UniSon 0.70b1
User’s Guide

UniSon is a software system which works in conjunction with a Sound Accelerator or AudioMedia card available from Digidesign corporation. It allows the user to create sound synthesis and signal processing algorithms by connecting functional units using a graphical user interface. New devices can be created by the interconnection of existing devices, or by writing 56001 DSP code “from scratch”.

Disclaimer: This is still a very preliminary version of the system. There are large sections which are missing completely (cutting and pasting, repositioning devices, stereo outputs, etc.), known bugs, and much to do in the way of future expansion. Since this is a part-time project for a single individual, progress is slow. Please send any comments, suggestions, complaints, requests, ideas, etc. to:

Dr. John A. Bate
Department of Computer Science
545 Machray Hall
University of Manitoba
Winnipeg, Manitoba, Canada
R3T 2N2

Phone: (204) 474-6791
FAX: (204) 269-9178
Email: BATE@CS.UMANITOBA.CA

A Quick Tour of UniSon

To see the basic operation of the UniSon system, follow the steps outlined below.

1. Start up UniSon by double-clicking on its icon. An untitled file will be opened. Close it using the “go away” box or by choosing “Close File” from the file menu. Open the file “Demos 0.70b1” instead using the “Open File” menu choice.
2. If using an AudioMedia card, select the “DSP Parameters...” item from the “Circuit” menu, and click on the “Audiomedia” button. (*NOTE*: Most of the

other options here are not yet fully implemented and probably will *not* work. Sorry.) Choose “Enable Card” from the “Circuit” menu. This will start up the signal processor. If this doesn’t work (because you don’t have a card installed), you will still be able to do everything, but you will get no sound (which is rather pointless, really).

3. Select the “Manual FM” circuit from the “Circuits” half of the file window, and then press the “Edit Circuit” button (or simply double-click on “Manual FM”). You now have an active UniSon circuit.
4. Click on the small “On/Off” button on the right side of the screen. You should hear a simple two-operator FM tone. Try changing the carrier frequency, modulator frequency, modulation index, and pitch by using the small slider controls. (Leave the 1/Pi control alone, if you want the modulation index to be accurate.) Click on the two envelope symbols to try different carrier and modulator envelopes.

This illustrates the concept of a *circuit* in UniSon. Each *device* showing on the screen in the circuit window represents a block of DSP code which performs a certain function. A device may have inputs and outputs, as well as internal *controls* which affect its operation. Each connecting line represents a 44.1 kHz digital signal stream, and the DSP code for each device is executed 44,100 times per second. Changes made to the controls affect the signal processor instantly. Changes made to the circuit itself also result in almost immediate reprogramming of the DSP. Here is how to construct a circuit of your own:

5. Close the “Manual FM” window and return to the file window. Click on the “New Circuit” button. This will create a new untitled circuit window.
6. Click on the name “Output” in the list of device names in the bottom left-hand corner of the window. An output device will appear in the “chute” above that list. Using the mouse, drag that output device into the main part of the window, positioning it centered at the bottom of the window. (In the current version, it is not possible to move a device. It must be positioned properly the first time.)
7. Click on the name “Cntl Sine Osc.” (you may have to scroll down to see it) and place one of these devices above the output device but not touching it. (Devices cannot be connected by simply placing them so that their pins coincide. You must use a short “wire” to actually make a connection. I told you it needed work!)
8. Connect the output pin of the Sine Oscillator to the input pin of the “Output” device. (Press the mouse button with the cursor on or near the tip of the

output pin. The cursor should be a small cross. If it is an arrow, you are too far from the pin. Drag the mouse to the tip of the input pin of the other device, and release the mouse button.)

