate how to use the first set of assignment commands, let's walk through the steps we would use to build a **Fixed Articulation** like that depicted in **Figure 3-3**. First, set the **Art #** knob (9) to articulation **6-3**. Next, in the **KGE**, check groups 19, 24, and 25. Then, in the **AGM**, click on **Fixed: Normal Groups**. Next, in the **KGE**, check groups 35, 37, and 42 and then in the **AGM**, click on **Fixed: 'Inside Groups'**. Finally, in the **KGE**, check groups 16, 18, and 20, and then in the **AGM**, click on **Fixed: Release Groups**.

When building a new articulation, you must always assign one or more **Normal** groups first. If building a **Fixed** articulation, you must use the assignment commands under the **Fixed Art** heading (or the **Multi Art** heading, but this will be discussed in **section 3.20**). Conversely, when building a **Chained Articulation**, you must use the commands under the **Chained Art** heading. If you are building a **Fixed** articulation, you can add additional groups at any time. To illustrate that, let's rebuild **Articulation 6-3** another way.

Let's use this opportunity to introduce another **Utility** command (the **Format & Verify** command was already discussed in **section 3.9**). Since we just built an articulation 6-3, we can use the **Clear Articulation** command to 'de-assign' all the groups that we assigned to it. You can do this by setting the **Art #** knob to **6-3** and then clicking on **Clear Articulation** in the **Utility** section of the **AGM**. If we don't take this first step we would have two problems with trying to re-create articulation 6-3. Since that articulation is already in use, we would have to build the new articulation with a different **Bank-Index** number. Secondly, all the groups we would need to assign are already in use by articulation 6-3 and remember, each group can only be assigned once to a single articulation. However, once we **Clear Articulation 6-3**, all the assigned groups will again be available.

Now, let's begin our re-creation of articulation 6-3. First be sure that the **Art #** knob is still set to 6-3 and then, in the **KGE**, check only group 19. Follow this by clicking on **Fixed: Normal Groups** in the **AGM**. Next, use the **KGE** to check groups 35 and 37 followed by clicking on **Fixed: 'Inside' Groups** in the **AGM**. Next, use the **KGE** again to check groups 16, 18, and 20 followed by clicking on **Fixed: Release Groups** in the **AGM**.

Now, at this point, articulation 6-3 is missing **Normal** groups 24, and 25 and **Inside** group 42. So, in order to complete the **Fixed Articulation** of **Figure 3-3**, proceed as follows. First be sure that the **Art #** knob is still set for **6-3** and then use the **KGE** to check groups 24 and 25. Follow this by clicking on **Fixed: Normal Groups** in the **AGM**. Finally, check only group 42 in the **KGE** and then click on **Fixed: 'Inside' Groups** in the **AGM**. Obviously, the first method requires fewer steps. However, if you started out only planning to use the lesser number of groups and then later decided you needed to add some groups, this ability could prove useful.

3.19 Making Chained Articulations

For our next example, let's again create the articulation of **Figure 3-3**, only this time as a **Chained Articulation**. It is not possible to change the class of an existing articulation (fixed to chained or vice versa). Therefore, the only way to convert articulation 6-3 from **Fixed** to **Chained** is to erase it and rebuild it from the ground up. Therefore, our first step will again be to set the **Art #** knob to **6-3** and then execute the **AGM** command for **Clear Articulation**.

Next, with the **Art #** knob still set to **6-3**, check groups 19, 24, and 25 in the **KGE** and then click on **Chained: Normal Groups** in the **AGM**. Then, check groups 35, 37, and 42 in the **KGE** and then click on **Chained: 'Inside' Groups** in the **AGM**. Finally, check groups 16, 18, and 20 in the **KGE** and then click on **Chained: Release Groups** in the **AGM**. Note that this procedure is identical with the first procedure we used to create the **Fixed** articulation except that we use the assignment commands in the **Chained Art** section of the **AGM** (instead of the assignment commands in the **Fixed Art** section).

As a final example of creating a **Chained Articulation**, let's make up the chained articulation depicted in **Figure 3-4**. We'll begin by building the first layers for each group type. Set the **Art #** knob to **4-6** and then check groups 1, 2, 4 in the **KGE** and then click on **Chained: Normal Groups** in the **AGM**. Next, check groups 13, 14, and 15 in the **KGE** and then click on **Chained: 'Inside' Groups** in the **AGM**. Finally, check groups 26, 27, and 28 in the **KGE** followed by clicking on **Chained: Release Groups** in the **AGM**. This will produce the first layer of each group type depicted in **Figure 3-4**.

Just like **Fixed Articulations**, we can add more layers to a **Chained Articulation**. What we **can't** do however is to **extend the length** of an existing chain. We can create another longer chain of the same type but it will be layered with the first chain (which will then be shorter). If these two chains are **Normal**, then the new chain length for the variation sequencer would become 4. But, if you create a chained articulation with **Normal** groups 6, 7, and 8 and then you later decide you really want it to be a chain of the **Normal** groups 6, 7, 8, and 9, you will have to erase the articulation and start over.

OK, now let's finish up **Articulation 4-6** by adding the 2nd set of layers. First be sure that the **Art #** knob is still set to 4-6. Then, check groups 6, 7, and 8 in the **KGE** followed by clicking **Chained: Normal Groups** in the **AGM**. Next, check groups 21, 22, and 23 in the **KGE** followed by clicking **Chained: 'Inside' Groups** in the **AGM**. Finally, check groups 29, 30, and 31 in the **KGE** followed by clicking **Chained: Release Groups** in the **AGM**.

3.20 Using the Multi Art Commands

While you will occasionally build articulations with all three group types, most of your articulations will probably consist of only **Normal Groups**. For example, let's say you wanted to assign groups **40 through 49**, each to their own articulation (ie one group per articulation). Assume also that you have a contiguous series of 10 articulations available (covering the range from **4-3** to **5-4**). One way to make these assignments would be to set the **Art #** knob to **4-3** and then, in the **KGE**, check groups **40 through 49**. Now simply click **Normal Series** in the **Multi Art** section of the **AGM**. The way this command works is to assign the lowest numbered group checked to the articulation number set by the **Art #** knob. Then, the next higher group checked will be assigned to the next articulation and so on until all checked groups have been assigned to the contiguous series of articulations. Note that while the articulation series must be contiguous, the groups assigned do not need to be contiguous but the groups will be assigned in ascending numerical order.

The **Multi Art** commands can also be quite useful if you need to build a series of articulations containing more than just normal groups but nevertheless, all of the same structure. For example, suppose you want to build 5 articulations that each have one normal and one release group. Let's say that the normal groups are to be **60, 63, 64, 70, and 75** and that the corresponding release groups are **61, 68, 72, 73, and 74**. If you have a block of unused articulations from **3-1 to 3-5** you can build your five articulations as follows. First set the **Art #** knob to **3-1** and then check groups **60, 63, 64, 70, and 75** in the **KGE**. Next, click **Multi: Normal Series** in the **AGM**. Then, check groups **61, 68, 72, 73, and 74** in the **KGE** and then click **Multi: Release Series** in the **AGM**. This procedure will assign **normal group 60** and **release group 61** to **articulation 3-1**, it will assign normal group **63** and release group **68** to articulation **3-2**, normal group **64** and release group **72** to articulation **3-3**, etc.

Let's examine one more scenario. Suppose you want to build an articulation that contains 3 normal groups (each group containing velocity samples at one of three levels). Further, let's say you have 5 sets of such samples and you want each set to be assigned to its own articulation. Suppose the velocity trios are in groups **1,2,3** and **4,5,6** and **7,8,9**, and **10,11,12** and **13,14, 15**. And, you want to assign the trios to articulations **4-1, 4-2, 4-3, 4-4, and 4-5** respectively. Now, let's compare two ways of building these articulations. Using the **Fixed Art** commands we first set **Art #** to **4-1**, then check groups **1,2,3** in the **KGE**, and then click **Fixed: Normal Groups** in the **AGM**. Next we would set **Art #** to **4-2**, check groups **4,5,6** in the **KGE** and click **Fixed: Normal Groups** command in the **AGM** again. Then we would advance **Art #** to **4-3**, etc.

Alternatively, using the **Multi Art** commands we first set **Art #** to **4-1** and then check groups **1,4,7,10,13** in the **KGE** followed by clicking **Multi: Normal Series** in the **AGM**. Next we would check groups **2,5,8,11,14** in the **KGE** before clicking **Multi: Normal Series** in the **AGM**. Finally, we would check groups **3,6,9,12,15** in the **KGE** before clicking on **Multi: Normal Series** in the **AGM**.

Comparing these two ways of building this set of articulations we note that in either case we end up having to check all 15 groups, in the first case we check them 3 at a time for 5 times while in the second case we check them 5 at a time for 3 times. With the first method we click the **Fixed: Normal Groups** command 5 times while in the second method we click on the **Multi: Normal Series** command only 3 times. Finally, with the first method we have to keep advancing the **Art #** while with the second method we only set it once at the beginning. Therefore, the second method does save us some clicks and drags.

In general, the **Multi Art** commands will probably show an advantage when you need to do a lot of the same thing while the **Fixed Art** commands will be better suited to making up articulations that are all a little different and/or are built over a longer period of time (one here and one there). Also, if you need to build a number of identical articulations but the articulations are not in a contiguous series, then the **Multi Art** commands will not be effective. In summary, there are usually several ways available for you to construct articulations and you should use the one that seems easiest for your situation.

3.21 Renaming Groups

As you are assigning groups to articulations, it may be very helpful if you rename each group before (or just after) you assign it by adding a special prefix to the existing name. The recommended prefix notation for the **Fixed Articulation** example would be as follows. For the normal groups 19, 24, and 25, use something like **6-3:FN_**. This indicates that these groups are assigned to the **Fixed** articulation **6-3** and the **N** indicates the group is assigned as a **Normal** group. For the inner groups, use the prefix, **6-3:FI_** and for the release groups use the prefix **6-3:FR_**. On the other hand, if **6-3** is a **Chained Articulation**, you would replace the **F** in the prefixes with **Cn**. The **C** tells us it's a **Chained** articulation and the **n** (a digit from 0 to 15) will tell you which position in the chain the group occupies. Note that the first group position in a chain is referred to as position zero. For example, consider the **Chained Articulation 6-3**. After assigning groups 19, 24, and 25, you would rename these groups using the prefixes **6-3:C0N_**, **6-3:C1N_**, and **6-3:C2N**. Suppose, for example, that group 25 was named **Loud Staccato**. You would then rename this group as **6-3:C2N_Loud Staccato**. Using a set of consistent prefixes like this will make it much easier to sort things out later (after your memory fades). This will be especially true if your instrument has a large number of groups.

I probably shouldn't tell you this but, if you don't rename your groups, there is another (but less obvious way) to determine how the groups were assigned. If you understand how **SIPS** programs the group-start parameters (see **section 3.41**), you can probably determine how any given group was assigned by studying its group-start programming. However, in the long run, renaming the groups is a far more convenient way to document what you have done.

3.22 Check All Groups and Assignment Errors

When executing the various assignment commands, **SIPS** scans all the instrument groups to ‘verify’ their group-start programming veracity. For example, this is done after executing the **Format & Verify** command. This verification process essentially checks that each group has the correct set of CCs assigned and that their values are set within the ‘allowable’ ranges. If any structural defect is discovered, an error will be reported. As part of this verification process, **SIPS** also ‘rebuilds’ its **Articulation Table** which is used to control how groups are combined into articulations.

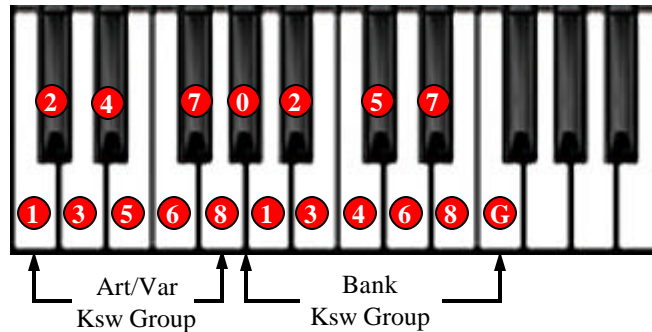
When you first install the **SIPS** scripts into an instrument, and each time the instrument is reloaded, an automatic verification (and **Articulation Table** rebuild) is performed. If the group-start programming structure is not correct and/or a proper **Articulation Table** cannot be built, a blinking error message will appear on Kontakt’s status line. For the most part, the verification process goes on automatically and, as long as you don’t do something to corrupt the group-start programming, you may not even be aware of the process at all. However, if for any reason you suspect that the group-start programming may have become corrupted, you can force a verification at any time by executing the **Check All Groups** command in the **Utility** section of the **AGM**. However, it should be emphasized that the verification process is somewhat like a syntactical error detector but it knows nothing about any symantical errors you may have made. If you wrongfully tell **SIPS** you want groups 5, 7 and 8 to be **Normal** groups in some articulation, then that’s the way **SIPS** will set it up. But, if you actually meant that it should be groups 5, 6, and 7, the verification process has no way of detecting this kind of error.

As you execute the assignment commands, the **Info Window (19)** will indicate what is happening and whether or not your command was carried out successfully. However, there are many kinds of problems that can prevent these commands from being completed successfully. For example, things like trying to re-assign one or more groups that are already assigned to an articulation, trying to change the class of an articulation, etc. When **SIPS** detects this kind of error, it will display a descriptive error message in the **Info Window** and also blink it on the status line for a period of time. When such an error message appears, it will also mean that your command could not be executed and you must make the necessary corrections first and then repeat the command.

3.23 Keyswitch Setup

The **SAS** uses a 2-group keyswitch configuration that is both very flexible and easy to reposition. The two keygroups consist of the **Art/Var** index group (always containing 8 consecutive keys), and a **Bank** group containing from 1 to 10 consecutive keys (depending on how many articulations you want to control). For the maximum configuration where the keyswitches can be used to select any of 64 Articulations, any of 8 Variations, and a silent or ‘ghost’ bank (discussed in **section 3.31**), a total of 18 keys are needed. The index and bank groups can be positioned adjacent to each other or separated (and can be anywhere on your keyboard) as long as both are ‘outside of’ the **Instrument Range** setting.

Positioning the two keygroups is very easy to do from the **Setup/Articulation** panel. To set the Art/Var index group, click on **button (5)** and follow this by hitting the root key for the group. **SIPS** will then setup this keygroup with 8 consecutive keys, starting from the designated root key. If there is currently no valid **Bank** keygroup defined, **SIPS** will temporarily assign a **Bank** group (with one key) to the next key after the **Art/Var** key group. This is done because both keygroups (or neither) must be defined. If such an auto assignment of the **Bank** keygroup is made and you don’t want it there, it is easy to move it. To set (or move) the **Bank** keygroup, simply click on **button (6)** and follow the prompts by hitting the lowest and highest keys you want the **Bank** keygroup to cover. If you enter the **Bank** keygroup first and there is as yet no **Art/Var** index keygroup defined, **SIPS** will create the group for you just under the lowest Bank key you specified. If this is not a suitable location, you can easily move it elsewhere. When you enter either or both of these keygroup locations, they must not overlap each other nor the defined **Instrument Range**. If they do, both keygroups will be automatically de-assigned.



Maximum Keyswitch Configuration
(With Adjacent Key Groups)

Figure 3-8

Figure 3-8 illustrates the maximum configuration with the 2 key groups adjacent to each other. In ‘performance’ mode, you can use the keyswitches to select articulations, variations, both, or neither depending on how you set things up. The first **Bank** key (0) always activates the **Variation Bank**, the next **8 Bank Keys** activate the corresponding **8 Articulation** banks (1 to 8), and the highest **Bank Key** can be used as an optional **Silent** or ‘**Ghost Bank**’ (G) (see section 3.31). If your instrument uses fewer than **8 Articulation** banks, you can set the **Bank Group** to be smaller than shown in **Figure 3-8**. For example, if your instrument only has 16 articulations and you have them ‘packed’ into the first two articulation banks (ie you are using only articulations **1-1** to **2-8**), you could define the **Bank Group** to contain only the first 3 bank keys (ie 0, 1, and 2 in **Figure 3-8**).

If you want to use the keyswitches to select only **Variations** (ie you won’t be using them to select articulations), you can define the **Bank Group** with only a single key (the **Variation Bank** key). However, if you use the keyswitches for either **Variations** or **Articulations** (or both), you must always define a valid 8-key **Art/Var** index group. If you don’t want to use keyswitches at all, simply define an ‘illegal’ **Art/Var** index group and both keyswitch groups will be de-allocated. You can define such an ‘illegal’ key group by simply entering a root note ‘within’ the **Instrument Range**.

NOTE: By default, **SIPS Keyswitches** are shown on Kontakt’s keyboard as ‘pink’ tinted keys so that you can easily identify where they are relative to the Instrument Range (which is displayed with ‘blue’ tinted keys). If you would rather not have **SIPS** highlight its keyswitches in pink, simply enable **User Preference Sw7** which will then show the keyswitches without the pink tint, thereby effectively no longer displaying them.

3.24 SIPS Variations

Besides articulation control, the **SAS** can also provide two types of **Variation** (alternation). We’ve already touched on **Natural Variation** in which we construct **Chained Articulations** that will allow us to cycle through these sampled variations. We will discuss the various means by which we can cycle such natural variations in later sections. But, in addition to natural variations, the **SAS** can also generate up to **8 TKT Variations** for each sounding group. **TKT Variations** are synthetic variations produced by ‘borrowing’ adjacent samples and retuning them. In the next sections, we’re going to describe some of the details of the TKT system provided with **SIPS 2**. Then in later sections we will explain how you can utilize TKT and/or Natural variations in a variety of useful ways.

3.25 The TKT Variation Index

Each articulation can be played with up to 8 alternate tonalities (besides the normal sample itself). The TKT effect provides alternate tonalities by playing a different multi-sample and retuning it back to the pitch of the played note. The **SAS** uses a **Variation Index** from 1 to 8 to select a specific variation. If you have an instrument that is fully multi-sampled, ie there is a separate sample for each note (and thus **Zone Size** is set to **1 semitone**), then the relationship between the **Variation Index** and the **Sample Offset** is as shown in **Figure 3-9**.

The meaning of the table in **Figure 3-9** is as follows. When the index is 1, play the next-higher sample. When the index is 2, play the next-lower sample. When the index is 3, play the sample that is two above the current note, etc. For example, if we were playing **C3** and the index were **1**, we would sound the sample for **C#3**. If the index were **6**, we would sound the sample for **A2**, etc. Of course we always retune these samples so they still sound with the pitch of **C3**. Finally, when the TKT index is zero, we simply play the normal sample (zero sample offset).

Index	Sample Offset
1	+1
2	-1
3	+2
4	-2
5	+3
6	-3
7	+4
8	-4

Sample Offset Vs Variation Index
Figure 3-9

To see this in action, set **Zone Size** to **1 Semitone** and activate the **Auto Step Var#** button (15). Now play a series of 9 or so notes (with the same key) and watch the **Play Monitor Window** (19). You should see that the reported **Var #** and the **TKT Offset** value follow the progression indicated in **Figure 3-9**. Now increase the **Zone Size** setting to **2 semitones** and repeat the experiment. Note that the **TKT Offset** now sequences through +2, -2, +4, -4, +6, -6, +8, and -8. In other words, the **TKT Offset** is now twice the **Sample Offset** indicated in **Figure 3-9**.

The purpose of the **Zone Size** setting is to allow for more-sparsely-sampled instruments. For example, if you have an instrument which is sampled every other note, then using the **Sample Offset** values of **Figure 3-9** might play the same sample for Variation 1 and Variation 3. So, the **SAS** computes the **TKT Offset** by multiplying the desired **Sample Offset** by the **Zone Size**. It's not perfect, because some instruments use a somewhat variable sample spacing. However, allowing you to specify the 'nominal' **Zone Size** provides for 'mostly' correct behavior. Of course, the TKT always works the best when the instrument provides a sample for every semitone, and you can set **Zone Size** to 1.

Another thing we need to consider is what to do when the **TKT Offset** would take us outside of the **Instrument's Range, IR**. The strategy used by **SIPS** when a TKT Offset 'runs off the end' of the **IR** is as follows. First compute the desired **TKT Offset** and test it. If the resultant note is within the **IR**, use the computed offset. However, if the offset extends beyond the **IR**, reflect it (ie if the computed offset is **-6** and this extends below the **IR**, change the **-6** to **+6**). Finally, if after reflecting the offset, the result is still out of bounds (this is very rare) then 'clamp' the offset to the **IR** limit on the 'reflected' side. To indicate when either of these things happens, the **Information Monitor** (19) displays the actual **TKT Offset** used but if it resulted from a reflection, a **single asterisk** is displayed to the right of the offset. If the offset resulted from reflected clamping, a **double asterisk** is displayed to the right of the displayed **TKT Offset** value.

3.26 Using the Play Mode Panel

Starting with this section we will now discuss the features of the main **Play Mode Panel** and how it is used during ‘performance’. It will be assumed that all the necessary configuration setups have already been made using the **Setup/Audition Panel**. In real life however, you are free to go back and forth between the two panels as often as you like (especially if you like to develop your instruments iteratively). Unless otherwise indicated, throughout the remaining sections, control legend numbers, such as the **Panel Mode** button (16), will now refer to **Figure 3-5**.

