ensoniq™

Advanced Sampler's Guide

Mirage™

DIGITAL SAMPLING KEYBOARD

FEATURES -

- Step-by-step sampling tutorials.
- Sampling theory and techniques for multisampling and looping.
- Fade-in, fade-out, merging waveforms, sound reversal.

CONTAINS -

- Mirage Advanced Sampler's Manual.
- Mirage MIDI specs and system—exclusive codes.
- 2 Mirage Advanced Sampler's Operating System (MASOS) Diskettes.

Model No. ASG-1
Part No. 9920000301

Credits:

Written and Designed by: J. William Mauchly
Additional Material: Alex Limberis, Thomas Metcalf, John O. Senior and Bob Yannes
Editing: Robin Weber and Bob Yannes
Illustrations: Bob Yannes

Limited Warranty

Although every effort has been made to insure the accuracy of the text and illustrations in this Guide, no guarantee is made or implied in this regard.

What is Covered:
This warranty covers all defects in material and workmanship for thirty days from the date of purchase from an Authorized **Ensoniq** Dealer.

How to Obtain Warranty Performance:
Return your diskette(s) in their original case to an Authorized **Ensoniq** Dealer along with proof of purchase. Defective diskettes will be replaced.

NO CLAIM FOR WARRANTY WILL BE HONORED WITHOUT PROOF OF PURCHASE.

Limitations of Implied Warranties and Exclusion of Certain Damages:
Any implied warranties, including warranties of merchantability and fitness for a particular purpose are limited in duration to the length of this warranty. **Ensoniq Corp.**'s liability, for any defective product, is limited to repair or replacement of the product.

Ensoniq Corp. shall not be liable under any circumstances for:
Damages based upon inconvenience, loss of use of diskettes, loss of time, interrupted operation or commercial loss.
Any other damages, whether incidental, consequential or otherwise, except damages which may not be excluded under applicable law.

Some states do not allow limitations on how long an implied warranty lasts and/or do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations and exclusions may not apply to you.
This warranty gives you specific legal rights and you may also have other rights which may vary from state to state.

*Apple and Apple IIe are registered trademarks of Apple Computer Corporation.

Table of Contents

INTRODUCTION

The **Mirage** is an eight-voice digital sampling keyboard combining the complexity and accuracy of digital sounds with an analog synthesizer section which provides programmable control over filtering, envelopes and modulation. Digital sounds can be tailored to a player's taste while the velocity-sensitive keyboard adds new dimensions in creative expression. The *Mirage Musician's Manual*, which comes with the instrument, is the introduction to operating the Mirage. This *Advanced Sampler's Guide* is for musicians who have experience using the Mirage and want to make original sounds for the instrument. It covers the challenging process of sampling sounds, along with many fine points of the Mirage and its hidden capabilities. It is an introduction to digital sampling and a reference manual for the Mirage.

Enclosed with this guide is a disk which contains the *Mirage Advanced Sampling Operating System* (**MASOS**). The MASOS disk is a special tool for the advanced user. It features new commands for manipulating sounds and supports the **Input Sampling Filter** and personal computer-based **Visual Editing System**. Use the MASOS disk for sampling, then store your samples on **ENSONIQ** Formatted Diskettes available from your dealer. This book also contains the reference manual for MASOS.

Part I - Understanding the Mirage, is an overview. It assumes the reader is familiar with analog synthesis as it shows how the Mirage is similar to, and quite different from, a conventional synthesizer. It introduces the vocabulary of digital sampling.

Part II - Mirage Parameters, delves into the details of the Program, Wavesample and Sampling controls: why they are there, what they do, and how they are used.

Part III - Sampling Tutorial, gets to the heart of the matter - how to make good samples. This is a step-by-step guide through four different sampling situations. It progresses from a simple unlooped sound to Multisampling.

Part IV - Advanced Sampling Techniques, contains the most complicated details about sampling, in theory and in practice. It explains the many techniques and special considerations involved in creating your own sounds on the Mirage.

It is surprisingly easy to come up with wonderful sounds on the Mirage, even from the most modest sound sources. It is only necessary to understand a little about how the Mirage works and some simple sampling techniques. The reproduction of natural sounding acoustic instruments, on the other hand, is sometimes extremely difficult - it is an art, not a science. As with studio recording, mastery takes a sensitive ear, a good understanding of the process, experience and a good deal of patience.