9. The upper scroll bar controls frequency and the lower one controls amplitude. Click on the “units” boxes which currently contain “Dec” to change them to “Hz” and “dB”, respectively. Drag the longer sliders around to change the amplitude and frequency and hear a nice pure sine wave. The shorter sliders control the sensitivity or range of motion of the large sliders (from ultra-coarse to ultra-fine). You can also click on the value to bring up a dialog which allows you to enter the desired value.
10. Click on the “scissors” icon in the top left corner. Delete the “Control Sine” device and the short wire by zapping them with the lightning bolt. Select the “drawing” (hand holding a pen) icon in the top left corner again.
11. Create a simple MIDI-controlled sine wave voice (if you have MIDI input to your Mac) using one each of the “Sine Osc.”, “Output”, “Multiplier”, and “MIDI Note” devices. (If you have no MIDI input device, use two scroll bars and a button instead.) Connect the “f” output of the MIDI device to the “f” input of the oscillator, connect the “A” and “Gate” outputs to the two inputs of the multiplier, and connect the output of the multiplier to the amplitude (“A”) input of the oscillator. Connect the sine oscillator output to the output device. This version of UniSon supports (and requires) the MIDI Manager, and so you should now use the “Patch Bay” to connect some source of MIDI data to UniSon. Change the MIDI channel number by clicking on it, if desired, and then play your circuit via MIDI. I think you get the idea now, right?

This illustrates operation of the circuit window. You can create your own primitive devices, however, and this is illustrated below.

12. Take a look at the “Modular FM” circuit by opening it from the file window. Note that it produces no sound, since there is no output device, nor is there a control for the frequency or the “gate” (on/off). Instead, it contains devices called “Input Pins” and “Output Pins” which simply give these signals names. These devices are available from the same list as the oscillators, etc.
13. Close the circuit window(s) and get back to the file window. Click the “New Device” button. We will create a new device which encapsulates the entire “Manual FM” circuit. First, you will need an appropriate picture. The HyperCard stack supplied contains an appropriate picture (labelled “Manual FM Picture”). Go and cut the picture from there. (If you are not running multifinder or System 7 you will have to exit from UniSon and then come back

later, of course.) Make sure that the picture is cropped absolutely precisely. The tips of the I/O pins must be exact multiples of 5 pixels away from the top left corner of the picture. The background of the cards in the HyperCard stack is a 5 pixel grid to make this easy. Cut out the picture carefully.

14. Once you have cut/copied your picture, get back to the device window in UniSon and select “Paste”. The picture should appear.
15. Now we need to tell UniSon what this device is supposed to do. Click on the “Convert Circuit” button and select the “Modular FM” circuit from the list that is presented to you.
16. The device editor now knows the DSP code needed for this device, and what I/O pins and internal controls it has. Now it must be told the positions of those pins and controls in the graphical picture of the device. Click in turn on each pin name and position it by clicking on the endpoints of the corresponding pins in the picture. Position all of the controls similarly *except* for the “1/π.bar” control. That is meant to be a constant which the user should not be able to see or modify. The mouse controls the position of the top left corner of each control. Note that the “Title” control automatically adjusts to use the entire width of the device.
17. Select the “Save Device As...” item from the File menu. It will warn you that you have not used all the controls, but that is intentional. Click the “Go Ahead” button. Give your device a name and then close the device window. Your new device should appear at the top of the list in the “devices” half of the file window. TWO IMPORTANT NOTES: UniSon does not currently warn you if you are about to lose something by closing a window before you save it. Also, you are simply saving the device into the file window at this point. You have not saved anything to disk until you save the file window itself.
18. Try out the new device. Open a new circuit window and use one of these devices. Connect an “Output” device and either a MIDI input device or a scroll bar (set to “Hz”) and a button to the inputs. Try it out.

It is also possible to create a device directly from raw DSP code together with specifications of the I/O pins, internal local storage requirements, and required controls, but that is covered later. By now you should have a fair idea of the operation of UniSon. Most of it is meant to be fairly simple to use, although it is as yet incomplete.

Reference Notes

1. Menus

1.1 Apple menu

Contains the standard desk accessories and the usual “about” item.

1.2 File menu

“New File” and “Open File” allow UniSon data files to be created and accessed. A file contains device definitions and circuits which use those devices. The “Save”, “Save As...” and “Close” items will change according to whether or not the top window is a file window, a device window, or a circuit window. Saving a file results in its being written to disk. Saving a device or a circuit only results in its being saved into the internal version of the file in memory. You must save the file itself before any changes to devices or circuits become permanent. Note also that the system is not polite enough to warn you when you are about to close something without saving it first. It gleefully destroys your work instead. (Insert evil laugh here.) The “Quit” item is self-explanatory but the same warning applies. The “Revert” item is not implemented.