3.27 Articulation Control Choices

With the **Artic Control** menu (5) you can choose whether you want to select articulations with **Key-Switches**, **MIDI Program Chg** messages, or **Both**. Note however that you must have **KSP+** installed in order for the **SAS** to respond to **Program Change** messages.

3.28 Selecting Articulations with MIDI PC Messages

To select (or change to) any given articulation, simply issue a **MIDI Program Change** message with the desired articulation index as the program number. For example, to switch to **Articulation 3-4**, issue a **PC** message with **program number 34**. Whenever you issue such a command, it will not change any notes already sounding but rather, it will take effect for the next note played and all those subsequent to it until another articulation change is made. The current articulation **will not change if a non-articulation code** (eg **PC = 39** or **105**) is received. You can issue a **PC** command from your keyboard as you play or you can embed such MIDI events in a sequence.

If you are issuing **PC** commands from a keyboard, take note that the **SAS** responds to the actual MIDI data byte received with each **PC** message. Most newer keyboards allow you to enter program numbers from 0 to 127 and thus have a one to one correspondence with the MIDI data byte. However, some keyboards require you to enter program numbers from **1 to 128** (but actually send a data byte of **0 to 127**). With such a keyboard, you will have to enter a program number that is one higher than the desired articulation index. If you are using such a keyboard, you can set the **User Preference Switch, Sw-3** (see also **section 2.5**) which will cause the **SAS** to add **+1** to any program number received. This will allow you to use a **1 to 128 keyboard** without mentally having to add **+1** to the desired articulation number. However, keep in mind that the **SAS** cannot distinguish where its **PC** messages are coming from. So, if you enable this preference, **all PC messages received by the SAS will be incremented**. If you are sending **PC** messages from your sequencer (as well as your keyboard), and your keyboard *differs* from your sequencer in this regard, the **User Preference** option will only correct your keyboard at the expense of your sequencer (whose codes will then all have to be reduced by one).

3.29 Selecting Articulations with Key Switches

To select (or change to) any given articulation, hit both the corresponding **Art/Var** key and **Bank** key (if need be). These keys ‘remember’ their last setting. So, for example if you are playing **articulation 3-4** and you want to change to **articulation 3-6**, it is only necessary for you to hit the **Art/Var** group **key #6**. Similarly, if you are playing with **articulation 3-4** and you want to change to **articulation 1-4**, you need only hit the **Bank** group **key #1**. However, if you are playing **articulation 3-4** and you want to change to **articulation 2-7**, you will need to hit both the **Art/Var** group **key #7** and the **Bank** group **key #2**. You can hit these keys at the same time or separately in either order, just as long as you hit both before the note to be affected is played.

SIPS 2 has a **User Preference** option **Sw7**, to display the current state of its keyswitches. With this option enabled, the currently active articulation and the current status of the keyswitches is displayed on Kontakt’s Status Line. If you activate the **Variation Bank** (see **section 3.33**), the **Art/Var** keys will be displayed as **0-1 to 0-8**. Also, if you select the ‘Ghost’ bank (see **section 3.31**), the keyswitch status (as well as the current articulation) will be displayed as **9-1 to 9-8**. By default this Articulation/Keyswitch Status display option is enabled but, if you would rather not display this info, simply turn off **Sw7** in **User Preferences**.

3.30 Triggering Release Groups

Release groups (when appropriate), are triggered by the **Legato Script**. Where and when the **SLS** triggers release samples depends on the current mode of the **SLS**. If the **SLS Mode** is set to **SLS Off**, release samples are triggered normally, ie at the end of each note (when it is released). For more information about the conditions under which the **SLS** will attempt to trigger release groups, see **section 4.12**.

However, it is important to realize that if you play a note (of a certain articulation) and then, if while that note is sustained you change articulations; when the note ends the release sample(s) for the new articulation will be triggered. So, normally you shouldn't change articulations during the last note of a phrase if you want the release sample to belong to the same articulation as the sustained last note of the phrase.

3.31 Using Silent (Ghost) Notes

When playing legato phrases, normally the first note of a phrase is played with a normal attack and only the 'inside' notes are 'effected' with crossfades and bends. However, there may be situations in which you also want the first note of a legato phrase to be played as though it were an 'inside' note. This can be accomplished by playing a 'silent', overlapping note ahead of the first note of the phrase. To accomplish this, you need to select an unassigned articulation before playing the 'ghost' note. Then, between the playing of the ghost note and the actual first note of the legato phrase, you change the articulation to the one you want to sound. When you play the actual first note of the phrase, if the 'ghost' note is still active (ie it's key is still down, thus overlapping the real first note of the phrase), the real first note will be played with the legato crossfade and bend effect; just as though the 'ghost' note had been an actual sounding note.

As far as the **SAS** is concerned, all you need to do to play a 'ghost' note is to choose an articulation that has not been assigned to any groups. So you could choose any unused articulation in any available bank. If you are using **MIDI Program Change** for articulation control, one such unassigned articulation is as good as any other. However, if you are using **Keyswitch** articulation control, you can avoid having to hit 2 keys if you have an entire bank of unassigned articulations. Then, when you want to play a 'ghost' note, all you have to do is hit the 'empty' bank key. Since the whole bank is 'empty' it won't matter which **Art/Var** key was active last (nor will you have to change it)..

But, what if all 64 articulations are assigned to groups? To handle this situation, a 9th, pseudo bank is available. While there are no valid articulation indices above 88, you can create a 9th bank for the **Keyswitches** as shown by the key labeled **(G)** in **Figure 3-8**. Therefore, you can 'pseudo-select' articulations that would be numbered 91, 92, 93, 94, 95, 96, 97, or 98. These can be selected with the **Keyswitches** or with **PC** commands and since they don't represent 'true' articulation indices, they are always unassigned. Thus, any of these pseudo articulations can be used to play a 'ghost' note.

3.32 Variation Selection Modes

You can choose from 3 different modes for 'selecting' **Variations** via the **Variation Mode** menu (8). there are two automatic modes plus a manual mode in which you can use the keyswitches to select variations. The three modes are **Auto FC Rand**, **Auto Rd Robin**, and **KeySw-Select**. Of these, the **Automatic** modes are the most complex so we'll save them for last. With the **KeySw-Select** mode active, you and you alone decide if and when a **Variation** will play and you alone specify which of the possible natural or TKT variations will play. Whereas, in the Automatic modes, whether or not to play a variation, and if so which one, is determined by algorithmic choice as specified by other settings as will be described in **Sections 3.35** and **3.36**.

If the active articulation is of the Chained class, ie if there are 'natural' variations, how things behave will also be affected by how the **Variation Type** control menu (9) is set. This menu determines if just natural or both natural and TKT variations will be utilized and how. There are three choices for this menu which are **All Natural**, **Natural + TKT**, and **TKT + Natural**. These three options will be explained in detail as we continue to explore **SIPS 2's** powerful Variation Control System.

3.33 Key-Switch Selection of Variations

When the **Key-Switches** mode or **Both KS & PC** mode is selected with (5), you can select variations by hitting the lowest **Bank 0** group key (see **Figure 3-8**), followed by any of the **Art/Var** group keys **1 to 8**. Once the **Variation Bank** key is activated, it remains active until some other **Bank** key is hit so you can keep playing **Variations** by simply hitting the desired **Art/Var** key **1 to 8**. The variation that will be selected by the keys depends on the class of the articulation and the setting of the **Variation Type** menu (9) but you can always return to the base variation by hitting **Bank 0** again.

If the active articulation class is **Fixed**, the **Variation Type** menu is ignored (because only TKT variations are available). In this case, the **Art/Var** keys from 1 to 8 will select the corresponding **TKT** variation number and hitting just the **Bank 0** key again will set the TKT index back to zero. Note also that whenever you hit any other Bank key, the TKT index is also reset to zero.

If the active articulation class is **Chained**, the three **Variation Types** affect **KeySw Selection** as follows. If (9) is set to **All Natural**, the **Art/Var** keys 1 to 8 will select the natural variations in positions 1 to 8 of the chain. Hitting **Bank 0** selects the base chain position zero. If the chain length exceeds 9 (ie 0..8), you won't be able to select the higher groups and if the chain is shorter than 9, selecting a higher number than the end of the chain will result in silence. For all of these selections, no TKT variation will be played (ie TKT index is zero).

With (9) set to **Natural + TKT**, **Bank 0** again will select the base chain position 0. **Art/Var** keys 1 to 4 will select the next 4 chain positions 1 to 4 and **Art/Var** keys 5 to 8 will select the **TKT** variations 1 to 4. For **Art/Var** keys 1 to 4, the **TKT** index will be zero while for **Art/Var** keys 5 to 8, the chain position will be zero. In other words, the first 4 keys choose the first 4 **Natural** variations (after the base) with the **TKT** at its base value. Whereas the 2nd set of 4 keys chooses the first 4 **TKT** variations with the **Natural** chain index at its base position.

With (9) set to **TKT + Natural**, the roles of the 8 **Art/Var** keys are reversed. **Keys 1 to 4** select the first 4 **TKT** variations with the **Natural** at its base variation while keys 5 to 8 select the first 4 **Natural** variations while the **TKT** remains at its base variation (**TKT = 0**). Note that the last two settings of (9) both give you access to the same variations, just in a different order. If you are wondering why we would bother to include two choices like this, the answer is for the benefit of the **Automatic** modes as we will soon see.

3.34 Using the Automatic Variation Modes

When the **Variation Selection** menu (8) is set to either of the two **Automatic** modes, there are several other controls and modes to reckon with. There are two distinct issues that must be decided by any automatic Variation system. The first issue is under what conditions a Variation should be played and the second issue is **which** Variation should be chosen. The first issue deals with what might be called **Triggering** options while the second issue deals with what might be called **Selection** options.

3.35 Auto-Triggering Options

The triggering options available can be selected from the **Auto Triggering** menu (10) and can be further human-influenced by assigning a **MIDI Footswitch** via the drop-down menu (11). Once the variation mode has been triggered, **Selection** of which variation to play depends on several other factors which we will discuss in the next section. The triggering choices are **Never**, **Match in 2**, **Match in 3**, and **Always**. When (10) is set to **Never**, no **Variations** will be played under any conditions. When (10) is set to **Match in 2**, the 'next' variation will be played whenever the second of any two identical notes in a row occur. Thus, if C3 occurs 8 times in a row, the first C3 will play the base variation but the next 7 C3s will each play a **Variation** pulled from those that are available and as determined by the selection mode that you are using (see **section 3.36**). For a long series of variations (longer than the full **Auto Cycle Length**), each new cycle will begin with the base variation again and then be followed by a new series of selected variations.

Now, if the ‘trill’ series C3, D3, C3, D3, C3, D3 occurs, no variations will be played for **Match in 2** triggering because there are no occurrences of 2 in a row of any note. However, if (10) is set to **Match in 3**, Variations will be played for the last 4 notes. In this mode whenever a note matches either of the two preceding notes, the trigger is set. Thus **Match in 3** can trigger variations on either a trill or a ‘machine-gun’ series. Finally, setting (10) to **Always** causes a variation to be played for **every note** (however remember that the base variation (zero for **TKT** or chain position 0 for **Natural** variations) is always included in each full cycle set.

For more human influence, but still somewhat automatic, you can assign a **MIDI Footswitch** from the **MC** assignment menu (11). When you do this, the **MC** is treated like a switch (similar to the sustain pedal) where a value of **0 to 63** is considered **Off** and a value of **64 to 127** is considered **On**. Once a **MC** is assigned, the status box (12) will reveal the current status of the **MC** by displaying either **Footswitch OFF** or **Footswitch ON**. When no **MC** is currently assigned, the status box (12) will show a row of asterisks.

When a Footswitch is assigned, its boolean state is logically **ANDed** with the setting of (10). If the Footswitch is **OFF**, it has the effect of preventing any TKT variation from playing no matter which setting of (10) is used. If the **Footswitch is ON**, the settings of (10) all behave as described previously (when no Footswitch was involved). So, for example, if (10) is set to **Always**, the Footswitch can be used to trigger a variation whenever it is **ON** and not otherwise. In the **Match in 2** or **Match in 3** modes, with the **Footswitch ON**, the behavior is as previously described. However, you can then countermand the trigger when you wish by changing to **Footswitch OFF**.

3.36 Auto-Variation Sequencing

When the **Variation Mode** (8) is set to either of its **Auto** modes, each time the **Auto Trigger** condition (10) is satisfied, the **Auto Sequencer** is advanced. This process continues until the **Auto-Cycle Length** is reached (or a different articulation is selected) at which point the sequencer is reset and the whole process begins over again. The sequencer can be made to cycle only through **TKT Variations**, only **Natural Variations**, or a combination of both **TKT and Natural Variations**.

For Fixed articulations, the setting of the **Variation Type** menu is ignored (just as it was for the non-auto mode) and of course, only TKT Variations are available. In this case, the **Auto-Cycle Length** is that which is set by the **Auto TKT Chain** menu (7), which allows you to choose either **2, 4, 6, or 8 TKT Variations**. Since there is also the **base variation** (for index 0), the **TKT Auto-Cycle Length** will thus be **3, 5, 7, or 9 variations**. For **Chained** articulations, and with the **Variation Type** menu (9) set to **All Natural**, the **Auto-Cycle Length** will be the length of the longest **Normal Chain** (for the active articulation). For **Chained** articulations and with the **Variation Type** menu (9) set to either **Natural + TKT** or **TKT + Natural**, the **Auto-Cycle Length** will be the product of the **TKT Length** and the **Natural Length**.

To help clarify this somewhat complex situation, let’s concoct a few sample scenarios and walk through them. Let’s first presume that we set the **Auto TKT Chain** menu to **4 Variations**. Further let’s assume we create a **Fixed** articulation with only **1 Normal Group** (group #8) and assign it to **Articulation 1-1**. Then let’s also create a **Chained** articulation with **4 Normal Groups (3,4,5,6)** and assign it to **Articulation 2-1**. Finally, let’s set the **Auto Trigger Mode** (10) to **Always** and the **Variation Mode** (8) to **Auto Rd Robin**. With this setup, each note played will advance the variation sequencer which will simply advance linearly (ie round robin fashion).

Now, for our first scenario, let’s activate **Articulation 1-1** and, because this is a **Fixed** articulation, it won’t matter what setting we use for the **Variation Type** menu (9). Since there are no Natural variations, the Auto-Cycle Length will be 5 (4 TKT variations + the base variation). So, as we begin to play notes, they will all be sounded from group 8 but with the following TKT index sequence. 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, etc. Note that the cycle length is 5 variations (counting the unprocessed sample).

For our next scenario, let's activate **Articulation 2-1** (the **Chained** articulation). First we'll set the **Variation Type** menu (9) to **All Natural**. Now, as we start playing notes, they will come from groups **3, 4, 5, 6, 3, 4, 5, 6, 3, 4, etc.** **TKT** variations **will not be played** because this is the **All Natural** mode (so only the base variation, **TKT = 0** will be used). To indicate that with the series, we can use the notation 3-0, 4-0, 5-0, 6-0, 3-0, etc. In this notation, the first digit (before the hyphen) is the group that sounds and the second digit (after the hyphen) is the **TKT** Variation index used). For this example, the second digit is always zero because we aren't using any **TKT** variations.

Next, let's try the **Natural + TKT** mode. When we start playing notes for this mode we can expect the following sequence in round robin mode. 3-0, 4-0, 5-0, 6-0, 3-1, 4-1, 5-1, 6-1, 3-2, 4-2, 5-2, 6-2, 3-3, 4-3, 5-3, 6-3, 3-4, 4-4, 5-4, 6-4, 3-0, 4-0, etc. Notice that the Auto-Cycle Length is now 20 variations long (4-Naturals * 5-TKTs). Note also that the Natural variations change the most rapidly and after each cycle of Natural Variations, the **TKT** variation advances one.

Finally, let's try the **TKT + Natural** setting. When we start playing notes we will now get the following sequence: 3-0, 3-1, 3-2, 3-3, 3-4, 4-0, 4-1, 4-2, 4-3, 4-4, 5-0, 5-1, 5-2, 5-3, 5-4, 6-0, 6-1, 6-2, 6-3, 6-4, 3-0, etc. For this setting, the **TKT** index advances the most rapidly and the Natural variations advance only once per 5 **TKT** advances.

3.37 Full-Cycle Random Mode

The **Auto FC Rand** sequencing mode has the same **Variation Type** options and produces the same **Auto-Cycle Lengths** as does the **Auto Rd Robin** mode. However, instead of sequencing through the **Natural** and **TKT** variations in **linear** order, the ordering is **randomized**. For example, with Articulation 2-1 and the All Natural mode, we obtained the round robin pattern **3-0, 4-0, 5-0, 6-0, 3-0, 4-0, etc.** In **Full-Cycle Random** mode, when we first start (and each time a full Auto Cycle finishes), we take the 4 group numbers (3,4,5,6) and we randomly shuffle them into some arbitrary order. Something like having a deck of cards with only 4 cards. So, for example we might end up with **5,3,6,4** the first time we shuffle them. After the sequencer runs through all four variations, we then reshuffle the 4 cards and let's say we get something like **3,4,6,5**. Again we sequence through these 4 variations, etc. This would then remap our round robin sequence to be: **5-0, 3-0, 6-0, 4-0, 3-0, 4-0, 6-0, 5-0**, reshuffle, etc. The shuffling algorithm used is such that it also prevents the last variation from being the first variation in the next shuffle. This is done to avoid the same 2 in a row. Also, keep in mind that when there are both Natural and **TKT** variations, each set of variations are reshuffled at the end of the composite cycle. This randomizing strategy provides the best features of both round robin and ordinary random selection and is an enhanced version of the **FCR** that was used in my old **Ultra TKT**.

3.38 Random Repeatability

While randomizing schemes like **FCR** are very musically useful, sometimes there is a strong desire to be able to repeat a particularly pleasant 'roll of the dice'. But, the very nature of true random numbers would of course preclude that (at least on demand). However, computer generated random numbers are seldom 'true' random numbers but rather deterministically-generated pseudo-random numbers. Therefore, while such sequences appear to be random, if we start with exactly the same conditions each time, we will get exactly the same 'random' sequence. Therefore to support the desire that some of you have expressed to be able to repeat your 'random' sequences, **SIPS 2** has a **User Preference** option, **Sw5**, that when set, causes the random number generator to be reset each time the articulation (or variation mode) is changed.

So, if you are one of those who often wish they could repeat some 'happy pattern' that occurred, try enabling **Sw5**, it might just fulfill your wishes. However, for those of you who just like a lot of musical variety and have little interest in repeatability, leave **Sw5** off to get the most variety from **FCR**.

3.39 The Keep Settings Menu

The **Variation** control parameters, specifically the settings for the controls (6), (7), (8), (9), and (10), can be changed at any time. However, if you have a variation setup that you always want to use with a particular articulation, you can ‘associate’ these settings with the articulation by opening the **Keep Settings** menu (13) and then clicking on **Attach to Articulation**. When you do this, every time you ‘activate’ this articulation, the same **Variation** settings will be restored. On the other hand, if you select an articulation that doesn’t have any **Variation** settings attached to it, the prior settings will remain unchanged when selecting an ‘unattached’ articulation. However, once a set of variation parameters has been ‘attached’ to an articulation, you can no longer change the variation parameters for that articulation. If you need to make a change to the variation parameters after you have ‘attached’ them to an articulation, first select the articulation and then use the **Keep Settings** menu to click on **Sever from Articulation**. You will then be able to modify the variation parameters again (and you can then re-attach them if you wish). You can also sever all attached variation parameter sets by clicking on **Detach All Settings** in the **Keep Settings** menu. All settings will also be detached whenever you **format** (or reformat) an instrument with **all groups checked**.

3.40 MIDI Menu Control

You can optionally change the settings of **SAS** menus (5), (7), (8), (9) and (10) by **MIDI Control** using **Program Change** messages. If you open these five menus, you will see a number in parenthesis on the right side of each option. This number is the **PC** code that will select the associated menu option. Remember that your current articulation will not change unless another valid articulation index is received. Valid indices are the ‘octal set’ from 11 to 98 (which includes the ghost bank, 91 to 98). Non-octal codes such as 20 or 39, or the codes ‘outside of the 11 to 98 range, will not affect the current articulation. Therefore, you can issue a menu option change without impacting any playing notes or changing the current articulation. However, if you are using **Kontakt Instrument Banks**, you may want to disable menu control since you will be using **MIDI PC** commands to change instruments. You can disable the **SAS Menu Control** by simply enabling **Sw-2** in the **User Preferences** (see section 2.5).

3.41 Group-Start Programming

As mentioned throughout the prior sections, **SIPS 2** must have **exclusive use** of the **Group-Start** parameters which it uses for **Articulation**, **Variation**, and **Auditioning** control. Normally, you should not have to deal directly with group-start programming. In fact, it is highly recommended that you never change these values directly but rather always use the **Assign Groups** command menu to configure your instruments. However, for those of you who are interested in how **SIPS** uses the group-start parameters, this information is included here.