PART I - UNDERSTANDING THE MIRAGE

The Mirage is different from conventional synthesizers in a fundamental way. Synthesis is the creation of sounds by combining basic building blocks of sound. The synthesizer can imitate acoustic sounds by approximating their shape and tonal color. A digital sampling keyboard, like the Mirage, is more similar to a tape recorder; it records sound and plays it back. The recording is made in digital memory, rather than on tape. The Mirage can then shift the pitch of the recording by changing the playback rate. This is exactly like changing the speed of a tape recorder; the faster the tape, the higher the pitch. This single feature sets the Mirage apart from all synthesizers that are based on oscillators with fixed waveforms (even those with digitally generated waveforms). To make digital sampling useful musically, it is important to let the performer control the sound and modify it. The Mirage gives the musician expressive control with a velocity sensitive keyboard and synthesizer-like modification of the sound, with the familiar Filters, Envelope Generators, and LFO's.

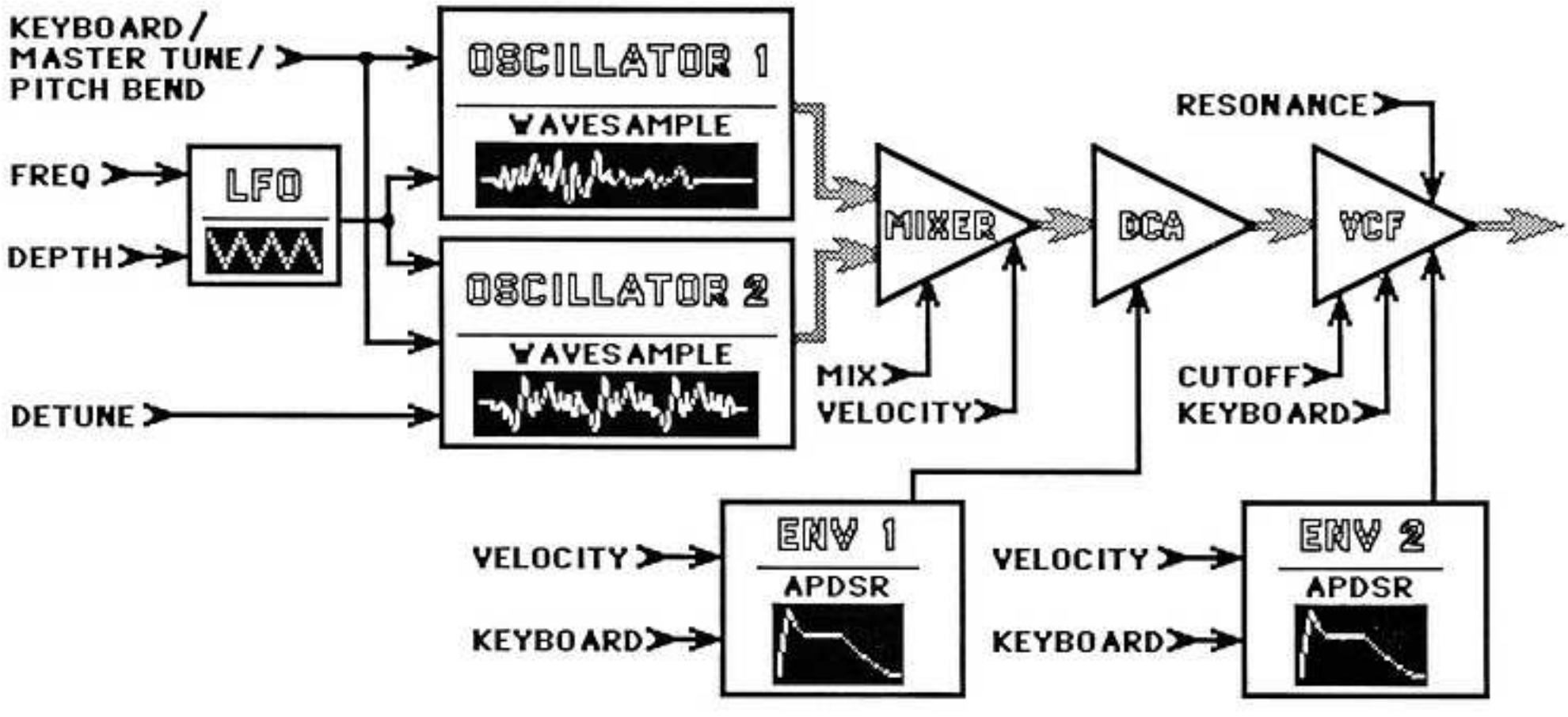
VOICE ARCHITECTURE

The Mirage has eight sound-producing channels, called *Voices*. When a key is pressed, one voice is called into play. It is given a set of explicit instructions about how to play a note. These instructions are based on what *Program* is selected, what key was pressed and how hard it was hit. A Program (or patch) is a set of parameters which affect <u>all</u> voices assigned to that program.