1.3 Edit menu

Unfortunately, this is largely unsupported. You use it with desk accessories, and to paste pictures into the device window. You may also use “clear” to delete devices and circuits from a file. Regrettably, cutting, copying, and pasted of devices, circuits, and portions of circuits is not yet completed.

1.4 Circuit menu

The “Compile” item forces UniSon to recompile the circuit in the top circuit window and reload it into the Sound Accelerator. This is around only for debugging and development reasons. Any change to a circuit automatically recompiles and reloads it anyway. You should never need this menu item at all. The “Enable/Disable Card” item will turn the Sound Accelerator or AudioMedia on and off. (Don’t forget to turn it on or you will not hear anything.) The “DSP Parameters...” item brings up a cryptic dialog for setting internal registers in the 56001 Digital Signal Processor. It can be used to set the internal clock rate correctly for the AudioMedia card. Most of the rest of this dialog should not be touched yet. You have been warned! “Choose Target Card” is for a multiprocessor DSP system (DSP4). (It is disabled.)

2. Windows

2.1 File Window

Shows the devices and circuits that make up the file. You may select a device or circuit and edit it by selecting its name and clicking on the appropriate “Edit” button. Double-clicking also does the job. You may also create a new device or circuit using the appropriate button as well. You may delete devices or circuits by selecting them and using the “Clear” menu item. NOTE: Editing a device that is already present in a circuit is usually fatal! You may edit the device, but you should then do a “Save Device As...” with a new name. You can then fairly easily zap and replace the old versions of the device with the new version in all circuits that use it.

2.2 Device window

Allows devices to be created or modified. Device pictures may be imported by using “Paste”. The picture should be in normal “PICT” format and all I/O pins MUST be positioned exact multiples of 5 pixels away from the top left corner. The “Read Code” button will read in a text file containing specifications of a device to be created “from scratch”. The format of this file is discussed later. The “Convert Circuit” button will allow a circuit to be encapsulated to form a device. All “Input pin” and “Output pin” devices and all controls will appear in the appropriate areas of the device window. I/O pins and controls may be positioned inside the picture by selecting their names and then using the mouse. Clicking on an existing pin or control in the device will highlight the corresponding name, and allow the pin or control to be repositioned. Don’t forget to save the device before closing it!

2.3 Circuit window

The most important type of window since it contains the “active” circuits that produce sound. The three icons in the top right corner allow drawing, deleting, and moving of devices and connecting lines. (Except that moving is not implemented yet.) The highlighting of these icons is flakey. The area below that is the “chute” where a copy of the selected device appears. Dragging the device from the chute into the main part of the window adds it to the circuit. The names of all available devices appears below the chute. Connecting lines are simply drawn with the mouse. They must be vertical or horizontal, and the system will automatically place everything on a 5-pixel grid. (So you can be sloppy by ±2 pixels.) Device pins will NOT be connected simply by placing the devices so that their pins are coincident. You MUST use a connecting line.

3 Operation of UniSon circuits and the supplied devices

3.1 Basic concepts that you must know

In a digital signal processor, all data is typically represented by fixed point real numbers in the range $-1 \leq X < 1$. All values, whether they represent amplitude, frequency, angles, or whatever must be mapped into this range. All input pins, output pins, and connecting lines in UniSon represent a 44.1 kHz stream of values in this range (which I will refer to as ±1 although +1 is actually not a valid value). When you are using a value to represent frequency, the range ±1 represents ±22.05 kHz (the Nyquist frequency). When talking about amplitude, I assume that the value 1 is 0 dB, and therefore 0.5 is approximately -6 dB, etc. When representing angles, ±1 is clearly used to represent ±180 degrees or ±π radians. But these are simply different ways of interpreting the same values. Underneath it all, everything is always in the range ±1.