SIPS configures every instrument group with either 3 or 4 group-start **modules**. **Groups 0 and 1** employ a **4th module** solely for the purpose of displaying the keyswitch groups on the Kontakt keyboard. Let’s defer talking about this little complication for now and just ‘pretend’ that **all groups** are configured with just three **modules** each. All three **modules** are setup to be responsive to a single CC value. **Module 1** is logically **ANDed** with **module 2** and the group will be enabled if **CC126 = m1 and CC127 = m2** (where **m1**, **m2** are the values the modules are set to ‘start on’). Further, **modules 1 & 2** are then logically **ORed** with module 3 which is set to ‘start on’ **CC126 = 90**. Therefore, the full expression for when a group is ‘started’ (or enabled) is given by:

$$\text{CC126} = \text{m1 and CC127} = \text{m2 or CC126} = 90$$

For example, if **group 5** has **m1 = 21** and **m2 = 31**, then **group 5** will be enabled if the **SAS** sets **CC126** to **21** and **CC127** to **31**. Or, alternatively, **group 5** will also be enabled if **CC126 = 90**. The **m1 setting** specifies the **articulation number** that a group belongs to and the **m2 setting** specifies the **class/type code** (to be discussed shortly).

Since all groups will be enabled when **CC126 = 90**, when you activate the **Group Audition** function, the **SAS** sends out **CC126 = 90** and then uses the **disallow_group/allow_group** functions to enable only the highlighted group. In all other modes, **CC126** is set to the ‘active’ articulation index from **11 to 98** (including the ghost note bank). When an instrument is newly formatted, all groups are set to **m1 = 99** and **m2 = 0**. And, since **99** is not a valid articulation index, all groups are effectively unassigned and will not respond to any articulation code.

When a group is assigned, **m1** is set to the articulation number and **m2** is set to convey the articulation class and group type. The group type codes are **0 = Normal**, **32 = Inside**, and **64 = Release**. The **Fixed** articulation class code is **31** while the **Chained** articulation class code is a value between **1 and 16**, representing the position in the chain occupied by the group. Note that for consistency with the TKT variations, the chain positions for ‘natural’ variations are referred to ‘externally’ as 0 to 15 (with position zero representing the base variation, or no variation). However ‘internally’, the chain positions are numbered from 1 to 16 since zero is used to indicate an unclassified articulation.

The actual value programmed for **m2** is the sum of the class code and the type code. Some examples may help to clarify the situation. If a group is assigned to **Fixed articulation 5-3** as a **Normal Group**, **m1** will be set to **53** and **m2** will be set to **31**. If a group is assigned to **Chained articulation 4-7** as an **Inside Group**, **m1** will be set to **47** and **m2** will be set to **cp + 32**, where **cp** is the chain position index (1..16) assigned to this group by the **SAS**. Whenever a configured instrument is reloaded, the **SAS** scans all the group start parameters and constructs an **Articulation Table** in which it has the max length of each Chained articulation discovered.

Now, we need to discuss the 4th group-start module used with groups 0 and 1. These groups are configured such that the logic expression they will respond to is as follows:

(CC126 = m1 and CC127 = m2) or (CC126 = 90 and not KeyRange)

The key range used for **group 0** is the same as that set for the **Art/Var** keyswitches and the key range for **group 1** is the same as the **Bank** keyswitch range. The **SAS** logic is such that **KeyRange** is always ‘forced’ to be true when **CC126 = 90** so that **groups 0 and 1** can be auditioned the same as any other group. The **KeyRange** is not included for the purpose of qualifying the group-start but rather, just to ‘force’ Kontakt to display the keyswitch ranges on its keyboard. When the keyswitches are de-allocated, the **KeyRange** settings are inverted (ie min > max) so no ‘pink’ keys will be displayed.

4.0 SIPS Legato Script

4.1 Introduction

Simulating a legato effect with a Script is all about connecting notes, or more specifically, controlling note transistions. When you play two overlapping notes, the Legato Script must ‘retire’ the old note and ‘establish’ the new note. But the old note can’t just be ended and the new note started or it won’t sound like legato playing. For a brief period of time, called the ‘transistion’ period, components of both the old and the new notes are present.

The two major functions performed by a Legato Script are Crossfading and Bending. The Crossfading function controls the relative volume of the old and new notes while the Bending function warps the pitches of the two notes during the transistion period. The objective is to do all this in such a way that the transistion from the old to the new note is done smoothly and *sounds*, for all practical purposes, just as it would when done with a real instrument by a real player. This turns out to be rather ‘a tall order’ because there are many conflicting requirements and the current KSP tool kit is rather limited. However, the **SIPS Legato Script**, when *properly set up* is capable of providing some extremely realistic and convincing legato sounds. Of course to *properly set up* the **SLS**, it is important that you understand what all of the knobs and such do. While a number of presets are provided, these should be viewed only as starting points for your further customization on an instrument by instrument basis. This is important if you are expecting the **SLS** to work with a wide variety and quality of sampled instruments.

4.2 Playing Legato

To play a phrase legato, the **SLS** requires that all the ‘inside’ notes of the phrase overlap. The first note of a new phrase is sensed by the script based on the fact that no other notes are still sounding (ie no keys are still held down and the sustain pedal is off). The **SLS** plays the first note of a phrase with its normal attack and if this first note ends before another note starts, the script will interpret the 2nd note as the start of another new phrase. Such ‘phrase-starting’ notes are not affected in any way by the script (unless the **SAS** ‘ghost note’ feature is used to force the first note of a legato phrase to be processed, see **section 3.31**). However, if the first note is still sounding when the 2nd note starts, the notes are considered overlapping **and the Legato Effect will be generated**. Similarly with the 2nd and 3rd notes, the 3rd and 4th notes and so on. When the last note played ends before another note is played, the script considers it to be the ‘end of the current phrase’. Unless you are using the special ‘**Key-Lift**’ release mode (see **section 4.6**), the amount of time overlap is not important to the script.

4.3 Use of the Sustain Pedal

Normally, you tell the script that you want the legato effect for a pair of notes by not releasing the first note’s key until you have pressed the next note’s key. Alternatively, if you depress the sustain pedal during the first note, then even if you release the key before striking the next key, if the sustain pedal is still down when the second key hits, the script will ‘see’ the notes as overlapping and add the legato effect. Starting with **V1.5**, **SIPS** processes the sustain pedal with its own logic (K2’s normal handling of the pedal is disabled). With **SIPS** in control, if you hold the sustain pedal down while playing a phrase, there will be no excessive ‘buildup’ of polyphony because the **SLS** fades out the prior note when each new note starts. In fact, the polyphony would never exceed two if it weren’t for the fact that some instruments have a long release tail. Because of this, if you play a fast legato passage, polyphony can exceed two for brief periods when multiple note tails may overlap (but, this will happen whether or not you use the sustain pedal). So, you can optionally play legato either by overlapping the keys or by using the sustain pedal (whichever is more convenient). **However, if your phrase contains two or more notes in a row that are the same pitch, you can’t actually overlap the keys. So, for this case, you will need to use the sustain pedal to legato-connect the notes.** Note also, when using the special **Key-Lift** release mode (see **section 4.6**) with the sustain pedal, both the key **and** the pedal must be released to shorten the fade-out note.

For those users who are willing to forego the sustain function of **CC64** so they can use the pedal as an ordinary assignable CC, **SIPS 2** includes a preference option to disable the sustain function. With preference switch **Sw1** enabled, **CC64** will no longer provide the sustain function and thus can be assigned to control whatever you wish. However, this is not recommended because the ability to play a series of legato-connected notes of the same pitch is a very valuable function. Since there are add-on accessories that can provide additional MIDI foot controllers for a nominal cost, you don't need to give up the sustain function just to get general pedal switch control.. But, if you still want to give up the sustain function, preference switch **Sw1** is there for you.

4.4 Mode Carryover and Playing Chords

Prior versions of the **SLS** allowed the playing of chords in **Solo** and **Bypass** modes but, there were several unforeseen problems with its implementation that made it awkward to use in actual practice. Starting with **V1.5**, the MIDI-controllable **SLS Mode Menu** function has been completely redesigned to provide a much more musician-friendly *modus operandi*, including a more musical way of introducing chords.

For **SIPS 2**, the **Mode Menu** choices are **Portamento Mode**, **Legato Mode**, **Solo Mode**, and **SLS OFF**. **Portamento Mode** will be discussed in **section 4.11**. Of course **Legato Mode** is the primary mode of the script. **Solo Mode** (as in prior versions of **SIPS**) allows for playing monophonic lines where each new note played forces the prior note to its release phase (but provides no crossfading or bending like **Legato Mode** does). However, **Solo Mode** no longer allows playing chords (with or without the sustain pedal depressed). Rather, the sustain pedal merely insures that the individual notes will be overlapped (as it does for the **Legato Mode**). Note also that **Solo Mode** now ignores the **Leg Ofst** settings (as it should have all along).

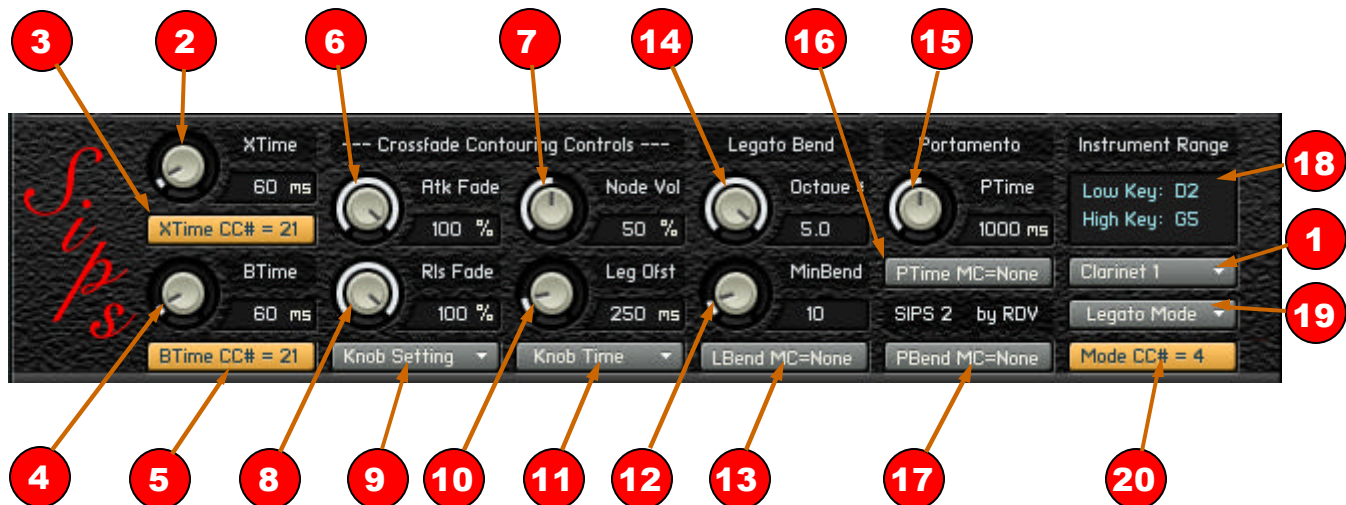
For **SIPS 2**, the **SLS OFF** mode completely removes the legato and solo-mode effects of the script and thus allows normal playing, **including the playing of chords if desired**. And, since the **Mode Menu** can be assigned to a CC, you can easily switch back and forth between **Legato** (or **Solo**) **Mode** and **SLS OFF** in real time as you play. However, unlike prior versions of the **SLS**, the transitions and overlaps provided are much more musical. For example, if you are playing a legato phrase and hold the last note of the phrase as you switch from **Legato Mode** to **SLS OFF**, subsequent notes will play normally (without the legato or solo-mode effect) but, the last held note will carry over until released. Similarly, if you play some passage in the **SLS OFF** mode and hold the last chord played as you switch back to **Legato Mode**, subsequent notes will play legato but the last held notes from the **SLS OFF** mode will continue to sound until released.

For lack of a better name, we might call this feature '**Mode Carryover**'. Basically it simply means that the last note or chord sounding in the prior mode can be overlapped or carried into the next mode. This can be done either by holding the key (or keys) down during the mode change, or by using the sustain pedal to accomplish the same thing. Hopefully, the logic used to implement this feature will provide for a more musically-satisfying and more intuitive behaviour for real-time mode changes.

SIPS 2 extends the **Mode Carryover** feature to include mode transitions to and from the **Portamento Mode**. If you are playing a legato phrase, and while you are holding a note (or if you activate the sustain pedal) and then switch to the **Portamento** mode, the last note of the legato phrase will 'carryover' and become the first note of the glide/bend. Once in the **Portamento** mode, the sustain pedal will have no further effect on new notes played but it continues to affect the carryover note until released. However, when playing a portamento phrase, if you depress the sustain pedal while holding the last note of the portamento phrase and then switch back to the **Legato** mode (with the pedal still depressed), if you release the key, the note will remain sustained until you also release the pedal. To summarize, the sustain pedal has no effect on the 'inside' notes of a portamento phrase, but it will sustain the mode transition notes at either end of the phrase (see **section 4.11** for details of the Portamento Mode).

4.5 Control Panel Layout

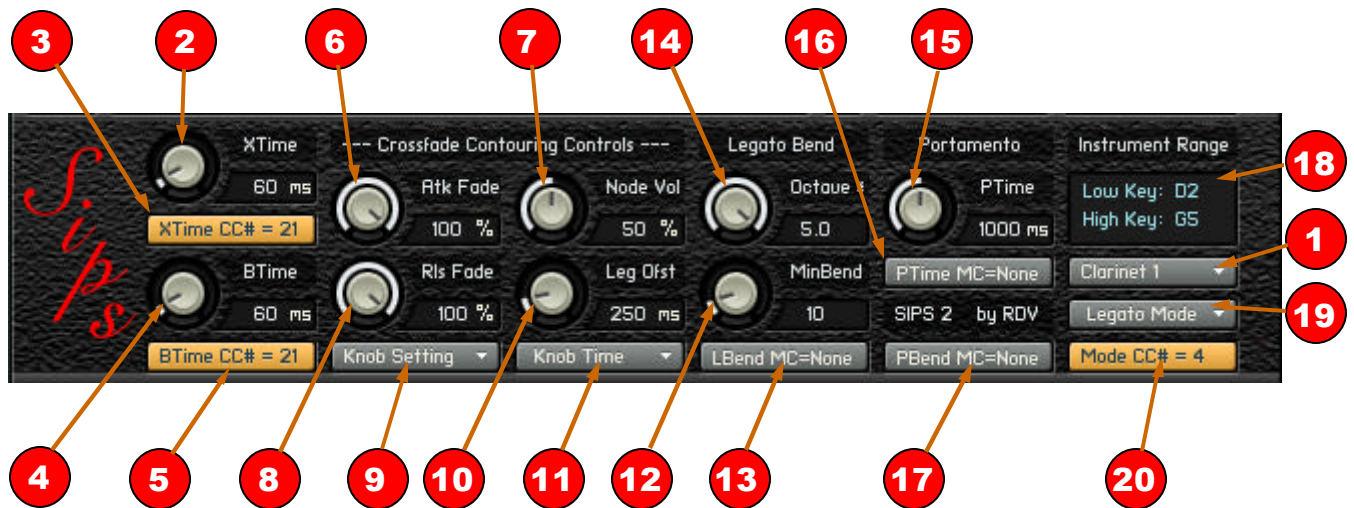
The Control Panel layout and legend for the **SLS** is shown in **Figure 4-1** (at the top of the next 2 pages).



The Legato Script Control Panel

Figure 4-1

- (1) **Preset Selector** drop-down menu. Selecting an Instrument or Instrument Class from this menu will set certain key parameters to a good starting point for you to further customize by ear. You can also store and recall your own **User Presets** here. Plus, you can access other useful functions via this Menu. (including **Rename** and **Import/Export**).
- (2) **XTime** knob. Sets the total **crossfade** time for note transitions in milli-seconds. If **RlsFade** (8) is set to less than 100%, **XTime** is still the time for the **fade-in** of the new note but the **fade-out** time of the old note will be less (specifically $\text{RlsFade} * \text{XTime}$).
- (3) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **XTime**. You can also set it to control only a subrange of **XTime** if desired (see section 2.4).
- (4) **BTime** knob. Sets the amount of time (starting when **XTime** begins) over which legato bending occurs.
- (5) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **BTime**. You can also set it to control only a subrange of **BTime** if desired (see section 2.4).
- (6) **AtkFade** knob. Sets the percentage of **XTime** that the first segment of a 2-segment fade-in contour uses to rise to **NodeVol**. **NOTE:** If **AtkFade** is set to 100%, **NodeVol** (7) is ignored and a single-segment, linear fade-in will be used over the full **XTime** period.
- (7) **NodeVol** knob. Sets the percentage of **Vmax** attained by the first segment of the fade-in contour.
- (8) **RlsFade** knob. When the release-mode menu (9) is set to '**Knob Setting**', this knob sets the fraction of **XTime** for the **pre-release** fade-out of the prior note. If **RlsFade** is set to 100%, the prior note fades out over the full **XTime** interval using a single segment contour.
- (9) **Release-Mode Menu** drop-down button. When **Knob Setting** is selected, the **RlsFade** knob (8) setting determines the fraction of **XTime** used for **pre-release** fade-out. When **Key-Lift** is selected, the **RlsFade** knob is ignored and the **pre-release** fade-out time is controlled by your keyboard legato note overlap time (see section 4.6).
- (10) When (11) is set to **Knob Time**, this knob sets the sample-start offset for 'inside' notes of a legato phrase or portamento bend. In **Legato Mode**, this is the **Leg Ofst** knob and in **Portamento** mode it is the **Port Ofst** knob (see section 4.11). In **Legato Mode**, when (11) is set to **Knob + Rand**, this knob sets the 'base' offset time but when (11) is set to either **Auto Mode 1**, **Auto Mode 2**, or **DFD Mode**, the knob setting is not utilized.



- (11) **Offset-Mode Menu.** Selects how sample-start offset time is determined when in **Legato Mode**. **Auto Mode 1** uses time since the start of the prior note. **Auto Mode 2** uses time since the start of the current legato phrase. **Knob Time** uses the fixed time set with the **Leg Ofst** knob and **Knob +Rand** uses a randomly varying offset with the knob setting as the minimum. The knob offset add-on will be a randomly selected multiple of 50ms over the range from 0 to 400ms but the same offset will never be used twice in a row. When set to **DFD Mode**, the knob setting is ignored. When (19) is set to **Portamento Mode**, the offset knob becomes relabeled as **Port Ofst**. In Portamento Mode, only the **Knob Time** and **DFD** modes are available. Operation of the **DFD Mode** is described in **Section 4.10** and operation of the **Portamento Mode** is described in **Section 4.11**.
- (12) **Minimum Bend Knob.** Sets the amount of legato bend (in cents) for a played interval of one semitone.
- (13) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **MinBend**. You can set it to control only a subrange of **MinBend** if desired (see **section 2.4**).
- (14) **Octave X knob.** Sets the factor by which **MinBend** is multiplied for a played interval of one-octave.
- (15) **PTime knob.** Sets the **Portamento Glide Time** in **Glider** mode or PBend Smoothing in **Bender** mode.
- (16) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **PTime**. You can also set it to control only a subrange of **PTime** if desired (see **section 2.4**).
- (17) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned as the **Portamento Bender** control. Note: If no controller is assigned, **Glider** mode will be used.
- (18) **Instrument Range Box.** Displays the Low/High Key of the instrument range currently set by the **SAS**.
- (19) **SLS Mode Menu** drop-down button. Used to select the **Legato**, **Solo**, or **Portamento** modes or, alternatively, to disable the **SLS** when set to **SLS Off**. In Solo Mode, each overlapping note played will terminate the prior note, but no legato crossfading, bending, or offset is performed. When **SLS OFF** is selected, the script has no effect on the MIDI stream other than to handle the sustain pedal response (see **section 4.3**). This mode can be used to play chords if desired (but without the legato or solo effect).
- (20) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to select the **SLS Mode**. The 4 modes selected by the **CC's** value is as follows:

127 selects **Portamento Mode**
64 to 126 selects **Legato Mode**

1 to 63 selects **Solo Mode**
0 selects **SLS OFF**

4.6 Understanding the Crossfade Contouring Controls

This section of the User's Guide will focus on the Crossfade Function and its parameters. The Legato Bend Function and its parameters will be discussed in the next section. The purpose of Crossfading is to gradually get rid of the old note while simultaneously welcoming the new note. When this script was first written, the `change_vol()` function was too noisy to be used for crossfading so the **SLS** was designed to use the fade functions with a novel, 2-segment shaping. Since this resulted in being so musically satisfying, there is little incentive to rewrite this code just for the sake of using the now-improved `change_vol()` function. However this improved function is now profitably utilized in the equal-power crossfade of the new portamento feature.

The simplest form of a crossfade is depicted in **Figure 4-2**. The old note is faded out over the period **XTime** and the new note is faded in over the same period of time. In **Figure 4-2**, the solid **Red Line** depicts the old note fading out from its full level, **Vmax**, to silence. The solid **Green Line** depicts the new note fading in from silence to **Vmax**. This type of linear crossfade, when done with the KSP fade functions, provides a fairly decent effect and has been used as the basis of several simple legato scripts. Naturally this 'bare-bones' form of crossfade works better with some kinds of instruments than it does with others.