Each device in UniSon represents a segment of machine code for a 56001 Digital Signal Processor. It may have one or more I/O pins which allow it to be connected to other devices. Each set of connected pins and “wires” in a circuit is assigned a memory location in the DSP (*not* in the Mac itself) and that location will always hold the current value of that digital signal (which will change at the rate of 44.1 kHz). Each device may also contain local values or tables internally. (For example, a sine oscillator stores the current angle value internally, and a envelope generator stores a table of envelope segments.) Devices may also contain *controls* which are objects that may be manipulated by the user. Each control may affect one or more I/O pins and/or internal storage values or tables of the device containing it. When a control is manipulated by the user, most of the work is done by the host Macintosh, which in turn will place certain values directly into the appropriate DSP memory locations using fast interrupts. Simple controls like scroll bars are effectively “free” as far as the DSP is concerned. They take up no memory and insert no additional instructions into the DSP code at all.

3.2 The Scroll Bar and the Probe

Create a circuit containing only a Scroll Bar connected to a Probe. This will illustrate the nature of data in UniSon (and DSP in general). The top half of the scroll bar informs the user of its present value. The “units” box in the top right corner determines how this value is displayed, and how the slider behaves. The actual value produced by the scroll bar is actually a value in the range ±1, as always. The choices are:

Dec :	Displays the value as a normal decimal number in the range ±1.
Hex:	Displays the actual 24-bit hexadecimal value of the output.
Rad:	Displays the value in radians in the range ±π.
Deg:	Displays the value in degrees in the range ±180.
Hz:	Displays the value as a frequency in the range ±22.05 kHz. When this unit is selected, the slider acts logarithmically, since this is more useful for frequencies.
dB:	Displays the value as an amplitude in dB. The output value 1.0 is treated as 0 dB, which means that 0.5 is -6.02 dB, 0.25 is -12.04 dB, etc. The output value 0.0 is really minus infinity on a dB scale, but the arbitrary value -140 dB is displayed. This unit also causes the slider to behave logarithmically.
±16:	Displays the value as a number in the range ±16. In other words, it pretends that the decimal point follows the 4th bit. Don’t forget that the actual value produced is still ±1, however. This just multiplies the value by 16 in the display. However, this can be useful if the output value is eventually multiplied by 16 by an appropriate device, so as to match the display. See the FM circuit examples.

The “sign” box in the top left corner may be used to restrict the output to only positive values, only negative values, or signed values. Note that when a logarithmic scale such as Hz or dB is chosen, this will automatically change to plus-only or minus-only since no logarithmic scale can possibly go through the value 0.

The larger slider controls the value of the output directly using either a linear or logarithmic scale. The smaller slider controls the sensitivity of the larger one. Every pixel that the smaller slider moves to the right decreases the range of the larger slider by a factor of 2. This allows a full 24 bits of precision to be used, with patience. By clicking on the displayed value, you may also simply type in the desired value.

The probe is simply the top half of a slider. The slider and probe may be set to any combination of units so that you can see what is really going on. Try it.

3.3 Other User-interface devices

The other user-oriented I/O devices are the button, the MIDI devices, the spinner, and the dial. The button is a simple on/off switch which produces a value of 0 or (almost) 1. The dial is a circular control that may be manipulated with the mouse, producing values in the range ±1. The spinner is an output-only version of the dial that may be used like a probe. It is useful because it shows graphically the circular nature of all data in a DSP system. The MIDI devices (four of them) respond to MIDI note, pitch bend, aftertouch, and controller messages. The small box containing the channel number is a control which will allow you to select any MIDI channel, and the MIDI controller devices contains a similar box to select the controller number. The outputs of the MIDI note device represent note frequency, note velocity (a linear scale 0..1), and a gate which indicates that a note is being played (a 0/1 value). Each MIDI note device responds only to note on/note off events, and is monophonic. However, you may have as may of these as you like, and if several are set to the same channel, UniSon will assign incoming notes to them relatively intelligently to provide a polyphonic system. This version of UniSon uses Apple’s MIDI Manager to obtain the incoming MIDI data. You must already have a copy of the MIDI Manager to use MIDI with UniSon. If you don’t, please let me know.

3.4 Sound generators

There are various oscillators and other signal generators provided. The white noise generator gives white noise at a specified amplitude. The “Random” device will give random output values which change at the specified frequency. For example, a frequency value of 2 Hz will change the output twice per second to a new random value. The sine oscillator will give a clean, linearly interpolated sine wave with a given frequency, amplitude, and initial phase. For applications where the phase is important, the “Sync Sine” device has an additional sync input which will force the internal angle to 0 when the sync input changes from 0 to 1. When used with a MIDI gate or button, this will cause all of the sine oscillators in an FM system, say, to begin together at phase 0. There is also a “Cntl Sine” which is simply a combination of a sine oscillator and two scroll bars. There are also square, pulse, sawtooth, and triangle oscillators. Be aware, however, that these produce “perfect” waves to the best of their ability. Such waves contain frequencies above the Nyquist frequency and will produce aliasing. The “Wave Table” device allows the user to specify a wave using a table of 256 24-bit values. Clicking on the small “wave” control will bring up a dialog box to let the user enter harmonic/amplitude/phase values to build a wave. Waves may also be saved and loaded from special wave files.

3.5 Modifiers

There are three envelopes and two filters provided. The ASR is a simple attack-sustain-release envelope with a gate input and an envelope output. The attack may be either linear or exponential. The Envelope (8 seg) is an 8-segment general envelope in which each segment may be log, linear, or exponential, or anywhere in between. This results in fundamental problems when working with a digital system with fixed time and amplitude resolutions. When you change one value, others may change to the nearest value that can actually be achieved. Keep the “Mult” field at 0 (linear segments) when in doubt. This thing needs work. Experiment with it. A much easier envelope is now provided by the “Envelope (table)” device. This uses DSP code that is really the same as the wavetable oscillator, and it is controlled by a 256-entry table in the same way. Try it. You’ll like it.

The devices “FOF” and “SOF” are simple first-order and second-order digital filters. The “a” values are the coefficients of the numerator polynomial whose roots are the zeroes and the “b” values are the coefficients of the denominator polynomial whose roots are the poles. (You have to have a certain degree of familiarity with digital filters to understand and use these devices.)

3.6 Arithmetic

There are devices which will perform addition and multiplication, which are fairly self-explanatory. There is also a “16xy” device which will multiply two values and then multiply the result by 16 (shifting it 4 bits left). This is useful in conjunction with the ±16 settings on the scroll bars at times. Note that the output of all of these devices is cyclic and produces a result which is congruent to the “correct” result modulo 2, and is in the range ±1 (not 0..2). For example, adding 0.75 and 0.75 will give –0.5 which is congruent to the correct answer (1.5) modulo 2, but lies in the proper range. Try connecting two scroll bars to a “16xy” device and connecting a “spinner” and a “probe” to its output. Play with it for awhile.

3.7 I/O and device creation

There is an “Output” device which will cause sound to be produced at the output of the Sound Accelerator. For now, only **one** of these devices should be used, and stereo outputs are not available.

There are also “Input Pin” and “Output Pin” devices which are used only to build circuits which will subsequently be encapsulated to form a device. They do nothing but give names to certain signals for use by the editor.

4. Creation of Devices from DSP code.

Besides encapsulating a circuit to form a device, you may also create one directly by specifying its DSP code and all of the related information. This is not for the novice user. You must be familiar with 56001 machine code, and have an assembler and development system available to you, and preferably a simulator for testing also. To create a UniSon device, you must then create a “Device File” such as the one shown partially below (this is the “Control Sine” device). (The device files for all of the supplied devices are included with UniSon. Look at them for more examples.)

```
Pins
 forward
 input phase
 output sine

Storage
 angle = 000000
 freq  = 000000
 ampl  = 000000

Controls
 ScrollBar FreqCntl->freq
 ScrollBar AmplCntl->ampl

Code
 448000        <freq>  ; move freq,X0
 568000        <angle> ; move angle,A
 578040        <phase> ; add  X0,A     phase,B
 540018        <angle> ; add  A,B      A1,angle
 21A600                ; move B1,Y0
 44F400 000080         ; move #>$000080,X0
    etc. etc. etc.
Resources
 21 TTTTT ;21 cycles, all processors
```