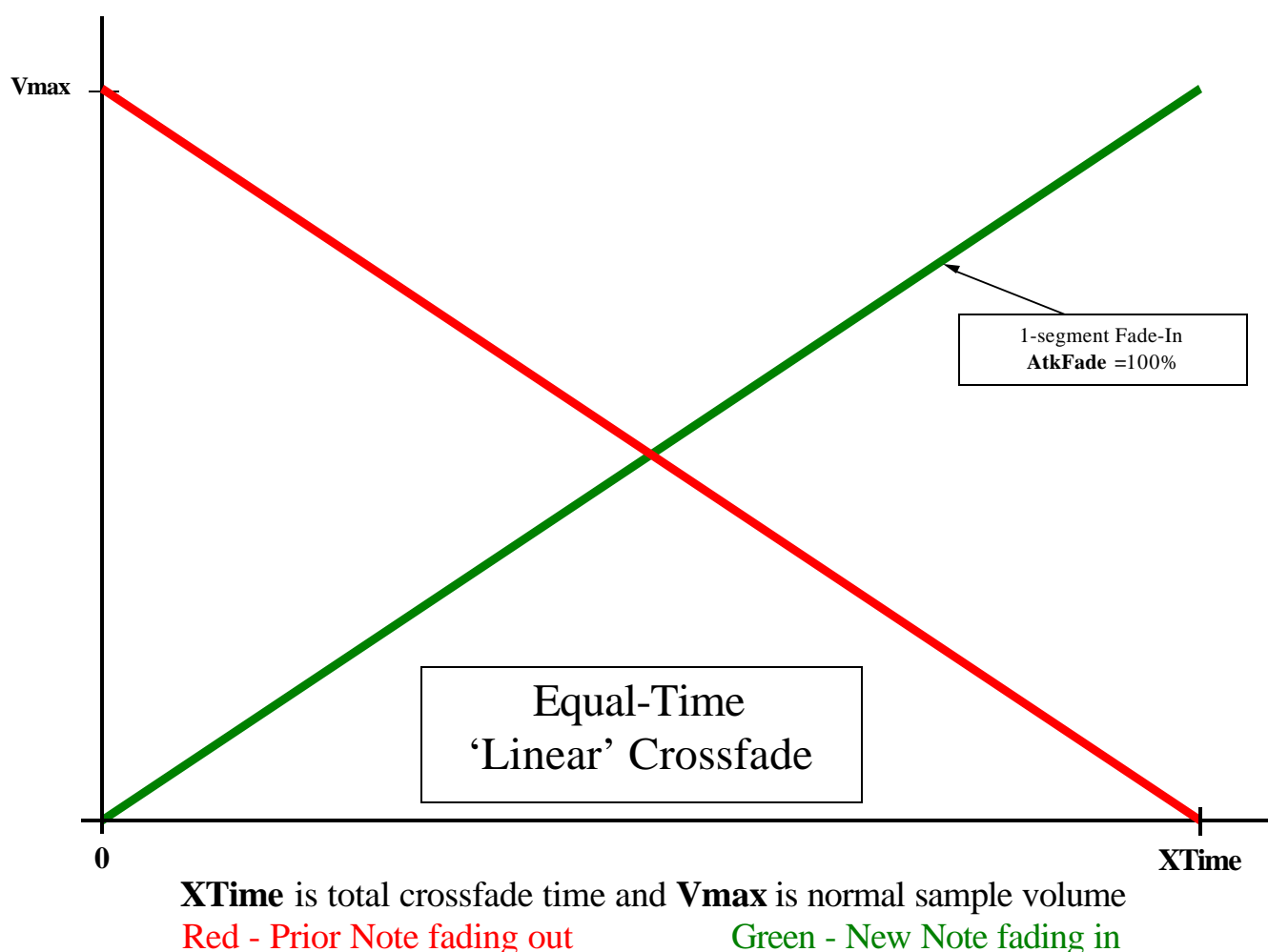


Figure 4-2

The next logical improvement in getting to a more convincing legato transistion is to remove the attack transient of the new note. With the KSP this can be done by starting the new note's playback farther into the sample. The **SLS** provides two automatic and two knob setting modes for determining the amount of sample-start offset. The **Auto Modes** produce an offset equal to the time since the start of the prior note or the time since the start of the current phrase. In the **Knob** modes, the offset is determined by the setting of the **Offset Knob** alone or together with a randomly chosen add-on. Additionally, **SIPS 2** provides a new **DFD Mode** which is discussed in **section 4.10**. These five offset modes are selected with the drop-down menu beneath the **Offset Knob**.

Different instruments require different **XTime** settings. For example, something on the order of 60ms works out well for a Clarinet but strings usually require a much longer crossfade time. For example, a Cello will typically need **XTime** settings from 300 to 600ms. For short **XTime** settings, the simple linear crossfade of **Figure 4-2** can be made to work quite well. However, as **XTime** gets longer, several undesirable things begin to happen. In order to avoid a 'chorusing-like' effect, as well as a general mudiness in the transistion sound, it is necessary to shorten the fade-out time relative to the fade-in time. So if the total fade-in time is always considered to be **XTime**, then the fade-out time must often be reduced to some fraction of **XTime**.

One way to do this is to provide an adjustment that sets the ratio of fade-out time to **XTime**. With such a knob set to say 25%, the crossfade profile would be as shown in **Figure 4-3**. The problem with this scheme is that a serious dip in volume occurs before the fade-in gets up high enough. But, apart from this volume dip, the legato transistion itself sounds much better than with the equal-time crossfade of **Figure 4-2** (at least for longer **XTime** values).

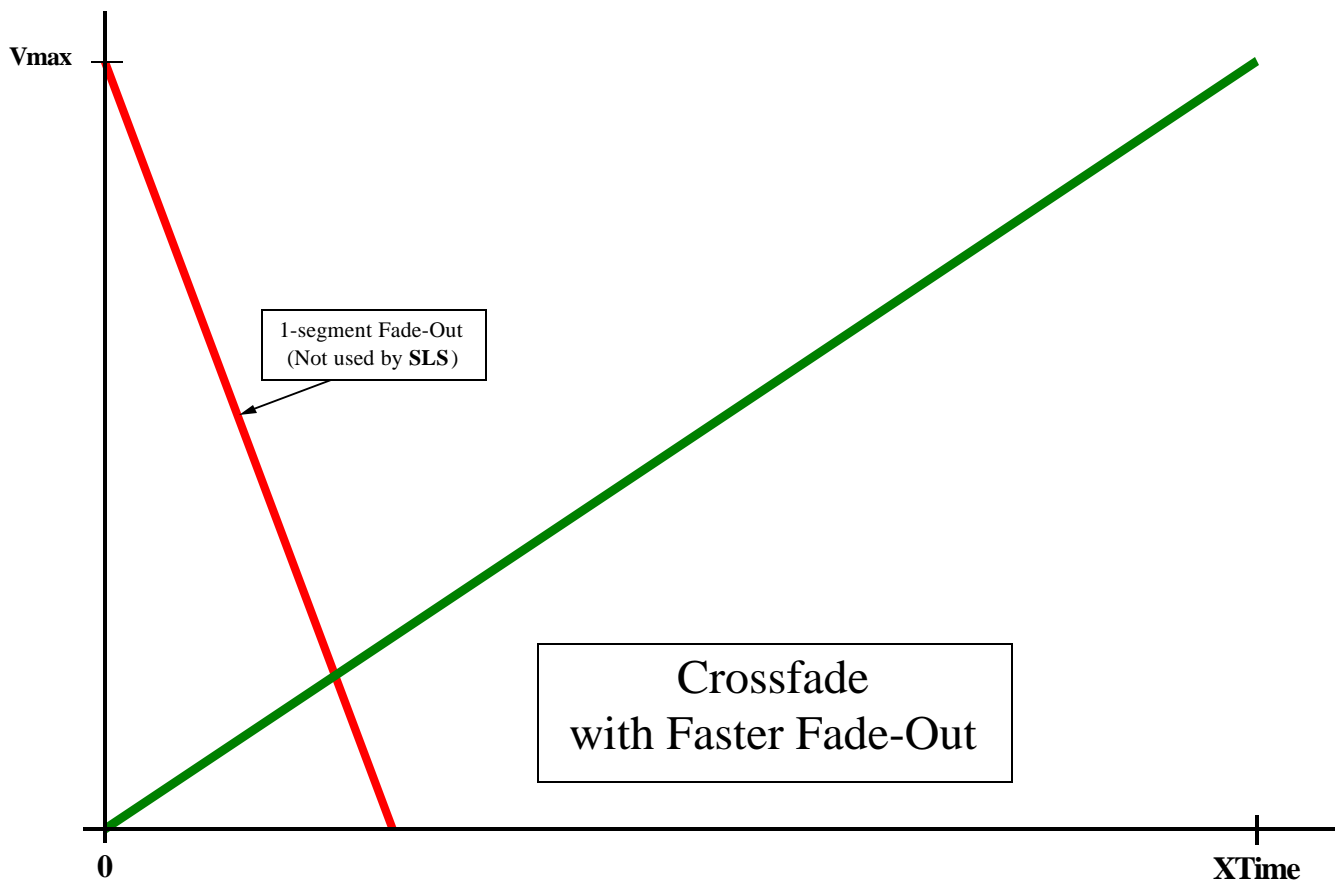


Figure 4-3

Another way to shorten the fade-out time is depicted in **Figure 4-4**. Here the old note fades out along the same path as it would in **Figure 4-2** until 25% of **XTime** elapses. Then, the note is terminated with the **note_off** function. This causes K2 to act as though the key was released which in turn advances the sample playback to the ‘release’ phase of its envelope. Thus, this scheme effectively has a 2-segment fade-out curve which greatly shores up the volume dip with little or no unfavorable effects. For high **XTime** settings, this scheme provides a better overall quality legato transition, yet greatly reduces the volume dip problem.

The reason this scheme is so beneficial is that the volume of the old note stays up where it is needed the most and yet the note is reduced to silence in a period not much longer than that depicted in **Figure 4-3**. The 2-segment fade-out shown in **Figure 4-4** is the technique used in the **SLS**. In addition, the **SLS** provides two different mechanisms for specifying what percentage of **XTime** is used for the first segment (the pre-release fade-out time). The selection is made with the release mode drop-down menu (9) just under the **RlsFade** Knob. When the **Knob Setting** mode is selected, the **RlsFade** knob sets the fraction of **XTime** given to the **pre-release** fade-out time. For example, if **XTime** is set for 500ms and **RlsFade** is set to 25%, then the pre-release fade-out interval will last for 125ms before the prior note moves on to the envelope’s release phase.

When **Key-Lift** is selected as the release mode, the **RlsFade** knob is ignored and instead, the **pre-release** fade-out ends when the prior note’s key is released.. The **Key-Lift** time is governed by your playing style and how much you typically overlap the notes when playing legato. So, if you hold the prior note longer (ie make the overlap longer), the **pre-release** fade will be longer and if you make the overlap shorter, the **pre-release** fade will be shorter. Thus, you can vary the **pre-release** in ‘real time’ as you play (by varying the length of ‘overlap’ that you use). Depending on your keyboard skills and/or your playing style, you might find this mode to be fairly comfortable to use and it may give you some additional measure of control over the sound.

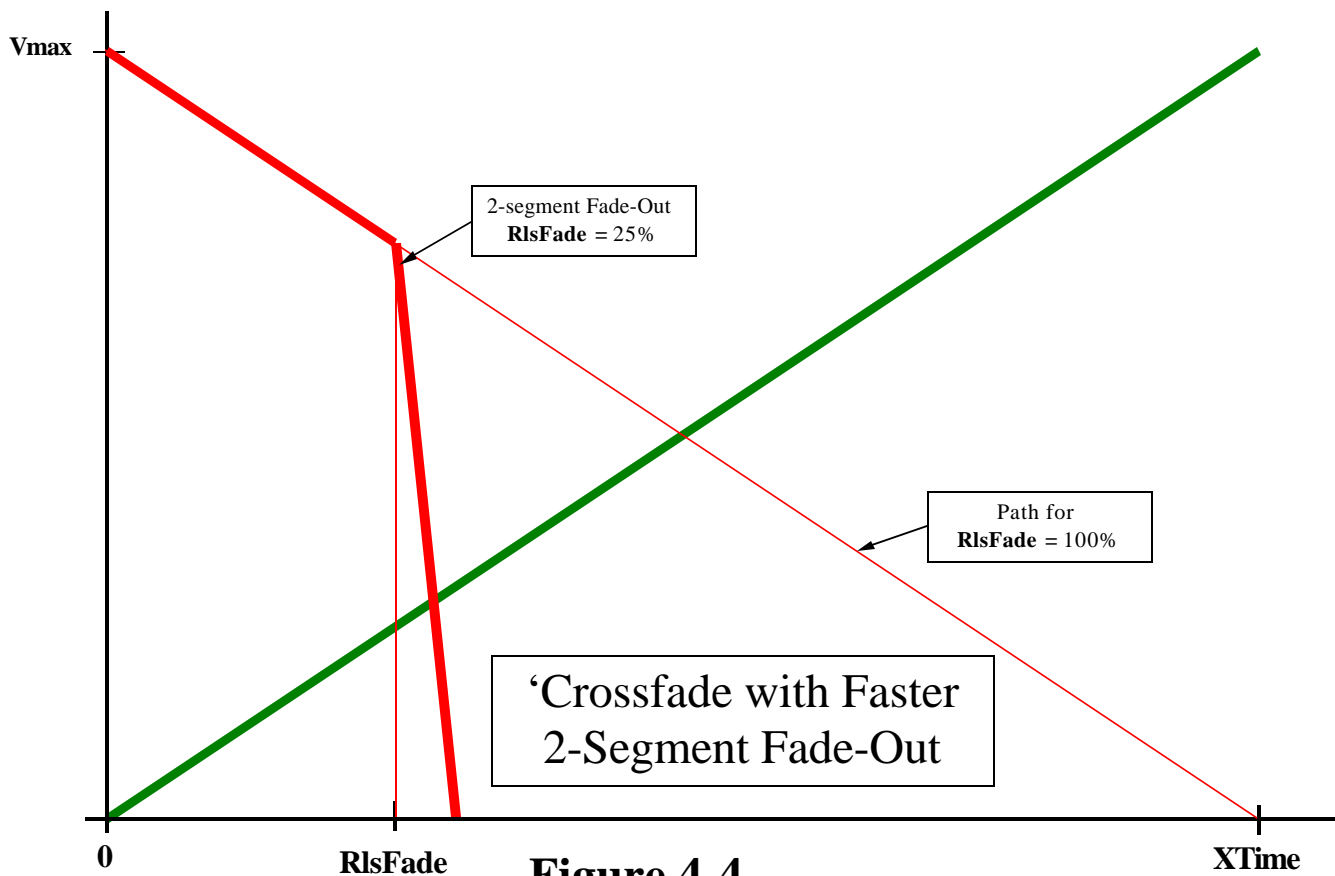


Figure 4-4

Now, while the fade-out depicted in **Figure 4-4** has two segments, unlike the first segment, the second segment is not directly under control of the script. The time of the second segment is governed by the release phase of the sample itself which in turn may be shortened by the release phase of its envelope. However, in spite of the fact that the script cannot directly set this release time period, the scheme of **Figure 4-4** for controlling the fade-out still works out to be quite musical in most situations. One reason this is so is because instruments that require a shorter total fade-out time than the fade-in time (ie **XTime**) are those instruments that have a slower attack time and a correspondingly slower release time. For such instruments, the natural release time is usually such that you can find a **pre-release time** ($\text{RlsFade} * \text{XTime}$) that dovetails nicely with it.

So, for faster attack instruments (which usually use rather short **XTime** settings), a **RlsFade** setting near 100% (as depicted in **Figure 4-2**) usually works out quite well. And, for many of the slower attack instruments (requiring longer **XTime** settings), all that is needed is to reduce the **RlsFade** setting as depicted in **Figure 4-4**. However, for some instruments that need higher **XTime** settings, the scheme depicted in **Figure 4-4** has a few shortcomings. During the legato transition, there may be a *small* (but undesirable) volume dip that occurs. Moreover, when **XTime** and **RlsFade** are set for the smoothest and best sounding legato effect, sometimes the response at high **XTime** values becomes kind of sluggish for playing faster passages. To overcome these problems, we need to get the new note up faster and yet not materially reduce the total fade-in time (to preserve the nice smooth legato sound). To accomplish this, the **SLS** provides a 2-segment fade-in that is faster at first and slower later. This is sort of the inverse of the fade-out curve (together providing something like an equal-power crossfade). Both the 2-segment fade-out (red) and the 2-segment fade-in (green) curves are depicted in **Figure 4-5**.

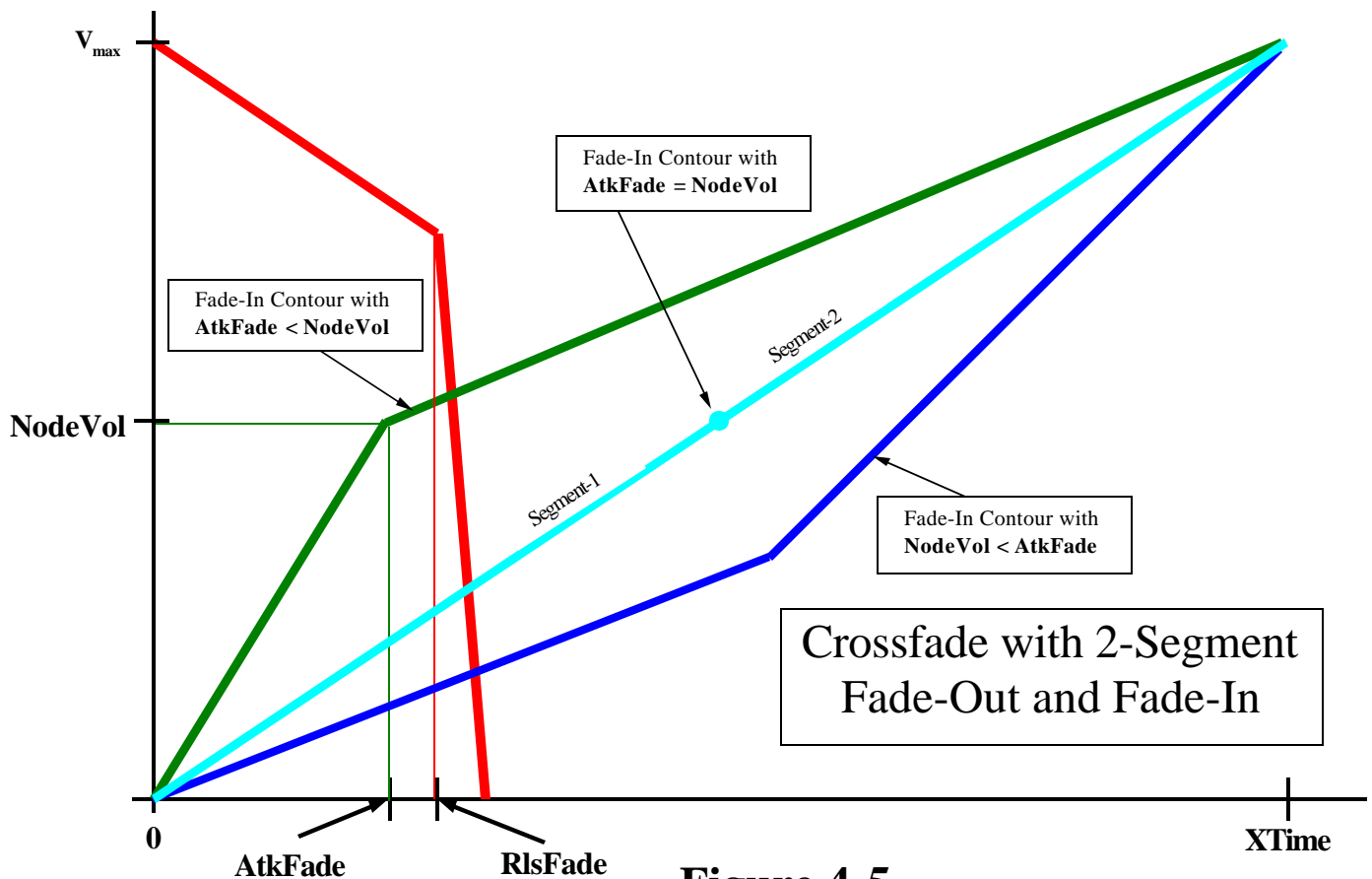


Figure 4-5

Two adjustment knobs are provided for setting the fade-in contour. The **AtkFade** knob sets the percentage of **XTime** over which the first segment of the fade-in curve rises from silence to where the 2nd segment starts. The **NodeVol** knob sets the percentage of **Vmax** that the first segment of the curve rises to before it changes its slope. Thus segment-2 of the curve rises from **NodeVol*Vmax** to **Vmax** over the remainder of **XTime**. With these two knobs a variety of different contours can be set, and with the proper settings, you will usually be able to overcome the volume dip and also make the script more responsive to faster passages (even at long **XTime** settings). Note that if you set **AtkFade** to 100%, the value of the **NodeVol** setting is ignored and a single-segment fade-in curve such as that depicted in **Figure 4-4** is obtained.