The device file consists of five sections identified by keywords. The “Pins” section begins with a line containing only the word “Pins” and it defines the I/O connections to the device. The next line may (optionally) contain only the word “forward”. This is a bit hard to explain, but it means that the code in this device should be executed BEFORE the code for any device connected to its outputs. Without this line, it will be done AFTER such devices. The “Unit Delay” device will NOT have “forward” in it so as to ensure proper execution order in digital filters. Almost all other devices SHOULD have “forward” in them. Each remaining line in this section consists of one of the keywords “input” or “output” followed by a name for the pin chosen by the user. This name will appear later in the device editor window in UniSon. The “Storage” section defines the internal memory locations (local variables/tables) needed by the device. Each line consists of a name for the location followed by an equal sign and its initial value(s). A table is formed simply by listing more than one value, separated by blanks, or by using #tablesize. For example “WaveTable = #256” could be used to allocate a 256-word table. All values entered are in hexadecimal. The “Controls” section defined the controls which will be provided to affect the operation of the device. Each line in this section contains the type of control, a name for the control which will appear in the device window, the symbol ‘->’, and a list of the I/O pins and/or storage locations (or tables) that should be affected by the control. The controls are really routines that form part of UniSon itself and which run on the Macintosh host, and not of the DSP. (It is possible, but much more difficult, to define your own kinds of controls as well.) Each type of control affects a fixed number of DSP values or tables, and the correct number of items must follow the “->” symbol. The type of controls currently defined are:

Type “ScrollBar” is a standard scroll bar which has a single output which is set to its current value.

Type “Button” is a simple on/off button which has a single output that is set to 0 or 1 (actually 7FFFFF hexadecimal).

Type “EGcontrol” provides an envelope consisting of 8 general-purpose segments and a release rate. It is intended for use only by the “EG” device, and will not be discussed further here.

Type “Spinner” is a circular indicator within a rectangular box which monitors the state of a single signal line.

Type “Dial” is a circular knob which may be manipulated by the user to input a single value.

Type “Probe” is the top half of a scroll bar which monitors rather than controls a single signal value.

Type “NameBox” is used to let the user give a name to a device. This is primarily useful when encapsulating circuits to form devices since there must be a way of identifying each control in the circuit uniquely. All devices which contain one or more controls will automatically be given a NameBox by the device editor. It should not be specified explicitly in a device file at all. A Name Box (or “Title”) has a dynamic width which always grows to the right edge of the device. Room must be left for it in the device’s picture.

There are also controls for the four MIDI controls and the wave table, but these are not really intended for use by the user.

The next section of a device file is the code section which contains the actual DSP code. Each line should contain a single 56001 DSP instruction in the form of one or two 6-digit hexadecimal values. These should come from the output listing of your 56001 assembler. (There is no automatic mechanism for importing .lod files, but the code must necessarily be quite small in order to be useful at 44.1 kHz so it is not a very big job to type it in.) There are a lot of restrictions on the instructions and addressing modes which may be used, the major one being that you may ONLY use an X memory reference OR a Y memory reference, but NOT both, in each instruction. This is because a signal may wind up in either X or Y memory and it cannot be predicted at compile time. The XY and Long and other more powerful addressing modes are too restricted to allow subsequent changes to a single instruction to be made without completely rewriting the code. Contact the author via email for a more complete list of restrictions if you are interested.

Each 56001 instruction that refers to a memory location that is actually an I/O pin or a storage location, or which contains a jump address, must be patched by UniSon’s built-in linkage editor in order to become part of a circuit. Therefore, all such instructions must be followed by the name of the pin or storage value and/or the jump label, enclosed in angle brackets. (See the example above.) The instruction referenced by a jump must contain the appropriate label, followed by a colon, at the start of the line containing that instruction. (See the EG.dev file for example.) It is convenient to place the original 56001 assembler source code on each line as a comment following a semicolon, as in the example above.

The creation of such devices directly from 56001 code is not meant for novice users (actually, it is not meant for anyone other than the author). Please contact me by email for more details if you need them. The device files for all supplied devices are provided, and they provide a fairly complete set of examples of what you can and cannot do.

The last section is the Resources section. It should consist of a single line containing an integer giving the number of DSP clock cycles needed to run the code for the device, and a string of 5 flags (T/F) indicating which processors it may be assigned to an a multi-processor system. Most of this information is ignored except in a multiprocessor environment (and right now the custom-built one in my office is the only one UniSon can use). So just use 0 TTTTT and you will be no worse off.

5. Device Pictures

There is not too much to say here. The picture for any UniSon device may be created by your favorite drawing or painting program. It must them be copied and pasted into the device window in UniSon. The only restriction is that the tips of all of the I/O pins must be aligned to a 5-pixel grid relative to the top-left corner of the picture. I have provided a HyperCard stack which contains the pictures of all of the supplied devices, and whose background is the required 5-pixel grid. This makes the creation of pictures fairly easy. One of the cards contains pictures of the various kinds of controls. These may be placed freely (no grid restrictions). These are simply “dummies” of course which should be used so that you can see what the device will look like. That portion of the picture will be overwritten by a real, live control only when it becomes part of an actual live circuit. These pictures also show you the exact size and shapes of the controls, which are established by the corresponding routines inside of UniSon and may not be changed. (Except for the title or name box, which always expands as far as it can to the right.)

6. A Note About Speed

When working at 44.1 kHz, there is not a lot that you can do in real time even with a fairly powerful 20 MHz DSP chip such as the 56001 on the Sound Accelerator. At the rate of 10.25 MIPS, there is enough time to execute a little more than 200 machine language instructions per sample. In a flexible system like UniSon in which virtually anything can change, all values must be stored in memory locations and the 56001’s registers may only be used for temporary results within a device defined directly by DSP code. This means that many of the powerful pipelining features of the 56001 go to waste, and a lot of “MOVE” instructions are needed. The “Sine” module generates very high quality linearly-interpolated sine waves with control over frequency, amplitude, and phase. It take about 20 instructions to do this. This means that only a few of these devices may be used in a circuit together with the other necessary devices before the Sound Accelerator loses its ability to keep up. The other devices, like adders and multipliers and envelope generators, use up instructions, too. The amount that a single processor can do in this environment is rather dissapointing. Don’t expect to patch together a 4-operator FM voice with 8-note polyphony. That would likely take 8 Sound Accelerators. I have a working multi-processor version of UniSon at the moment but it is a prototype which cannot easily be used by anyone else. :(

7. Communication and Requests

This version of UniSon is a long way from being as complete as I would like it to be, and there is a long list of additions and fixes waiting to be made. Please let me know of specific needs, wants, problems, bugs, comments, suggestions, etc. at the address(es) on the first page. If you need a specific kind of device, and if it is not too complex, and if I can come up with the time or a graduate student, then please send a request. I can create the appropriate device file for you. On the other hand, if you have 56001 expertise and create any useful or interesting devices, please mail them to me so that I may distribute them to other users. If you want to get future (more complete) versions of UniSon or new devices for it, please send me your address (email, preferably).

NEW FEATURES ADDED FOR VERSION 0.62

1. Midi Manager support has been added. You must have MIDI Manager installed in your system and you must use the Patch Bay desk accessory or application to connect something to UniSon. Please contact the author if you don’t yet have MIDI Manager. The types of messages supported are note on/off, channel pressure (aftertouch), pitch bend, and all of the controllers (mod wheels, volume, joysticks, pedals, etc.). You must know the controller number, however, and type it into the dialog box that appears when the mouse is clicked on that number. (The ever-present mod wheel is controller number 1, which is the default value.)

2. There is a “wave table” oscillator which will produce any wave shape with up to 128 harmonics using a 256-entry table. A dialog box allows you to specify wave shapes by means of amplitude and phase values. The results may be stored in special UniSon “wave” files, two of which are supplied as examples. If you wish to create these files from scratch or convert other sources of waveshapes into this format, please contact the author and I will supply you with the details of the required file format.

3. Clicking on the value in a scroll bar now brings up a tiny dialog box allowing a new value to be entered directly. (Its about time I added this one!)