It should be mentioned that while these two controls can be set many useful ways, they can also be set in many **nonsensical** ways that should be avoided. To clarify this a little, consider the following. When **AtkFade** is set to the same value as **NodeVol**, the 2-segment fade-in contour coalesces effectively into a one-segment fade-in because segment 1 and segment 2 both have the same slope. For example, if **AtkFade** and **NodeVol** are both set to 50%, the (light blue) fade-in curve of **Figure 4-5** will be the result even though there are two segments. The first segment will traverse the first 50% of the straight-line and the second segment will traverse the last 50% of the **same straight line**. Now if **NodeVol** is set to a value lower than **AtkFade**, the contour will dip at the node which is just the opposite of what you will likely be trying to do with these controls. This sort of situation is depicted by the dark blue contour shown in **Figure 4-5**.

Thus when attempting to correct for side effects of the simple, 1-segment linear fade-in contour, you might begin by setting both **AtkFade** and **NodeVol** to 50%. This will start you out with the same sound as you had when **AtkFade** was still at 100% (ie a single-segment linear fade-in). Then, as you begin to tweak these controls, you will want to move them in such a way that keeps **AtkFade** less than **NodeVol** so that you don't get an inverse contour like that of the dark-blue curve in **Figure 4-5**. Remember that what you are trying to do is to make the segment-1 slope steeper and the segment-2 slope flatter as it approaches **Vmax**.

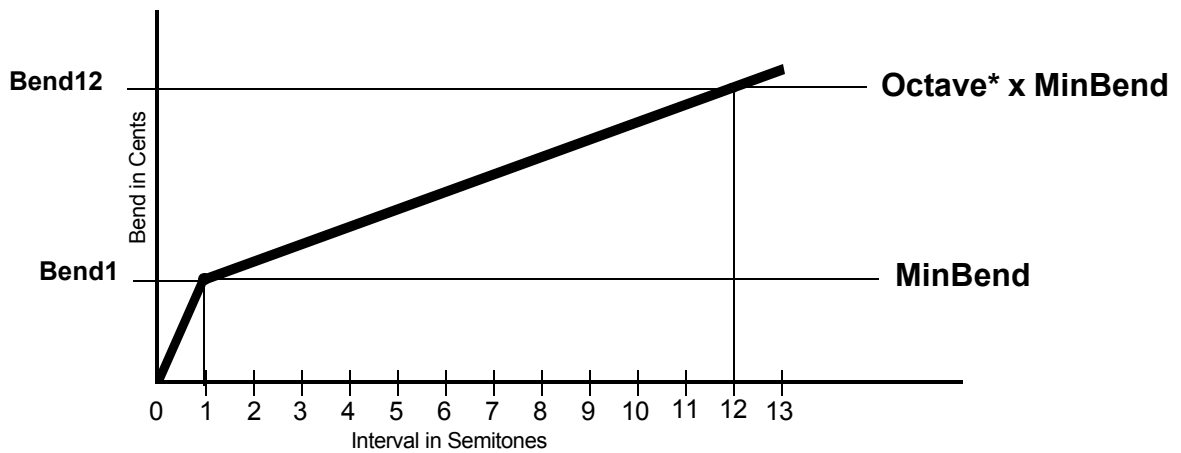
4.7 MIDI Control of the Crossfade Function

The **SLS** also provides for MIDI Control of **XTime**. The *assignment-button* beneath the **XTime** knob allows you to assign a **MC** to provide control of **XTime**. Furthermore, the range of control of the assigned **MC** can be set to whatever fraction of **XTime** you might want to zero-in on (as described in **section 2.4**).

4.8 Understanding the Bend Contouring Controls

When musicians play legato passages on real instruments, there is usually a certain amount of pitch bend that takes place in moving from the prior note to the new note. Just as the prior note smoothly crossfades into the new note, the prior note's pitch also starts to bend toward the new note and the new note bends toward its target value (from the side of the prior note). Thus, if an up interval is played, the old note starts to sharpen as it fades out while the new note fades-in somewhat flat as it bends toward the desired center value. Conversely, if a down interval is played, the old note starts to flatten as it fades out while the new note fades in somewhat sharp as it bends toward the desired center value.

The amount of bend depends on the instrument class but, for a given instrument, the amount of bend usually increases as the played interval widens. The **SIPS Legato Script** allows you to set the amount of bend for a played interval of a minor 2nd (ie one semitone) and also lets you specify the linear rate at which the bend will increase as the played interval increases. **Figure 4-6** illustrates the relationship between bend amount and the played interval. In **Figure 4-6**, the left side annotation identifies the amount of bend for a played interval of one semitone as **Bend1** and the amount of bend for a played interval of one octave as **Bend12**. The right side annotation shows these same two bend levels related to the **SLS** parameters of **MinBend** and **Octave*** (octave factor). The **MinBend** knob setting is precisely the same as **Bend1** (ie the amount of bend for a played interval of a semitone).



Pitch bend as a function of the Interval
between a pair of notes played Legato

Figure 4-6

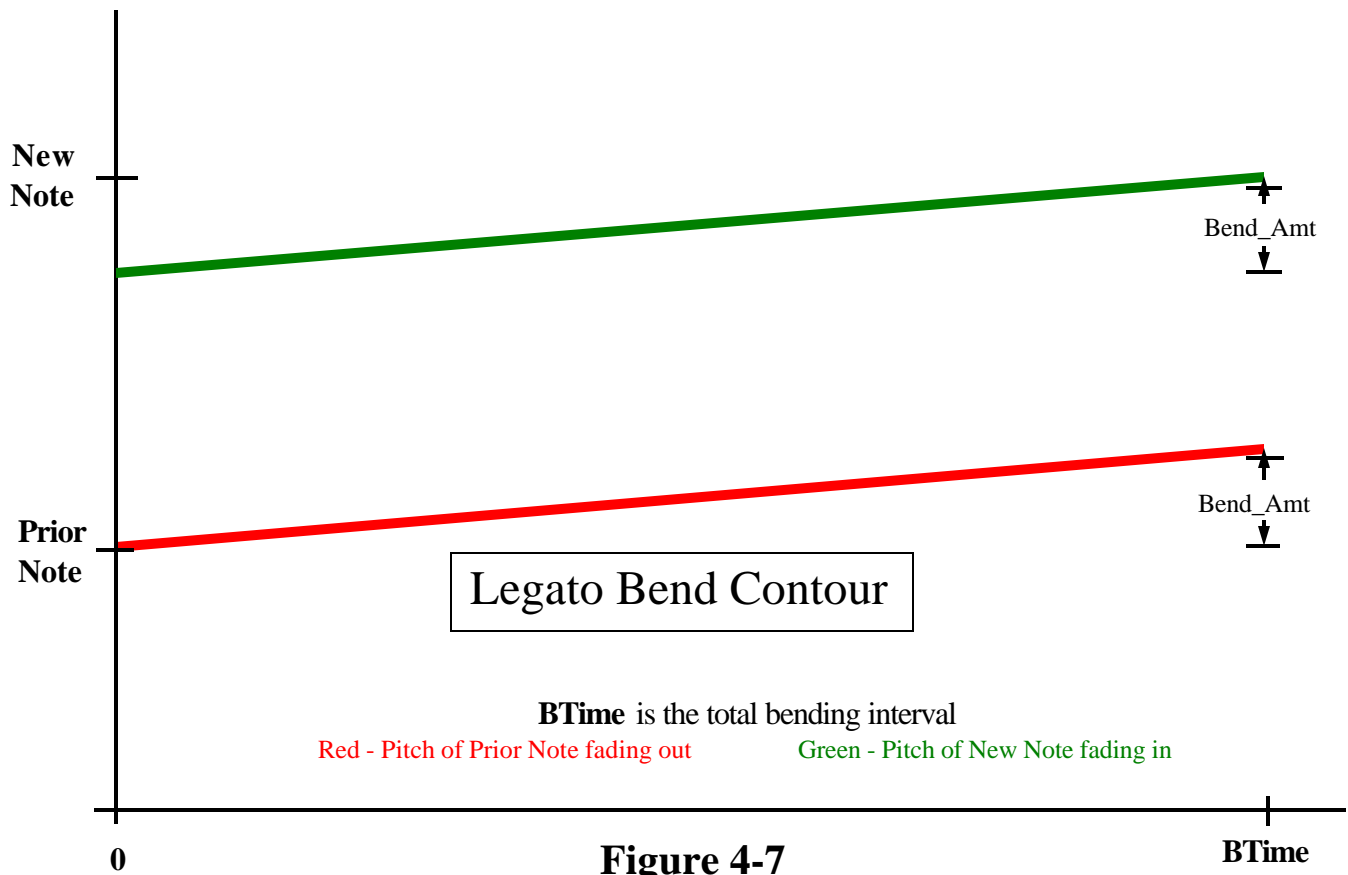
So, you should set the **MinBend** knob for the bend (in cents) you desire for a played interval of a minor 2nd. Then you can set **Octave*** as follows. Determine how much bend you want for a played interval of one octave (**Bend12**). **Octave*** should then be set to $\text{Bend12} / \text{MinBend}$. For example, if you want a bend of 20 cents for a minor 2nd, and a bend of 45 cents for an octave, set **MinBend** = 20 and **Octave*** to $45 / 20 = 2.3$. Note that if you want the same amount of bend for all played intervals, you can accomplish that by simply setting **Octave*** = 1.0.

Now in addition to setting **MinBend** and **Octave***, you also need to specify the time interval over which the bend takes place. The bend always starts when the crossfade starts (ie when **XTime** begins) and continues for the time interval set by the **BTime** knob. Thus the rate of pitch bend is governed by the amount of bend and the time over which that bend takes place. Usually you will want a **BTime** setting close to that of **XTime** but you can set it shorter or longer as the occasion warrants. Since **BTime** and **XTime** can both be assigned to a **MC**, if you always want **BTime** to track with **XTime**, you could simply assign the same **MC** to control both.

The remaining issue that needs to be discussed is how the bend is shaped over the **BTime** interval. Currently, the **SLS** uses a simple linear bend contour as depicted for an up-interval in **Figure 4-7**. This is by no means the only way to apply the bend over time but, it does provide a fairly ‘musical’ sound for most situations. One could, for example, have both the Prior and New notes track the same pitch (from the prior to the new) and follow something like an S-shaped contour over the **BTime** interval. Any number of such schemes could be devised, but it seems there is a psycho-acoustic phenomenon at work here that leads to a discrepancy between what **should** sound good and what actually **does**. Some early experiments with various bend shapes (that were theoretically promising) produced disappointing results from a musical point of view. On the other hand, the simple scheme that was adopted seems to be quite musical.

4.9 MIDI Control of the Bend Function

The **SLS** provides for MIDI control of both the bend time and bend amount. The *assignment-button* below the **BTime** knob allows you to assign a **MC** to provide **BTime** control and, similarly, the *assignment-button* below the **MinBend** knob allows you to assign a **MC** to provide **MinBend** amount control. The ranges covered by the assigned **MCs** can of course be set to any desired fraction of the knob they control as described in **Section 2.4**.



4.10 Using DFD Mode

As discussed in **section 4.6**, removing the attack portion of the samples used for the ‘inside notes’ of legato phrases often enhances the legato effect. But, unfortunately, a script’s ability to control the sample-start offset **only works when K2 is in Sampler Mode and not in DFD mode**. So if the instrument you are using requires a sample offset to sound right, you will need to operate that instrument’s groups in **Sampler mode**. Conversely, if you need to operate the instrument in **DFD** mode, the crossfade alone will have to try to cover up the attack transient (since the sample-start offset settings will have no effect). Thus, for many instruments, the legato effect may not sound quite as smooth in **DFD** mode as it does in **Sampler** mode.

To help you overcome this disparity, **SIPS 2** has a new feature. If you set the Offset Menu to **DFD Mode**, the **SLS** will play all the ‘inside notes’, of a legato (or portamento) phrase, from the current articulation’s ‘**Inside**’ groups (if they exist). To illustrate how you can utilize this feature, suppose you have a sustain articulation contained in a single group that you have assigned to **Articulation 3-4** in the **SAS**. With K2 in **Sampler Mode**, select **Articulation 3-4** and load (or construct) an appropriate legato preset for this articulation. Set the Offset Menu to **Knob Time** and then adjust the **Leg Ofst** knob until you find the best sounding, fixed offset time for this articulation. Once you have found this offset time in milliseconds, multiply it by the sample rate in KHz to express the offset time in samples. For example, if the desired sample-start offset time is **360ms** and the group’s samples were recorded at **44.1KHz**, the offset in samples will be $360 \times 44.1 = 15,876$ samples. Write this number down somewhere.

Now, make a copy of the group (with samples) and use K2’s zone/loop editor to change the **S.Start:** value to the offset time (in samples) that you wrote down. Unfortunately, you will have to do this for each zone in the group individually (at least until NI finally provides what we need to edit all sample zones at once). Once you have completed these edits, rename the group identifying it as an ‘**Inside**’ type for **Articulation 3-4** (see **Section 3.16**).

Now move to the **SAS** and open the **Setup/Audition Panel**. Set the **Articulation Index** knob to **3-4** and then open the **Assign Groups** menu and click on **Clear Articulation**. This will de-assign both the original and your new **‘Inside’** group (which were both assigned as **Normal** groups when you made the **‘Inside’** clone). Now click on the original group in K2’s group editor display (be sure it’s the only one checked) and then open the **Assign Groups** menu again and select **Normal Groups** in the Fixed Articulation section. Now in K2’s group editor, check only your new **‘Inside’** group and then open the **Assign Groups** menu again and this time click on **‘Inside’ Groups** in the Fixed Articulation section. Now both the original and the new **‘Inside’** group will be assigned to **Articulation 3-4** but the original group will be assigned as **Normal** and the new group will be assigned as **‘Inside’**.

Now, you can change both groups from **Sampler** to **DFD** mode in K2’s Source module. Then, when you set the **SLS**’s Offset Menu to **DFD Mode** and play a legato phrase, the **SLS** will play all the ‘inside notes’ from the **‘Inside’** group and it should now sound the same as it did when you were using the **Sampler** mode with a fixed **Leg Ofst** setting.

The foregoing may seem a little complicated, but apart from the tedium of changing the **S.Start** value, zone by zone, the **SIPS** part is really quite simple once you understand how to use the **SAS**. To soften the zone by zone editing tedium, Nils Liberg has a really neat ‘Robotic Batch Tool’ that can do most of the work for you. You can download it here <http://www.nilsliberg.se/ksp/tools/SampleStartOffset/>. Also keep in mind that the steps to clear and then re-assign the groups for **Articulation 3-4** would not normally be required because generally you will know in advance when you are going to use **DFD** mode and you can setup all the **‘Inside’** groups as you format and configure your instrument in the **SAS**.

4.11 Using Portamento Mode

For **SIPS 2**, the **SLS Mode** menu now has a 4th mode labeled **Portamento**. The **SLS Mode** menu can of course be assigned to a **CC** for remote controlled mode changes. When a **CC** is assigned, its min position will select the **SLS OFF** mode and it’s max position will select **Portamento** mode. Any **CC** position in the lower half (but not at min) will select the **Solo Mode** and any **CC** position in the upper half (but not max) will select **Legato Mode**. Note also, when **Portamento Mode** is selected, the **Leg Ofst** knob becomes re-labeled as the **Port Ofst** knob. This is an independently set, fixed sample-start offset that you set for use only in the **Portamento mode**. When you switch the **SLS Mode** back to **Legato**, the previously set **Leg Ofst** value (and/or mode) will still be in effect.

Now we can begin to discuss the operational modes of the **Portamento** effect. Once the portamento effect is selected by the **SLS Mode** menu, there are 3 parameters of interest; namely, **Port Ofst**, **PTime**, and **PBend**, which will be discussed as we go along. The portamento effect is produced using equal-power crossfading of adjacent pitch samples as you glide or bend through an interval. This basically keeps the timbre ‘formant-corrected’ even over wide intervals (assuming enough multi-samples exist). For similar reasons as for the legato effect, you will usually want to start the ‘inside’ samples after their attack portion. This is the purpose of the **Port Ofst** knob. With the **Port Ofst** properly adjusted (in K2/3’s Sampler mode), the glide effect can usually be made very smooth. The **Port Ofst** knob has only one mode which is the same as the legato’s **Knob Time** mode. This mode provides a fixed offset time interval from 0 to 2000ms as set by the knob. But this setting is independent of any **Leg Ofst** setting you make with this same physical knob when not in the **Portamento Mode**.

While **Port Ofst** and **Leg Ofst** can be set independently to two different values, remember that these offsets only work in **Sampler** mode. If you intend to use **DFD** mode, you should try to find a common value for both offsets that will give acceptable performance. When the Offset menu is set to **DFD Mode**, the **SLS** will play all ‘inside samples’ from the articulation’s **‘Inside’** group (thus the same group will be used regardless of whether you are in **Legato** or **Portamento** mode). Therefore, you should find a common offset value that you can use when you prepare the **‘Inside’** clone group (as described in **section 4.10**).

The portamento effect has two major modes which mimic that of a **Glider** and **Bender** respectively. To explore all the facets of these functions, first assign a **MC** using the **PBend** menu. Henceforth, this **MC** will be referred to as the portamento bender, or just the 'bender'. **CC25** is recommended for this assignment because it will be required if you want to use the **Super Bender** script (which is described in **section 4.14**). Set the **Port Ofst** and **PTime** knobs to their default values of **250ms** and **1000ms** respectively and, for the following experiments, select a nice sustain articulation, preferably one with looped samples (or at least one with some very long sustains). First we'll discuss the standard **Glider** mode. Select the **Portamento mode** and then set the **bender at maximum**.

Now, play any note and while still holding down the key, play another note (just as you would if you were going to play a legato interval). When you do this, the first note will 'glide' to the second note. If you then release the second note, it will glide back to the first note (still being held). Now try this. Hit and hold the first note followed by an overlapping second note. After the glide, this time release the first note while still holding down the second note. Notice that the 2nd note continues to sustain. Now, while still holding down the 2nd note, hit another key and the 2nd note will glide into the new 3rd note just played. Play around with this for a while until you get the hang of it. Using this technique you can continue to glide between any number of successive intervals. As long as you leave one key down, you can continue to glide to the next note played. Once all keys are released, the next note played will then start a new phrase.

You can of course change the **PTime** setting for different glide speeds. In the **Glider** mode, **PTime** directly controls the **Glide** time so that the shorter the **PTime** setting, the faster the glide will occur. You can also assign a **MC** to **PTime** and then change the glide time under MIDI control. In fact, by changing **PTime** *during* a glide you can even alter the pitch-time-contour of the glide. However, for really precise control of the pitch-time contour, rather than using the **Glider Mode** you will want to use the **Bender Mode** (to be discussed shortly).

Now try this, play C3 and while still holding it down, play an overlapping G3. The C3 will sound and then (after you hit G3) the C3 will glide up to G3. Now, while still holding down both C3 and G3, add A3. Notice that the G3 continues to sound and that the playing of A3 has no effect. This is because the portamento mode deals with only two notes at a time. The first note of a phrase establishes the 'root' pitch whereas the second (overlapping) note establishes the 'target' pitch for a glide or bend. But, until at least one of the first two held notes are released, any additional notes played are simply ignored. When one of the first two notes are released, the **SLS** glides to the remaining note's pitch (which could be no change if the pitch is already that of the remaining note).

Now let's explore the **Bender Mode**. If you set the bender **MC** to its **minimum position**, when you play an overlapping interval, only the first note continues to sound. However, if you keep both keys held down, and then you slowly move the **bender** from min to max, the pitch will bend from that of the first note to that of the 2nd note. In the **Bender Mode**, the 2nd key played sets the maximum bend interval but, you can control the pitch-time-contour with the bender **MC**. The bend can be made as fast or slow as you want and with any kind of curvature desired (just listen as you bend and you can interact). So, as in the **Glider Mode**, the second note held down determines the 'target' pitch (but you have to move the **bender** to maximum to reach it). While both keys are held down, the **bender** controls the pitch. However, as soon as you release one of the keys, the pitch will 'glide' to that of the remaining key held and, **the bender will be put in 'neutral'**.

To illustrate this, try the following. Release all keys and set the **bender to min**. Play and hold C3 and move the bender to max and back to min. Notice that the pitch that sounds is simply C3 and that the bender has no effect. Now, with the bender at minimum, continue to hold C3 but also add G3. Because the bender is at minimum, the pitch will still stay at C3. Now, raise the bender slowly to maximum and the pitch will move upward to G3. When the pitch reaches G3, release the C3 (keeping G3 held down). You can now move the bender to minimum and the pitch will stay at G3. Now, with the bender at minimum, play C4 (while keeping G3 held down). G3 will continue to sound but when you move the bender from min to max, the pitch will move upward again from G3 to C4.