4. Bigger circuit windows are now possible.

5. The Audio Media card is now supported by its own menu item under the “Circuit” menu.

NEW FEATURES ADDED FOR VERSION 0.70

1. **Communication with MAX.**
UniSon circuits may now communicate directly with patches in Opcode's MAX program. This operates at a very low interrupt-driven level of the Mac operating system. It is quite complex "under the hood" but should be very easy for you to use. Place the "UniSon MAX Object" in your "max-startup" folder in your Max folder. Place the "PitchTable" file where Max can find it easily, as well. You *must* be running system 7 for any of this to work! Run both UniSon and Max at the same time. Open the "Demos 0.70b1" file in UniSon and open the "Max FM Module" circuit. Enable the DSP card. Note that there are two "Max->UniSon" devices at the top. Open the file "UniSon 21" from Max. Note the corresponding "UniSon" devices in this patch. You make the connection between Max and UniSon by sending a "connect" message to ANY UniSon object. (I generally make a "UniSon master" object for this purpose.) Select "UniSon's M->U Port" from the dialog that appears. By the way, this works even if UniSon and Max are running on different Macintoshes connected by a network! Now Max and UniSon are communicating. If the is a UniSon object in Max and a "Max-UniSon" device in Unison with the *same name*, then they are connected. That's all there is to it. The lower 24 bits of a long integer value in Max will be sent to UniSon, which will treat it as a number in the range -1..1 as usual. This means that Max may have to scale things a bit to make the range fit properly. See the files "UniSon 21" and "Test MAX->UniSon" to see how to do this. The Max table in the file "Pitch Table" is useful for translating a note number in Max into a frequency value that UniSon can understand. The pascal program that I used to create this file is also supplied, for your interest.

2. There is a new f(x) device which will use linear interpolation in a 256-entry table to compute any function you like. The function must be supplied as a file containing 257 long integer values. A short pascal program which will generate such a file is included (CreateFxFile.p).

3. There is a much nicer table-controlled envelope generator. (Device name "Envelope (table)". Most of it is self-explanatory. Try it. (Double-click on a vertex point in the envelope to remove it. But you have to be precise.)

4. The big one: UniSon handles a multiprocessor card for increased capabilities. Unfortunately, this one does not do you, the user, any good

since there is only one prototype card in existence. I have disabled all of the options that support this card (I hope!). If you find one that I missed, *don’t* play with it. It will likely crash any Mac without the special card in it!

5. Devices may be a bit slower in this version of UniSon. There is a long and boring technical reason for this that involves the multiprocessor.

6. The format of the .dev files (for users that want to create their own devices directly from 56001 code) has changed. See the appropriate section earlier in the manual.

7. There is a new dialog that appears when you select “DSP Parameters” from the “Circuit” menu. *Most of this is not fully debugged yet.* I suggest that you use the “Audiomedia” button if needed, but leave the rest untouched. If you know a lot about 56001’s and the Sound Accelerator, go ahead and experiment.

8. There are likely a lot of other things that have been fixed, improved, or added which I have neglected to mention. Look around and try things. Please feel free to contact the author by email with any questions, problems, suggestions, or whatever. I will be glad to answer you. (I just don’t have much time to write manuals at the moment. :)

WISH LIST FOR UniSon:

There are a lot of unimplemented sections in UniSon, and a lot of missing devices. Here are the main things on my “to do” list right now, in no particular order. Please feel free to let me know which ones are high on your priority list, or suggest additions to the list.

A selection of better higher order filters.

Support for colour pictures and devices.

Printing of circuits.

Cut/Copy/Paste of circuits, subcircuits, devices, etc.

Movement of devices and lines in a circuit.

More flexible line drawing, allowing “L” shaped lines to be drawn in one operation. Also allow devices to be connected directly without a line.

Text fields for comments in a circuit.

Stereo outputs.

Automatic summing if several output devices appear in the circuit.

Prompting for a save operation if changes have been mode and the user attempts to close a circuit or device or file.

More robust operation when memory space becomes low. UniSon, like many other Macintosh applications, finds it difficult to handle all possible situations in which memory suddenly runs out. When memory space becomes low, system crashes are not uncommon. Make sure there is enough memory to avoid these problems.

An “undo” facility.

Improved device selection using hierarchical pop-up menus instead of the current “flat” scrolling list of titles. Also smaller and better (and reliable) icons for tool selection.

More devices, including a delay line device.

Support for a multiprocessor to extend the capabilities of UniSon. I currently have a card containing 4 56001 DSPs which I built specifically for UniSon. Perhaps some commercially available card will be supported in the future.

Statistics on the amount of DSP time and space consumed by a given circuit.

etc. etc. etc.