This effectively gives you the means to bend from C3 to G3, stay there a while and then continue to bend from G3 to C4. Of course you could do the same thing by playing C3 and C4 to begin with and then only moving the bender part way until G3 is reached. Then wait a while and continue on from G3 to C4. However, if you do it that way, you have to find the G3 plateau by ear and the **MC** used for the bender will not have as much resolution because the **MC**'s 0 to 127 steps have to cover the whole octave from C3 to C4. Doing it in stages, gives you nice, well-defined interim pitches and more controller resolution.

Now try this. With all keys released and the **bender at min**, play and hold C3. Then, move the bender about halfway to max and then play G3. The pitch will glide from C3 to about E3 (depending on just where you have positioned the bender). At this point if you release the C3, the pitch will glide to G3 which then becomes the new root key. The bender is again in neutral and may be positioned anywhere without effect. Conversely, if you repeat the foregoing but instead of releasing C3 you release G3, the pitch will glide back from E3 to the original root key of C3.

We can summarize the glider/bender behavior with a few simple rules as follows:

1. When no notes are sounding, the first key of a new phrase becomes the 'root' pitch which will be the same as the played key's pitch (regardless of the bender's position).
2. Whenever only one note is held, the bender is in **neutral** and can be moved without affecting pitch.
3. When a second overlapping key is depressed, the pitch will glide to a 'target' pitch determined by the bender's position and the interval between the two notes. For example, if the bender is at 50%, the pitch will glide to halfway between the two notes. If the bender is at 100%, the pitch will glide to that of the 2nd note while if the bender is at 0%, the pitch will not change (ie it will stay at the first note's pitch).
4. While both notes are held, the bender will be able to slide the pitch anywhere between the two held notes.
5. When either of the two held notes are released, the pitch will glide (from wherever it is) to the pitch of the remaining note held.

Note that you can bend or glide over as wide an interval as you wish without the dreaded chipmunk effect because all available multisamples are used and thus the timbre is effectively formant-corrected. Also note that when using the bender to change the pitch, **PTime** acts as a smoothing filter time-constant. If you move the **bender** fast, the pitch may not track your movement if the **PTime** setting is too high. Like most smoothing filters the trade off is between smoothness and response time. So, in the **Bender Mode**, you normally will want to set **PTime** low enough to 'track' your moves but no lower.

If you only want to use the **Glider** (and not the **Bender** function), you need not assign any **MC** as a bender. If no **MC** is assigned (via the **PBend** menu), the **SLS** will automatically setup an internal 'virtual bender' and set it to max. With the bender (virtual or otherwise) set to max, **rule #3** above dictates that each time you add a 2nd note to one note being held, the pitch will glide to that of the 2nd note. However, if you have a bender assigned, there is an **alternate method** that can be used to glide.

If you set the **bender to min** and you play two overlapping notes, the pitch will remain at that of the first note (again because of **rule #3**). However, if you now release the first played note, the pitch will glide to that of the 2nd note because of **rule #5**. Therefore, with the **bender at its minimum position**, you can glide to the second note by **releasing the first note** whereas with the **bender at max**, you can glide to the 2nd note when it is played. Therefore, you can glide with the bender set to either min or max but since the 'bender at max' mode is apt to be more familiar than the 'bender at min' mode, the **SLS** sets its 'virtual bender' to max rather than to min. However, if you assign an actual bender, you will be able to take advantage of whichever glide mode is most convenient for the situation at hand. In fact, the alternate Glider method is exploited by the **Super Bender Script** (see **section 4.14**).

When changing the **SLS** mode from **Legato** to **Portamento** or vice versa, mode carryover has been implemented in a similar manner to that of changing from **SLS OFF** to **Legato Mode** or vice versa (see **section 4.4**). If you are playing a legato phrase, and while you are holding the last note (or if you activate the sustain pedal) and then switch to the **Portamento** mode, the last note of the legato phrase will ‘carryover’ and become the first note of the glide (or bend). Once in the **Portamento** mode, **the sustain pedal will only sustain the current root pitch**. For example, if you are ‘between phrases’ in the standard glider mode (bender at max), hit the C3 key with your foot on the sustain pedal. Once the pedal is down, you can release the C3 key and the ‘root’ pitch will continue to sound. Now if you hit the G3 key, the pitch will glide from C3 to G3 (even though only the G3 key is held down). If you then release the G3 key, the pitch will glide back to C3 (as long as you keep the sustain pedal down). However, if you sustain the C3, play G3, and then release the sustain pedal (while the G3 key is still down), the G3 becomes the new root key so if you again depress the sustain pedal, you can release the G3 key and it will continue to sound.

Now, suppose that you are in the Bender Mode with the **bender at min**. Play C3 and then depress the sustain pedal. Now when you release the C3 key, the root pitch will of course continue to sound (while the pedal remains down). Now hit and hold down the G3 key (keeping the pedal down also). The C3 continues to sound. Now raise the bender from min to about halfway to max. The pitch will then rise from the root pitch to about E3. If you then release the sustain pedal, the pitch will glide to G3. Alternatively, if while holding G3 and the pedal down, (while sounding E3 or thereabouts), you switch the **SLS Mode** to **Legato**, the sustained E3 pitch will continue to sustain but it will now become the current note of a legato phrase. Note however, that the E3 (or thereabouts) that was sounding as the last portamento pitch could be ‘in the cracks’. Whereas, in legato mode, all pitches are bound to the chromatic scale and there are no pitches between D# and E nor between E and F. So, how does the **SLS** handle it when you slide to somewhere in the cracks and then switch the mode to legato? The answer is that the **SLS** computes and then moves to the nearest chromatic pitch from where you leave off in the portamento mode.

Probably one of the most common things you will want to do with the **Portamento Mode** will be to play a legato phrase where two of the notes are connected by a slow glissando rather than a more rapid legato bend. This sort of thing can be done quite easily. For example, suppose you want to play a legato phrase consisting of the notes C-D-E-F-G and you want to make the E to F transition a glide. With **SIPS** in **Legato Mode**, begin to play the 5-note phrase. After you reach E, switch to the portamento mode and after you glide to the F, switch back to legato mode. In Glider Mode, the E-F transition will simply glide whereas in Bender Mode, you can completely control the glissando from E to F. All the other transitions will of course be done as normal legato intervals.

While the procedure just described for inserting portamento glides or bends within a legato phrase is conceptually easy, it does involve the manipulation of at least two and possibly three or four **MCs**. Of course if you are driving K2/3 from a sequencer this is not too difficult to arrange. However, for live playing or to enable you to play such legato phrases with a single pass, you might find yourself wishing you had more than two hands. However, **SIPS 2** has a new ‘front-end’ script called the **Super Bender Script**. This script can be used in place of the simple **Starter Script** and it will enable you to control most of the portamento and legato bending functions using only your **Pitch Wheel**. The **Super Bender** operation is described in **Section 4.14** and the **SLS** configuration needed to work with it is described in **Section 4.15**.

The best way to familiarize yourself with how the portamento mode functions under various conditions is to ‘play around’ with it doing things similar to the illustrations already presented. There are a lot of combinations, especially if you switch in and out of the portamento mode to/from the other modes (including **SLS OFF**) and especially if you involve the sustain pedal. Once you get the ‘hang of it’, you’ll find that it’s all quite logical and consistently predictable (I hope ;-)).

4.12 Release Sample Triggering

For **SIPS 2**, the **SLS** (in conjunction with the **SAS**) now provides release sample triggering (see also **section 3.15** and **3.30**). The Legato Effect is generally more convincing for instruments that are sampled fairly dry because this allows the needed reverb to be added **after** the legato effect is produced, thus avoiding any ‘mangling’ of the reverb during the crossfade interval. For dryer libraries, release samples (if they exist) only need to be played at the end of a legato phrase in order to provide a more realistic closure of the sound (rather than to provide a reverb tail). Therefore, if release groups exist for the current articulation, the **SLS** triggers the release samples only at the end of each phrase while in **Legato**, **Portamento**, or **Solo Mode**.

As already explained in **section 4.4**, when the **SLS** is disabled using **SLS OFF**, all played notes are simply passed through the script without processing (except for the sustain pedal function). However, if the current articulation contains release samples, **they will be triggered** at the end of each note. Therefore, release samples, if they exist, are properly triggered for all 4 modes of the **SLS**.

Release samples vary in length, so in order to be sure the entire sample is played through completely, **SIPS** triggers release samples to essentially run forever. This means that release samples will play until the sample itself runs out, at which point the note event will be retired (this is basically the same way **K2** handles release sample triggering). This works out well for almost all release samples. However, you might happen to have some release samples that are looped and simply use an envelope to decay them to silence. Such samples, since they are looped, will essentially run forever, silently but nonetheless consuming resources. If you have such a situation, **SIPS 2** has a **User Preference** option switch (**Sw4**) that can be set to limit the playback time of release samples to 3 secs. This should be long enough for most release samples yet it will prevent running looped samples forever.

4.13 Guidelines for Making Legato Settings

Coming up with the right settings for a great-sounding legato preset is not unlike programming a synthesizer. By studying what the various synthesizer controls do and how they interact you can then experiment with settings and learn by listening to the results. Synthesizers usually come with a number of factory presets and it’s often easier to pull up a preset for an instrument or sound that’s similar to the one you are trying to develop and then try to intelligently tweak it.

Similarly, when setting up the **SLS** for use with a new instrument patch, the easiest way to begin is to pull up a preset for an instrument that has similar characteristics and then tweak the settings for the best sound. However, in order to do this, you must have a good grasp of what each control does and how they interact. So obviously you should first read the various technical discussions presented in this **User’s Guide** to gain a good understanding of the various parameters from a theoretical point of view.

While tweaking an existing preset may seem like it will provide a fairly quick way to bring up a new instrument, you’ll probably find this to be the case only after you have acquired some ‘hard won’ experience with the process. In order to get some of this experience, it is suggested that you try to bring up a few instruments from scratch using a procedure similar to the following.

NOTE: be sure to set the Instrument Range before you begin. Throughout the legato preset design, disable the **SVS** by setting its mode menu to **SVS OFF**. This is to make sure that you aren’t being confused by any vibrato effect being added. Next, set up the **SLS** parameters as follows. Turn off the bend function by setting **Bend** = 0. Then set the release mode to **Knob Setting** and set **RlsFade** = 100%. Next set the offset mode to **Knob Time** and set the **Offset** knob to 0 ms. Set **AtkFade** = 100% (remember **NodeVol** is ignored when **AtkFade** = 100%). Finally, set **XTime** to about 100 ms as a starting point. During the course of designing your preset, you will want to experiment with the **Leg Ofst** parameter and since that only functions in **K2**’s **Sampler** mode, you should develop your preset initially in **Sampler** mode (even if you intend to later use this preset in **DFD Mode**).

Now, before changing any settings, disable the legato effect entirely (use the **SLS Mode Menu** in the lower right-hand corner of the control panel) and select **Solo Mode**. Next play a few legato phrases (ie play with overlapping notes or keep the sustain pedal down) to get familiar with how the patch sounds when playing with just the **Solo Mode** active. When you play a new note in **Solo Mode**, any prior note still held receives a note-off command causing it to proceed to its envelope's release phase. Thus the notes are packed tightly together to the extent that the prior note's release occurs during the attack of the new note. In particular listen to how sharp an attack the patch was sampled with.

Now, enable the legato function (using the **SLS Mode Menu**) and play a legato phrase. Listen to how well or how poorly the crossfade softens the note transitions compared to the **Solo Mode**. If the instrument in **Solo Mode** has a fairly sharp attack, you can usually set **XTime** rather low (50 to 150ms or so). On the other hand, if the attack is rather slow in **Solo Mode**, **XTime** may have to be set somewhat higher. However, resist the urge to fix everything by raising **XTime** excessively. High values of **XTime** bring with it all sorts of bad side effects that have to be dealt with. One of the first things to try before raising **XTime**, is to raise the **Leg Ofst** time to see if getting rid of the attack portion of the 'inside' notes helps. But, don't raise **Leg Ofst** any higher than needed either. Rather use just enough of both **XTime** and **Leg Ofst** but don't overdo it.

Also, it can't be emphasized enough that setting **XTime** too high and then trying to compensate with the other parameters is a prescription for a 'less than stellar legato preset. You can always raise **XTime** a little as needed during your final rounds of tweaking. Now, once you have established minimal starting points for **XTime** and **Leg Ofst** (that seem to provide fairly smooth legato transistions,) it's probably a good time to introduce the bending effect. For a starting point, set **BTime** to the same value as **XTime** and then set **Octave*** = **1.0** and **MinBend** = 20 cents.

Now play some semitone intervals and listen to the bend effect. Raise or lower the **MinBend** value to provide the right amount of bend (for when you play a semitone interval). Now play some octaves (legato of course) and slowly raise the value of **Octave*** until the amount of bend sounds good for when you play an octave interval. Then, start listening to how wide legato intervals sound (5ths to Octaves) and listen for sounds of harmony or chorusing during the crossfade interval. If you have set **XTime** fairly low, it's unlikely you will hear anything like this but it's not uncommon at higher **XTime** settings. If you hear some undesirable effect along these lines, try lowering the value of **RlsFade**. You may need to set the value as low as 10% (but don't set it any lower than necessary) to subdue any chorusing or harmony artifacts, especially when you play wider intervals.

Now as you play lots of legato phrases, both slow and fast, listen for problems. For example if everything sounds fine for slow passages but fast passages sound a little sluggish (usually when you have **XTime** fairly high), you may have to introduce a 2-segment fade-in contour. Another reason you may have to do this is if you seem to get a little volume dip between notes (again this usually occurs only for high **XTime** values). If you feel you need to introduce a 2-segment fade-in contour, begin with about 50% for both **AtkFade** and **NodeVol**. Then slowly lower **AtkFade** relative to **NodeVol** (ie lower **AtkFade** or raise **NodeVol**). **Generally you will want to avoid settings where NodeVol is set lower than AtkFade.**

You may also find that once you introduce a 2-segment fade-in contour, you will be able to raise **XTime** a little without as much ill-effect as there was with a 1-segment fade-in contour. Since all 3 of these controls interact quite a bit, try changing all of them iteratively in small amounts rather than changing one a lot. In the end your ears will have to be your guide but you should also have a good feel for what to expect when you change something. As already discussed in the section titled '**Understanding the Crossfade Contouring Controls**' there are a lot of **nonsensical** settings for these controls which should be avoided.

On the bend function side of things listen to not only the bend amount but the bend rate. If you think a faster rate would sound better, first try lowering **BTime** a little rather than raising **MinBend**. On the other hand if you think a slower rate would sound better try the opposite. Of course you may now want to tweak the **MinBend** and/or **Octave*** setting. Also, somewhere along the line you might want to assign the **Pitch Wheel** (or some other **MC**) for MIDI control of **MinBend**. Then as you play legato phrases you can increase the bend effect as you play when it seems musically appropriate. Initially try this with the **MC** range set to the full range of the **MinBend** knob. Then, try narrowing the control range such that you are using most of the range of the **MC** but never running out of headroom. If you are running out of **MC** headroom, increase the range and if you are using too small a fraction of the **MC**'s range, decrease the amount of knob range covered by the **MC**.

When you think you have the settings pretty close to optimum, you might want to listen to the effect of using something other than the fixed, **Knob Time** offset mode. The **Knob + Rand** mode can sometimes help reduce the legato-form of the machine-gun problem. However, if you intend to use this preset in **DFD** mode, you will have to settle for a fixed offset time determined in the **Knob Time** mode. Remember that to eventually run in **DFD** mode, you will need to create a clone group of edited samples and having a fixed offset time will be the most convenient for that scenario (for details review **section 4.10**).

Depending on your playing style, you might also want to try the **Key-Lift** release mode. Of course you can also assign **MCs** for **XTime** and **BTime** so that more things can be varied in real time. Obviously you may be limited as to how many **MCs** you can manipulate at one time but, if you are recording to a sequencer, you can always make multiple passes if necessary. **The more real time controls you can use artfully, the more realistic and convincing your legato passages will be compared with a ‘set it and forget it’ approach.** After you get the hang of it, you'll no doubt settle down to your own style of making legato settings but the plan just outlined will serve as a good introduction to the basics.

4.14 The Super Bender Script

The **Super Bender Script**, hereinafter called the **SBS**, can be installed in slot 1 (in place of the basic **Starter Script**). While the **SBS** also provides the startup function, it's main purpose is to perform pre-processing of your **Pitch Wheel** (hereafter referred to with the abbreviation **PW**). With the **SBS** installed (and the **SLS** properly configured as discussed in **section 4.15**), you will be able to control a number of **SLS** bending functions and modes using only your **PW**. You will be able to use the **PW** to vary legato bending amounts when in **Legato Mode** and as the portamento bender when in **Portamento Mode**. You will also be able to conveniently use the **PW** to switch back and forth between **Legato** and **Portamento** modes as well as to control **PTime** (to vary glide time). Thus, with the **SBS** installed, your **PW** can perform most of the functions that would ordinarily require manipulation of up to four separate MIDI Controllers.

The **SBS**'s panel contains a **Pitch Wheel Mode** drop-down menu with which you can select one of three modes for the **PW**. In the **Normal Bender** mode, the pitch wheel functions ‘normally’ in that it merely bends the instrument's pitch (either up or down) with the range in semitones specified by the **Edit Box** to the right of the menu button. Bending is accomplished conventionally (without any formant-correction) by simply controlling K2/3's instrument tuning (see **section 4.15**). The two remaining menu choices are **Super Bender** and **Controller Only**. In the **Controller Only** mode, the **PW** is simply passed through the script without any interdiction by the script.

Of course, the **Super Bender** mode is what we want to focus on here. Ordinarily in **SIPS**, the **PW** works the same when moved in either direction away from its center position. However, in **Super Bender** mode, the **PW** functions differently in its negative region than in its positive region. By negative region, we are referring to moving the **PW** in the direction that would conventionally produce a lowering of pitch.

Each time the **PW** is moved into its negative region (from its rest or positive position), the **SLS** mode will be toggled between **Legato** and **Portamento** mode. For convenience, the current mode of the **PW** is displayed to the right of the **PW Mode** menu button as either **Legato Bend** or **Portamento Bend**. In the **Legato Bend** mode, the positive region of the **PW** controls the amount of Legato Bend (ie the Min Bend knob). In the **Portamento Bend** mode, the positive region of the **PW** functions as the portamento bender **MC**. In addition, when the **PW** is in its negative region, its position can control the **PTime** knob value.

A few examples should clarify how you can use the **PW** in **Super Bender** mode. While in the normal **Legato Bend** mode, the **PW** can be used in its positive region to increase the **Min Bend** used by the **SLS**. In other words this functions the same way it would without the **SBS** if you simply assigned the **PW** with the **LBend** menu, **except that**, you can only use the positive half of the **PW** (and not the negative half as would be the case without the **SBS**). So as long as you keep the **PW** out of its negative region (and the **SLS** is still in legato mode), the **PW** can serve as the **Min Bend** controller.

Now, suppose that you want to play a legato phrase consisting of the scale run C3, D3, E3, F3, G3. and that you want to glide between D3 and E3. Starting in legato mode, you would play the C3 followed by the D3 (overlapping of course). Then, once you have played the D3 and are holding the key down, wiggle your **PW** by moving it down slightly (and then let it spring back to its center rest position). Because of **SIPS** mode carryover, the D3 will still be sounding but the **SLS** mode will have changed from legato to portamento. Next play the E3 (while still holding down D3) and then, when you want to glide to the E3, **release** the D3 key. Since the **PW** (which is now functioning as the **portamento bender**) is at its zero position, the glide from D3 to E3 is accomplished with the alternate (or key-release) gliding method discussed in **section 4.14**. Once you have released the D3 key, you can again wiggle the **PW** negatively which will toggle the **SLS** back to legato mode. The E3 will still continue to sound and you can now play an overlapping F3 followed by an overlapping G3 to finish the legato phrase.

The glide between D3 and E3 in the foregoing example will use the **highest PTime** value programmed (when you configure the **SLS**, see section 4.15). If you want to glide faster, instead of wiggling the **PW** negatively and back to zero, you can instead hold the **PW** in the negative region. While the **SLS** is in portamento mode, the negative region of the **PW** will control the **PTime** value inversely. That is, **PTime** will get smaller the farther you pull the **PW** down. So, for example, after you play the D3, instead of wiggling the **PW**, pull it to its maximum negative position and then play the E3. Now, when you release the D3 key, the D3 will glide to E3 using the **lowest PTime** value programmed for the **SLS** (again see section 4.15). If you are trying to lower **PTime** only slightly from its highest limit, you will have to be careful that you don't accidentally wiggle the **PW** back through its zero position and then back into the negative region. Doing so will cause the **SBS** to toggle the **SLS** mode back to the legato mode. It will of course take some practice to avoid things like this.

When the **SLS** is in its Portamento Mode, the positive region of the **PW** functions as the portamento bender. So, if instead of gliding, if you would like to 'slide' from D3 to E3 (using the bender mode), after you play the D3 and wiggle the **PW** (negatively and back to center), play the E3 key but this time keep both the D3 and E3 keys down. Then raise the **PW** into its positive region and you can use it to slide the pitch from D3 to E3 with any kind of pitch-time contour you desire. Once you reach the destination E3 pitch, you can release the D3 key and then move the **PW** without affecting the pitch. So, back it off and again wiggle it slightly negative (before returning it to its center rest position). This will cause the **SLS** to toggle back to legato mode so you can finish the legato scale phrase.

When you setup the **SLS**, you will assign a set of CCs to control the **SLS Mode** and the various related legato and portamento parameters. The **SBS** merely translates your **PW** movements into the corresponding set of CC messages needed to effect the desired control. However, you can still use the basic CCs that are assigned to the **SLS** to control these parameters in the usual way. Therefore, you can view the **SBS** as simply providing an alternate and hopefully more convenient way to control the primary functions of your assigned controllers.

4.15 Configuring the SLS for use with the Super Bender

When you setup the **SLS**, you can assign any desired set of CCs to control the various functions associated with legato and portamento bending. However, to take advantage of the services provided by the **SBS**, it is necessary that you assign certain specific CCs (unless you are also willing to edit and recompile the source code for the **SBS**). The **SBS** translates **PW** movements into a corresponding set of CC messages that will effect the desired control of the **SLS**. In order to do that, the **SBS** makes certain assumptions about which CCs are assigned to the needed functions. Therefore, in order for the **SBS** to work, you must configure the **SLS** properly.

Before assigning CCs to the **SLS** parameters, you first need to prepare your Instrument. If your instrument uses the **PW** as a modulator of anything, you will need to disassociate it from those functions. For example, many instruments have the **PW** assigned in each group's source module as a tuning modulator. This is of course to provide conventional **PW** behavior. However, when using the **SBS**, such **PW** modulations must be defeated. Even when using the **Normal Bender** mode of the **SBS**, the script controls the overall instrument pitch (via an engine parameter) rather than via modulation of the individual groups. This is done by the **SBS** in such a way that it will still work properly even if you set the Instrument Tuning knob to some position other than 0.

In the **SLS**, assign **CC21** to the **SLS Mode** menu and also assign **CC25** to both the **LBend** and **PBend** functions. The same CC can be used for both these functions because they are mutually exclusive. Therefore, when you switch the **SLS** to **Legato Mode**, you can use **CC25** to control the **Min Bend** knob (and its control of the **portamento bender** will have no effect). Similarly, when you switch the **SLS** into **Portamento Mode**, you can use **CC25** as the **portamento bender** (and its control of **Min Bend** will have no effect). After you have assigned **CC25** via the **LBend** menu, use the procedure described in **section 2.4** to set the lower and upper limits for **CC25**'s control of the **Min Bend** parameter. These same range limits will be in effect if you use the **Super Bender PW** for legato bend control.

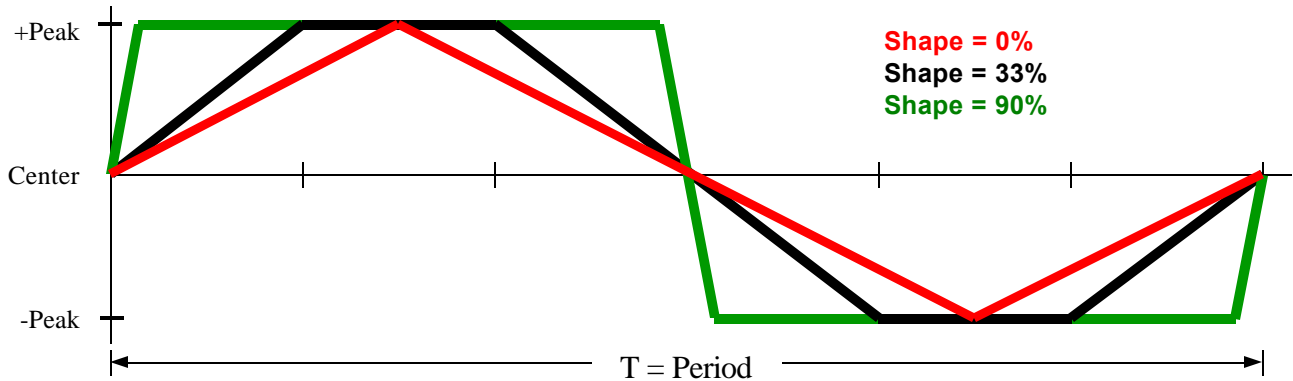
Next assign **CC23** to control **PTime** and then set the range limits for **inverse control** (as also described in **section 2.4**). If you will be using the portamento bender mode, you should set the max range value of **PTime** (which will occur when **CC23** is minimum) to the highest value that will still allow the bender to track the fastest bends you intend to make. This will then also be the slowest glide time you will be able to achieve when gliding. Finally, set the minimum range value of **PTime** (produced when **CC23** is at maximum) for the shortest glide time you want to use.

It is important to realize, that you can now use **CC21**, **CC23**, and **CC25** in the normal way if desired. However, because you have assigned these specific CCs to these specific functions, you will also be able to use the **SBS** to control these parameters using only the **PW** if desired. Of course the **PW** (via the **SBS**) can only toggle the **SLS Mode** between Legato and Portamento mode. So, if you want to change to either **Solo Mode** or **SLS-OFF**, you will need to use **CC21** directly.

5.0 SLPS Vibrato Script

5.1 Introduction

The **Vibrato** effect modulates both the pitch and volume of a held note. The **SVS** allows you to set the maximum amount of pitch and volume modulation independently. In addition, the modulation rate and its basic waveshape can be controlled. The basic modulation is trapezoidal but the ‘flat-top’ duty cycle can be altered to provide waveshapes ranging from triangular to nearly square as depicted in **Figure 7**.



Vibrato Modulation Waveform Shapes Available

Figure 5-1

The maximum amount of pitch and volume deviation is set by the **Vibrato** and **Tremolo** panel knobs. The frequency of these modulations and the waveform shape is set by the **Speed** and **Shape** knobs respectively. To these four parameters, various kinds of random variations can be applied over time using the **Random Variation** edit boxes named **Vibrato**, **Tremolo**, **Shape Drift**, and **Speed Drift**. The combination of all these modulations and variations produce the ‘**Vibrato Effect**’ and the overall intensity of this effect can then be controlled over time in several different ways.

With the **SVS Mode Menu** set to **Amt Knob Only**, the overall intensity of the ‘**Vibrato Effect**’ (from 0 to 100%) is determined by the (MIDI-Controllable) knob named **Efx Amt**. In this mode you can use the assigned **MC** to vary the amount of the ‘**Vibrato Effect**’ over time in any way you desire. You can also vary the **relative** proportion of pitch and volume modulation with the MIDI-Controllable **Balance** knob.

5.2 Using the Vibrato Envelope

With the **SVS Mode** set to **Knob + Env**, you can also superimpose a time varying **Envelope** on the **Efx Amt** setting. First consider how the envelope operates when the **Efx Amt** knob is set to **100%**. The **Envelope** has four parameters which can be set to control the effect amount over time. The **Onset** edit box can be used to set an onset delay (in vibrato cycles) from when the note starts until the effect is brought in. For example, if vibrato **Speed** is set to **4.0 Hz**, setting **Onset** = 8 will produce an **Onset** delay of 2 seconds (8 x 1/4). The **Rise** time edit box can be used to set the time it takes (after the onset delay) for the vibrato to ramp up to max. The units for **Rise** time, like **Onset** delay are vibrato cycles. Once the vibrato is fully established (ie the **Onset** delay has expired and the **Rise** time has passed), the vibrato intensity can then be reduced slowly over time to simulate the ‘relaxation’ of vibrato often used by performers when very long notes are held. The **Decay** and **Sustain** parameters are used to set this ‘relaxation’ effect. The **Sustain** edit box sets the ‘final level’ (as a percentage of max) that the vibrato intensity will decay toward, and, the **Decay** edit box sets the **time** it takes for the decay to occur (in vibrato cycles).

5.3 Using Both the Efx Amt Knob and the Envelope

The foregoing describes the effect of the **Envelope** only (since the **Efx Amt** knob was set to 100%). If you want to use both the **Envelope** and the **Efx Amt** knob to control the effect intensity, keep in mind that both these are *attenuation* controllers and that they are effectively cascaded. Thus, if **VI** is the maximum intensity of the effect, then the intensity heard is given by:

$$(1) \text{ Intensity Heard} = \text{VI} * (\text{Efx-Amt} / 100) * (\text{Env} / \text{EnvMax})$$

Note that if **Efx-Amt** is minimum, ie its value is **0%**, then no vibrato effect will be heard regardless of what value the envelope generator assumes. Similarly, if the envelope generator value is zero, then no effect will be heard regardless of the setting of **Efx-Amt**. Thus, the maximum effect is 100% and the **Efx-Amt** knob can only reduce that. Similarly, the **Envelope** can only reduce the intensity of the effect. So, if the **Efx-Amt** knob is set to 75% while the **Envelope** is also passing only 75%, the overall effect will be reduced to about 56% (0.75 x 0.75). With either of the two controls passing 0%, the other control can have no effect. In other words, the level being passed by one cannot be raised by the other (only lowered farther).

5.4 Humanizing the Vibrato Effect

To add more realism to the vibrato effect, several humanizing functions provide pseudo-random changes to the parameters. You can add a small amount of random deviation to either the **Vibrato** or **Tremolo** (or both independently) using the corresponding **Random Variation** edit boxes. These edit boxes allow you to specify a percentage limit for the variations. Also, the vibrato **Speed** can be varied with a small, random ‘drift’ component to simulate the human fatigue and imperfection factor (using the **Speed Drift** edit box). For even more realism (for non-zero **Shape** settings) you can apply a random drift to the vibrato ‘waveshape’ using the **Shape Drift** edit box.

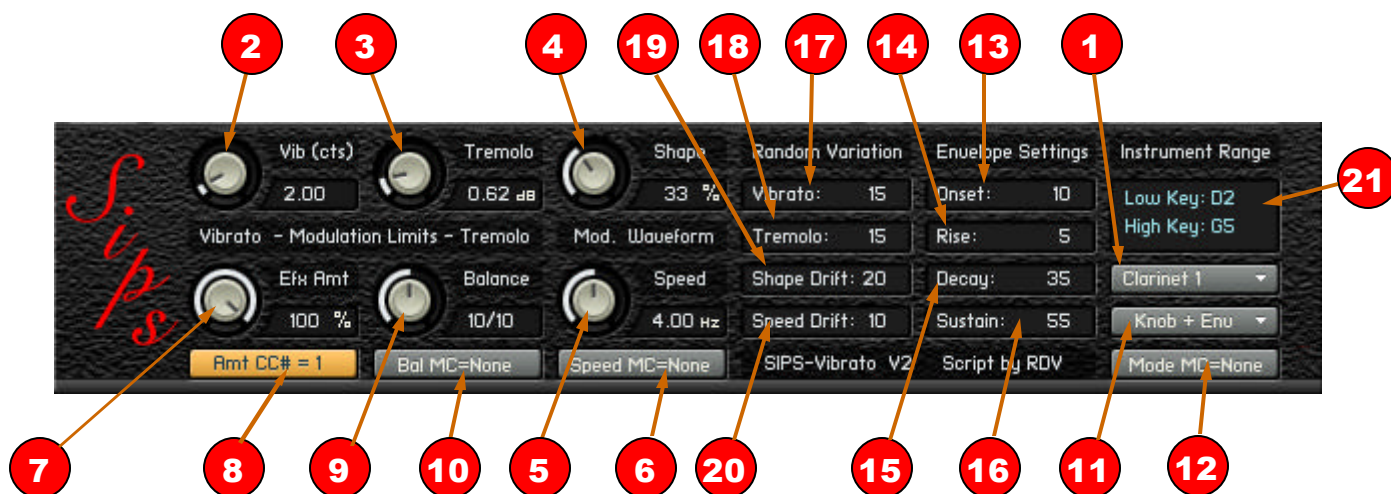
5.5 Setting Random Drift

The **Shape Drift** and **Speed Drift** edit boxes allow you to add a special kind of random variation to the waveform **Shape** and **Speed** of the vibrato modulation. Unlike the random variation made to **Vibrato** or **Tremolo** (which are always made from the knob value), **Drift** changes are made from the last value (ie the knob value plus the accumulated random changes). Since there will be on average an equal number of ups as there will be downs, you might think that over the long haul, the ‘average’ value of such an accumulated drift would remain near its knob setting. However, it is the nature of random numbers that periodic ‘streaks’ of more ups than downs (and vice versa) will occur over time. When this happens, there will be an accumulated ‘drift’ from the center value. The **Drift** edit boxes set an upper limit on this **accumulated drift** that will be allowed (so that **Shape** or **Speed** doesn’t ‘drift’ too far from nominal). While the edit boxes specify the maximum total drift allowed, the maximum amount of change in a single cycle is limited to one-half of the drift setting. This kind of drift away from the center value of vibrato waveform shape and speed is known mathematically as a ‘random walk’. It is also closer to what happens to these parameters in the real world and therefore lends more realism than a straight randomization ‘from the center’.

While random drift can be applied to either the **Shape** or **Speed** of the waveform, it is important to realize that if you set **Shape = 0**, **no variation will be made** because any percentage change of zero is still zero. So, if you want to apply some drift to the waveshape, **you must set the Shape value above zero**.

5.6 Control Panel Layout

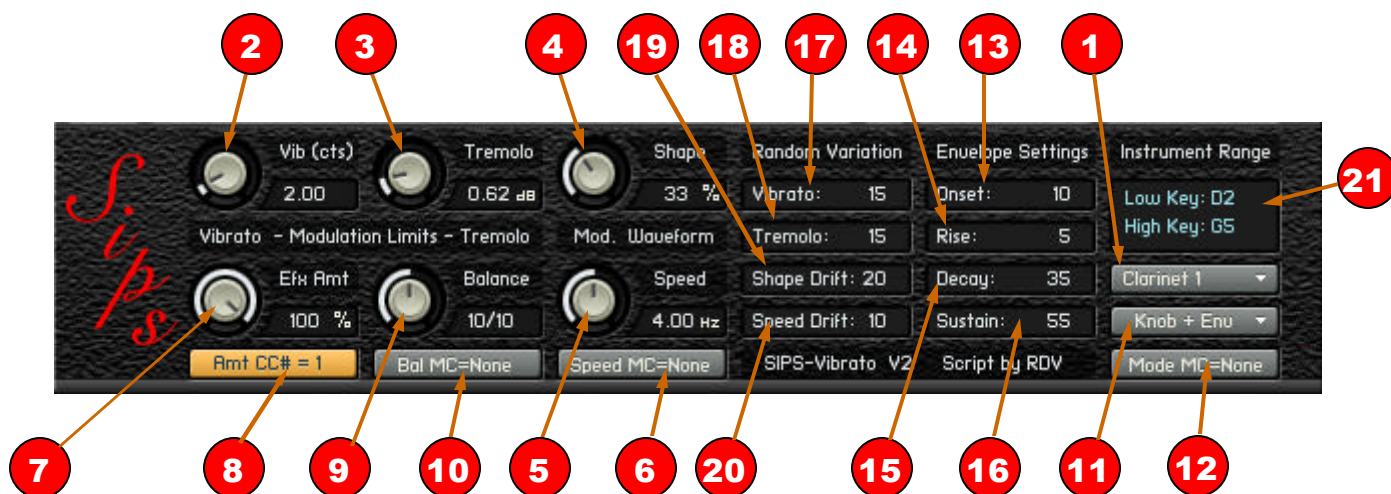
The Control Panel layout and legend for the **SVS** is shown in **Figure 5-2** (at the top of the next 2 pages).



The Vibrato Script Control Panel

Figure 5-2

- (1) **Preset Selector** drop-down menu. Selecting an Instrument or Instrument Class from this menu will set certain key parameters to a good starting point for you to further customize by ear. You can also store and recall your own **User Presets** here. Plus, you can also access other useful functions via this Menu. (including **Import/Export** and **Rename**).
- (2) **Vibrato** width knob. Sets the maximum amount of **pitch** modulation (in cents).
NOTE: pitch modulation is always synchronous with any volume modulation set with (3).
- (3) **Tremolo** depth knob. Sets the maximum amount of **volume** modulation (in db).
NOTE: volume modulation is always synchronous with any pitch modulation set with (2).
- (4) Modulation waveform **Shape** knob. Sets the width of the flat top and bottom of the trapezoidal waveform used for the Vibrato modulation. With **Shape** set to 33%, the waveshape is that of an equilateral trapezoid (see the Black curve in **Figure 5-1**)
- (5) Modulation waveform **Speed** knob. Sets the frequency of the pitch and volume modulation in Hz.
- (6) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **Speed**. You can also set it to control only a subrange of **Speed** if desired (see **section 2.4**).
- (7) **Efx Amt**, Effect Amount knob. When (11) is set to **Amt Knob Only**, controls the total overall amount (0 to 100%) of the **Vibrato Effect** that will be heard. This knob can be assigned to a **MC** using (8) to manually control the overall vibrato intensity over time. **Efx Amt** can also be combined with the **Envelope** by setting (11) to **Knob + Env**.
- (8) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **Efx Amt**. You can also set it to control only a subrange of **Efx Amt** if desired (see **section 2.4**).
- (9) **Balance** knob. This MIDI-Controllable knob allows you to set the relative proportion of pitch and volume modulation. When in the center (denoted by a 10/10 reading) the proportion is the same as that set by the **Vibrato** and **Tremolo** knobs. When rotated left (CCW) toward the **Vibrato** knob, the amount of **Tremolo** is reduced and when rotated right (CW) toward the **Tremolo** knob, the amount of **Vibrato** is reduced.
- (10) Click this *assignment-button* **twice** to see a drop-down menu of MIDI Controllers that can be assigned to control **Balance**. You can also set it to control only a subrange of **Balance** if desired (see **section 2.4**).



- (11) **SVS Mode Menu** selections include: **Knob + Env**, **Amt Knob Only**, and **SVS OFF**.
- (12) Click this *assignment-button* twice to see a drop-down menu of MIDI Controllers that can be assigned to control **SVS Mode**. The relationship between the assigned **CC** and the **SVS Mode** is as follows:
0 = SVS OFF, **1 to 126 = Amt Knob Only**, **127 = Knob + Env**

The following **Envelope** settings (13) to (16) apply when (11) is set to **Knob + Env**

- (13) Envelope **Onset** edit box. When a new note is played, vibrato isn't added until after the onset delay period set with this edit box. This delay time is in vibrato cycles. For example if (5) is set for 5.00 Hz, a delay setting of 8 will delay the onset attack of vibrato for 1.6 seconds (8 x 1/5).
- (14) Envelope **Rise** edit box. After the onset delay expires, the '**Vibrato Effect**' will ramp up to maximum over the time period set by this edit box (again in vibrato cycles). In other words this parameter governs how slowly or abruptly the vibrato effect is brought in after the **Onset** delay.
- (15) Envelope **Decay** edit box. After the '**Vibrato Effect**' is established (ie after the onset delay and ramp up), the envelope can be made to decay slowly over time. This edit box allows you to set the number of vibrato cycles over which the envelope decays to the level specified by (16).
- (16) Envelope **Sustain** level edit box. This box sets the percentage of the full 'Effect Amount' that the intensity envelope will decay to over the number of cycles specified in (15).

The following are the **Random Variation** settings (17) to (20)

- (17) The **Vibrato** edit box sets the limit for random variations (as a percentage +/-) of the **Vibrato** knob.
- (18) The **Tremolo** edit box sets the limit for random variations (as a percentage +/-) of the **Tremolo** knob.
- (19) The **Shape Drift** edit box sets an upper limit (as a percentage +/-) of the **Shape** knob setting that can occur from **accumulated** random variations in **Shape**.
- (20) The **Speed Drift** edit box sets an upper limit (as a percentage +/-) of the **Speed** knob setting that can occur from **accumulated** random variations in **Speed**.

- (21) **Instrument Range Display Box**
 Displays the Low Key and High Key of the range set (by the **SAS**) for the Instrument.

5.7 Guidelines for Making Vibrato Settings

Generally, the easiest way to setup the SVS is to recall a preset for a related instrument family and then tweak it for the desired instrument. However, to familiarize yourself with what each parameter does, and for those situations where you may just want to start building a preset from scratch, this section presents a set of simple guidelines for making SVS settings.

A good way to start building a preset is to select the Preset named ‘* **Basic Setup** *’. This calls up a basic vibrato configuration by turning off the **Envelope** and all random variations. It also turns off volume modulation (**Tremolo**) and sets the waveform to a equilateral trapezoid (**Shape** = 33%). It leaves just a simple 4 Hz pitch modulation of about +/- 3 cents. After recalling the **Basic Setup**, assign a convenient MC (such as the **Mod-Wheel**) to control the **Efx Amt** knob and turn off MIDI control of **Speed** and **Balance** so these parameters won’t change accidentally. For the rest of this discussion, we’ll assume you assigned the **Mod-Wheel** to **Efx Amt**. Now, push the **Mod-Wheel** to max (so that **Efx Amt** will be 100%) and then you can begin the preliminary setup.

While playing and holding a note, adjust the **Vibrato** knob for just a little more intensity of the effect than you would ever want for a maximum. You may also want to adjust the **Speed** knob (if 4.0 Hz is not a suitable vibrato rate) for the instrument you are setting up. You may then want to experiment with different waveshapes for the vibrato by adjusting the **Shape** knob.

Now reduce the **Mod-Wheel** a bit until you have a ‘pleasant’ amount of vibrato going and then raise the **Tremolo** control to add some synchronous volume modulation to the pitch modulation. **Don’t overdo this**. Usually only a small amount of volume modulation is needed for a realistic effect. Now, start playing some phrases and use the **Mod-Wheel** in the conventional way to control the intensity of the effect and see if you have things approximately right.

Next, try adding small random variations to **Vibrato** and **Tremolo** using the corresponding **Random Variation** edit boxes. Also, set some **Speed Drift** and, if **Shape** isn’t set to zero, you can try adding some **Shape Drift**. This can provide a particularly realistic-sounding variation to the vibrato (if you don’t overdo it). The drift edit boxes are set as percentages of the parameters they modify so **if Shape is zero, there can be no variation produced by the setting of Shape Drift**.

If you want to avoid multiple sequencer recording passes, and you think you might have your hands full ‘riding the MCs’ for the **SLS**, you might be able to automate some of the **Mod-Wheel** movements you would use by employing the **SVS Envelope** feature. To setup an appropriate envelope, first use the **SVS Mode Menu** to select the **Knob + Env** mode and set **Efx Amt** to 100% (Mod Wheel at max). Then, dial in some appropriate settings for **Onset** delay and **Rise** time and play and hold a long note while you experiment with the **Decay** and **Sustain** settings. You might just be able to come up with suitable settings that will handle most situations without the need for riding the **Mod Wheel**.

Now when it comes to actually using your setup with a specific instrument, you may also want to assign additional MCs to allow you to vary vibrato **Speed** and possibly **Balance** in real time as you play (or under control of your MIDI sequencer) to lend additional variation related to musical context. Finally, you may want to do some iterative tweaking of the parameters and when you get it sounding the way you want, save it as a **User Preset** and **Rename** it accordingly. Then, so you won’t lose your new preset, you can either re-save the **.nkp** for the SVS or, you can re-save the instrument **.nki** (along with the scripts).

The **Efx Amt** knob needn’t be stationary while the **Envelope** is changing. With the **Knob + Env** mode, both controls are active at the same time. However, be sure you understand how the **Envelope** and the **Efx Amt** knob combine their effects before you try to use this mode (see **section 5.3**).

6.0 SIPS Tips, Techniques, and Musings

6.1 Theo Krueger

SIPS provides such an amazing series of tools and you'll quickly notice it can elevate your sample libraries to new levels of realism and natural phrasing. To help you get the most out of it, here are some guidelines that I have found helpful.

Shoot for Equal-Energy Transitions

Shoot for Equal-Energy Transitions. The first and most important thing to keep in mind when experimenting is, "Equal Energy". We want every transition we make from note to note to be as seamless as possible and with equal volume to get the best legato effect. Try loading a Flute patch and keep on playing the same two notes. The first thing you will realize is that there is a small fade-in or a small volume 'jump' whenever a new note comes in. Depending on the XTime you have set, you will need to set the AtkFade and NodeVol accordingly in order to make the transitions seamless and with 'equal energy'. The best way to hear this is to listen at the transients of a sound — the violin bow or the flute breath noise, etc. Once you get those noises to be consistent you are on the right path.

Generally, for a small XTime (of 0.150 and below) you will need a small AtkFade and NodeVol to get the best results, but, if you need a bigger XTime you will realize that these three values are somewhat proportional. So, for an XTime of say 0.500, you will need both a larger AtkFade and NodeVol to keep the transitions seamless. It is important that your samples are 'compatible' with this equal volume thinking. In modern libraries there are a lot of expressive samples with rises and drops in volume. It will be much harder to get good results with those because moving from sample to sample will have volume inconsistencies. Generally, in testing, we found that simple "Sustain" or "sustain vibrato" samples worked the best since they are the most uniformly recorded.

Bending and intervals:

Depending on how lyrical and lively you want your bends to be, you can set the bending value and the time in which it occurs to more or less. Although setting the bend to cover the full interval at first seems like the most reasonable way to think -- since two notes should be connected like that in reality--, for most situations less bend produces better results. When the instrument is in a mix, the bending happens 50% with the script and the other 50% of it is psychoacoustical. So even though a transition with the script might not have as much bend as in reality, your brain will recreate the rest and you will actually hear it. Also, sections require much less bending (around 5-15) while solo instruments can be more expressive with up to 50-70.

The Xtime knob:

Setting Xtime depends mostly on how fast the instrument you choose can play or will play in your song. If it's a piccolo, you will have XTime very low so it can be used even in really fast runs and trills. For a french horn and Cello you can increase the XTime more. Don't forget that you can always change these parameters in realtime by using midi CC's from within the sequencer. No one setting can be perfect for all occasions.

Setting the optimal sample position:

Generally, in order to get better legato phrasing, the sample position (sample-start offset) should be set after the attack of the instrument. You will soon find that this defeats the "overlapping notes" philosophy you might've been following until now... **what a relief!!** Depending on what result you are after, some instruments will need to have the start position right on the attack and others way after it, so if you want the trombone blowing sound to be heard in each transition, setting the position right at the start will give you that effect. If you are programming instruments with embedded vibrato (violin, oboe, etc.), the best sample position is right after the attack, before the vibrato kicks in. When going from vibrato to vibrato there can be some ugly sounding effects while from straight tone to straight tone it is less likely.

6.2 Andrew Keresztes

All of my preset demos were created using East West Quantum Leap Symphonic Orchestra XP Pro to illustrate the SIPS Legato tool. These are just examples of what can be done with the SIPS in real time without tons of editing to get a legato type sound. These demos are not benchmarks... like any instrument, with time and practice using SIPS, **I'm sure much better results can be achieved**. Most of the patches I used had modulation cross-fade features that would allow the instrument to be more expressive. For example, if the modulation controller was at 127, the violin would have a sharp attack and if the mod wheel was at 0 then it would have a slow attack.

Prepare Your Library Instrument

So, first things first. In order to incorporate the SIPS script with EWQLSO I had to eliminate any release groups in the patch. Otherwise, there would be a legato pitch-bend **and** a reverb decay **without a pitch-bend** playing back at the same time. The release groups can be identified by the suffix 'rel' in the group names when you click on the Group Editor button inside the patch editor. Select all the groups that have a "rel" appended to them and under the Group edit drop-down menu choose "Delete Selected Group(s)". Since so many libraries now include release samples, I thought this might be useful information.

EDIT: Since **SIPS 2** now provides options for release sample triggering you may want to try using your library's release groups. Please read **Sections 3.15 and 4.12** for more information on how to configure release groups for the **SAS**.

Use MIDI CCs

Now, from the Script Editor, load in the SIPS Legato tool. In all my mp3 examples, I used continuous controllers (CC) to vary the way SIPS controlled the legato passages. In my case, I assigned CC 110 to both XTime CC **and** BTime CC (both controlled by the same slider). I then assigned CC 111 to Bend CC. This way I was able to ride faders as I was playing parts and vary the amount of the cross-fade and Bend Time between the parent and child note (the 1st and 2nd note) with just one slider. To make the legatos into more of a glissando, I would then ride the CC 111 fader (assigned to Bend CC) as I was playing. It can be a little tricky, but it becomes more intuitive as you do it. Again, this was just my approach to it... I'm sure there are better ways that we will hopefully all share with each other.

So, if you were (for example) playing my solo violin SIPS Legato preset, you would be able to play faster passages with the sliders (in this case CC110 and CC111) all the way down. Then to get a slower glissando effect, you would ride both sliders up 2/3s or all the way up. This is all based upon the initial settings of the SIPS Legato Patch. You can customize it anyway you'd like.

Use an Appropriate Patch

I want to stress how important it is to select the right sound patch to be used in conjunction with SIPS. If your patch has a really slow attack, you won't be able to play faster legato passages... So... always bear in mind what kind of sounds you want associated with SIPS.

Have fun.

Thonex

7.0 File Types and Installing Scripts

The **SIPS** scripts are supplied in readable text formats as well as in NI's proprietary preset formats. The most readable text files have the suffix **_KS.txt**. These source-code files are written using **Nils Liberg's KScript Editor**. Using various extensions and enhancements to the 'bare' KSP syntax, **KS** files are nicely structured and well commented. This, together with the syntax highlighting provided by the editor makes the source code much easier to follow and maintain but, **KS** source files cannot be directly loaded into the KSP editor and run. However, once in the **KScript Editor**, a **KS** file can be quickly 'compiled' into a **K2-Ready** source file by merely hitting the **F5** key. This places a **K2-Ready** version of the source text into the clipboard and from there, you can easily paste it into the KSP editor and hit **Apply**. The **SIPS** package also includes such **K2-Ready** source code saved as files with the suffix **_KR.txt**.

The **KScript Editor** also has a source file 'import' feature that allows larger scripts to be effectively spread over several source files which also enables certain common code such as the **ISCS** or the **KSP Math Library** to be 'factored out' from the main script and used with multiple projects. In **SIPS**, such 'importable' source code modules are designated with the suffix **_KSM.txt**. Usually such 'importable' support files are kept in a separate folder from the main source code files but they must be 'available' when the main script is compiled to **_KR** format.

KR files can be pasted directly into the KSP editor and compiled. However, the KSP has problems handling larger scripts and the more text the source code contains, the slower and more problematic KSP performance becomes. For this reason Nils added several options that can reduce the size of the **KR** file. With the compact output option checked, the **KR** source will have all comments and unnecessary white space removed. Optionally, names of variables and such can also be hash-compacted to 5 characters each and, in addition, there is now a code-optimizer option that further reduces the size of **KR** files. While these compacted **KR** source files are not very easy to read (especially with name compression), they are perfectly 'legal' source text files for the KSP which will have no trouble 'understanding' them. But, because **KR** source files are basically the smallest format acceptable to the KSP, they cause the least amount of KSP slowdown when running.

SIPS scripts are also supplied as **.nkp** files. These files are produced by pasting the **KR** source into the KSP editor and hitting **Apply**. Then a script name is assigned and the script is saved as a **.nkp** file. These 'preset-format' files are the easiest to install because all you have to do is put them in a suitable place under the K2 scripts folder and the next time you launch K2, you will be able to use the K2 Script button to load them into your instrument. Of course you can then save them after that as a part of your instrument using the **.nki** file format.

So, if the **.nkp** files are the easiest to install, why are the others provided? First of all, certain operations such as Auto-Importing of User Presets (during the version upgrade process), require that you open the KSP editor and paste the new version of the source code **on top of** the prior version. The easiest file type to use for that kind of operation will be the **KR** format files. The **KS** files take the most effort to install but if you want to study or edit the source code, you will definitely want to work with the **KS** files because they are the easiest to understand and maintain.

Because of the multiple compacting options now provided, **KR** files can be made several different ways. Assuming that **Code Optimization** is always used, then **KR** files can be made using either the semi-compact mode (which leaves the identifiers uncompressed) or the fully-compact mode. Which mode is used for a given script package depends on a number of factors. Generally, newer scripts (without a legacy chain) should be using the fully-compact mode. However, the current **SIPS** scripts use the **semi-compact mode**. This is because the early versions of **SIPS** were written before the compact modes were added to the **KScript Editor**. By using the semi-compact mode, the names of persistent variables can be consistent throughout the update series (and without the need to add extra conversion steps)

Therefore, if you edit and recompile the current **KS** files, be sure you **set the compact output option** for the **semi-compact** mode. This is important because the KSP persistence machinery requires that new names be identical to the prior names. If you want to be able to update your **SIPS** scripts in the future, the old scripts and new scripts **must use** the same compacting mode unless a special conversion script is supplied. The **SIPS 2** scripts also use the new **Code Optimization** option, so if you want to re-compile any **KS** files, you should set the **KScript Editor** options for **Optimized Code** as well as **Compact Output** (but **not** compact variables).

7.1 Fresh Install vs Update

Kontakt-Ready source code can be placed in the clipboard either by loading the **KS** source into the **KScript Editor** and hitting **F5** or, by loading the **KR** source into the **KScript Editor** (or any suitable text editor) and using **Ctl-A** followed by **Ctl-C** (Windows). Once in the clipboard, the **KR** source can be pasted into the **KSP Editor**. If you are installing for the first time or you want all the parameters to be initialized to their original defaults, before pasting the source code into the KSP Editor, you should first load the **Empty** script to clear out the persistence buffers (alternatively you can hit **Ctl-A**, **delete**, and **Apply**). This is called a **Fresh** install. Conversely, for purposes of updating a script, where you want to transfer your old control panel and/or **User Presets** and such, you need to paste the new **KR** source **on top of** the prior script's source code without hitting **Apply** in between. Therefore, with the old source text showing in the KSP editor, hit **Ctl-A** to highlight all the text then use **Ctl-V** to paste the clipboard over it (actually replacing the old text with the new text) then hit **Apply**. The important distinction between this and a **Fresh** install is that for a fresh install you first clear out the old code and hit **Apply** (or alternatively load the **Empty** script), then you use **Ctl-V** to paste the new code into the **empty** text editor and hit **Apply** again. This is an important distinction because with a **Fresh** install, the persistence buffers will be cleared whereas without this step, prior persistent settings will be carried into the new version.

7.2 Installing and Removing KSP+

KSP+ can be installed in any existing multi or as part of creating a new multi. To add **KSP+** to an existing multi, copy the **SIPS_KSP+.nkm** file to any convenient folder that you keep multis in. Then, load the multi and when Kontakt asks if you want to replace the existing multi, answer **Yes**. This will result in an empty multi containing the precompiled **KSP+ Multiscript**. Now load your instrument multi and when Kontakt asks if you want to replace the existing multi, answer **No** this time. This will create a new multi with the same instruments as the old multi but with the **KSP+ Multiscript** added. You can now resave the updated multi (preferably identified as including **KSP+**).

If you want to create a new multi, begin by loading the **SIPS_KSP+** multi into the empty rack and answering **Yes** so it becomes the 'parent' multi. Then load/drag the instruments and/or banks one by one to build the multi. When you finish building it, save the multi with an appropriate name.

If you later want to remove the **KSP+ MultiScript** from a multi, first load/drag a 'scriptless' multi (such as the **No Multiscript.nkm** file in the **Precompiled** folder) to the rack and answer **Yes**. Your rack will then contain an empty multi without any **MultiScript**. Now drag the multi containing **KSP+** to the rack and answer **No**. The multi without **KSP+** can now be resaved (preferably with a new name indicating the absence of **KSP+**).

7.3 Get the KScript Editor

Even if you don't write scripts, I highly recommend that you get a current copy of the **KScript Editor** in case you need to install a source file written with **KS** extensions. Even for scripts written without extensions, you may still want to use the **KScript Editor** because you will be able to view and/or print the code with syntax highlighting and enjoy the many other useful features as well. Since Nils has graciously made his editor available free, every Kontakt user that intends to use or write scripts should download a copy at <http://nilsliberg.se/ksp/> Of course once you've had a chance to use **Nils' Editor** (not to mention the many other very useful scripts available on his site), maybe you'll want to click his **Pay Pal** button and make a nice donation to show your appreciation for all the wonderful contributions he has made and continues to make for the benefit of the Kontakt User community.

More than you ever wanted to know **About Big Bob**

I'm including this page, about my background and interests, because my motivation for using K2 and writing scripts may be quite different from most of you. As a result, what may work very well for me, may not be suitable for what you are trying to do. However, to the extent that one of my scripts may be useful to others, I offer them freely to the K2 community. I also try to provide a complete documentation package so that anyone skilled in the art of programming can easily adapt and/or improve upon any of my scripts for similar or other purposes without the need for a lot of 'reverse engineering' effort. So, if you find one or more of my scripts not quite 'hitting the mark' for your situation and you want to tweak it, knowing what my objectives were when I wrote it might be helpful.

I'm a retired Engineer (BSEE) and hobby musician. I retired in 1990 after about 30 years that was almost equally divided between Analog and Digital Circuit Design and then (in the latter half of my career), Microprocessor and Software Engineering. On the musical side of things, I've been making (or attempting to make) 'one-man-band' recordings since about 1951 via various forms of 'multitracking'. I've always used real acoustic instruments but, since I'm getting up in years, my 'chops' are beginning to wane. So, slowly over the last 10 years or so, I've become very interested in sampling technology as a way to allow me to continue recording past my prime. However, I won't use synthesis unless I can make it sound like the 'real thing' and, until recently, extremely realistic synthesis has been very hard to do, especially for wind instruments. So my main focus has been in that area. Since I'm an old geezer, I like doing older musical styles. For example, I do a lot of Big-Band Swing and Dixieland stuff. While I can still play all the needed instruments, one of the consequences of 'old age' setting in is that I'm quickly starting to lose manual dexterity (arthritis in the joints and all that sort of stuff). So, I know it's only a matter of time before the quality will begin to suffer noticeably.

I'm telling you all this so you will know where I'm coming from. To do convincing Dixieland, for example, the trombone has to use the slide a lot and to synthesize a realistic trombone glissando, I need to do formant-corrected bending. That's my motive for adding the new Portamento function to the **SLS**. Similarly, I'm hoping that my new **SIPS** scripts will eventually provide convincing legato and vibrato emulation; simple LFO vibrato just doesn't sound realistic enough. And, good sampled vibrato is too hard to come by (and too inflexible). So I've been recording my own wind-instrument playing and analyzing the vibrato and legato sounds and trying to get a handle on what I need to do to simulate it. However, the thing I want to emphasize here is that I may be trying to make a sampled clarinet sound good and you may want to apply these scripts to a string patch. While legato and vibrato techniques vary considerably between instrument families, there is also much common ground. So I'm hoping that the **SIPS** Scripts will eventually have enough flexibility to be used on any instrument. But, keep in mind that it was initially crafted for wind instruments, especially Trombone and Clarinet (my trumpet chops are still fairly good so I'll tackle that last). Another thing I should emphasize is that I intend to use these scripts with Sequencer-controlled Playback, not Live Playing.

All that being said, I'm hopeful that by making my scripts available to the K2 community and encouraging an open discussion about ways of improving them, we may all benefit in the end. I know that many of you are much more accomplished musicians than I am and your input will be most valuable. As a community we are blessed with a number of members who are also excellent programmers and many of these have shown a willingness to share their work with us also. So, I'd like to encourage as many of you as can, to participate in future script development and testing. The Good Lord willing, we may reap a rich harvest of very useful tools and, if nothing else, we may get to know each other a little better, and, you can never have too many friends!

Bob

The Best Things In Life Are Free

SIPS is distributed free of charge, all you have to do is download it and print out the .pdf manual and you'll be all set to go. **SIPS** is also 'open source', which means you can alter it or add to it as you wish. To facilitate that, **SIPS** comes with a complete documentation package including heavily-commented source code.

So, you can either just use **SIPS** as it is or, if you are skilled in the art of programming, you can easily modify **SIPS** without having to do a lot of 'reverse engineering'. However, if you decide to modify or build on **SIPS**, I would like to ask you a favor. Please personalize what you do to distinguish it from the original. By that I mean at least change the Title Block of the source code and use your name as the author. It would also be nice if you would give the new script a different name, and version number series, such as '**Son of SIPS**'. The reason I'm asking you to do this is that I intend to continue the **SIPS** series (at least for a while) and I want everyone to be able to distinguish the original from yours so that the waters don't get too muddy.

My intention from the beginning was to give **SIPS** free to anyone who might benefit from it. However, knowing the effort that goes into something like this, many well-meaning friends have suggested that I at least ought to allow grateful users of **SIPS** to make some kind of gift (if they were so inclined). To that I have always responded that I really didn't want to receive any monetary compensation, knowing that others were benefiting from my effort is reward enough. After all, it's more blessed to give than receive you know.

But, persistent bunch that they are, the next suggestion was that I should at least encourage **SIPS** users who felt the desire (and had the means), to make a donation to one of my favorite charities. Well, after prayerful consideration of this idea, I see no harm in it. It's no secret that I'm an evangelical Christian and I'm always interested in raising money for the Lord's work. So, here's the deal. If you are using **SIPS** and you find it useful, and, you feel an overwhelming urge to want to do something nice for me in return, please click on one of the links below and make a nice contribution to one of these fine organizations. Whether or not you do this will just be between you and the Lord. This is strictly voluntary and I don't want anyone to feel under any obligation whatsoever to do this, but God Bless you if you do.

Have a marvelous day,

Bob

1. Make a donation to the Billy Graham Evangelistic Association.

<https://www.billygraham.org/donate.asp>

2. Make a donation to The Moody Bible Institute of Chicago.

<https://safeweb.moody.edu/support/index.php?afterset=1>

3. Make a donation to Bible League Organization.

<http://www.bibleleague.org/donate/index.php>

4. Make a donation to Union Rescue Mission, Los Angeles.

<https://giving.silaspartners.com/donatenow/unionrescuemission/>

5. Make a donation to Salvation Army.

https://secure2.salvationarmy.org/donations.nsf/donate?openform&t=US_USC*USE*USS*USW&redirect=1

6. Make a donation to Grace Brethren International Missions.

<https://donatelinq.net/donate/gbim-donate.asp?mid=gbimorg>