Each voice is made up of two **Digital Oscillators**, one **Low Frequency Oscillator** (LFO), a **Mixer** and **Digitally Controlled Amplifier** (DCA), a **Voltage-Controlled Filter** (VCF), and two **Envelope Generators.** They are permanently set in a standard synthesizer configuration, with the LFO feeding both oscillators, the oscillators going into the filter, one envelope modulating the filter, and the other controlling the volume. However, the digital oscillators in the Mirage are quite different from conventional oscillators.

Digital Oscillators

We use the term 'Oscillator' loosely to mean a sound generator. While a conventional oscillator will repeat the same waveform over and over, thereby giving it a frequency, the Mirage plays back a recording of a sound. If the original sound had a pitch, then the output of the Mirage oscillator will have a pitch. Likewise, if the original sound had little pitch content, as with a cymbal or a gunshot, the output will be unpitched. What the Mirage oscillators do is transpose the sound up or down. They do this by changing the rate at which the recording is played back.



Wavesamples

The digital recordings used in the Mirage are called *Wavesamples*. When a key is pressed, each oscillator is told to play back one wavesample. When both oscillators play the same wavesample, the resulting sound can have a flanged or chorus effect. The oscillators can also play two different wavesamples, giving a variety of mixing and layering effects.

Wavesamples can use different amounts of memory. There is a fixed amount of memory in the Mirage, but it can be divided into as many as 16 different wavesamples. The length of the wavesample in memory can be deceptive; through *looping* , even a short sample will sustain forever (or as long as you care to hold the key down).

Mixing

The output of the two oscillators are combined in a special Mixer which also controls the final volume of the sound. The mix can be controlled by the velocity of a key or by the mod-wheel. The output of the mixer then feeds the filter.

Mix Mode and **Chorus Mode**

The two oscillators of each voice usually play the same wavesample. This is *Chorus Mode*; by detuning the second oscillator slightly, the two oscillators will beat against each other causing a phasing effect. *Mix Mode* is the alternative to Chorus Mode. In Mix Mode, the two oscillators play two different wavesamples (Oscillator 1 uses the odd numbered wavesamples and Oscillator 2 the even). This can produce a variety of effects, especially when the mix between the two is varied. For example, the mix between a clean guitar and a distorted guitar can be controlled by the velocity. Layering of two sounds, such as saxes and trumpets, can be balanced with the mod wheel.

In Mix Mode, as in Chorus Mode, the second oscillator can be detuned and the mix and mix modulation work the same way in both modes. The only difference between the two modes is in which wavesample the second oscillator plays.

Low Frequency Oscillator

The LFO in each Mirage voice modulates the pitch of the two oscillators for vibrato. The waveform is a triangle wave. The amplitude can be preset in a program or it can by controlled from the mod-wheel.

Filter

Each Mirage voice incoprorates a 4-pole Voltage-Controlled Low-Pass Filter with variable Resonance. The filter serves three functions: It filters out unwanted by-products of digital oscillators, called *aliases*, which are a kind of high-frequency noise. The filter is driven by an envelope generator to provide dynamic control over the shape of a sound varying from subtle effects (making the piano sound duller as it fades out), to more obvious frequency sweeps, as in a conventional "synthesizer brass" sound. Finally, the filter provides velocity-sensitive brightness for the sound.

Envelopes

The two Envelope Generators, one for amplitude and one for the filter, give the Mirage remarkable touch sensitivity. They are similar to traditional ADSR envelopes, with the addition of one parameter, the *Peak* level (APDSR). The seperate Peak parameter allows more precise velocity control as compared to a simple envelope amount control. The Peak and Sustain levels are independently velocity sensitive, allowing a wide variety of dynamic envelope shapes. Additionally, the envelope time variables - Attack, Decay, and Release - are also modulated in the following ways: the attack time can decrease with higher velocity; the decay time can decrease with keyboard position and the release time can be affected by how fast you lift up a key.

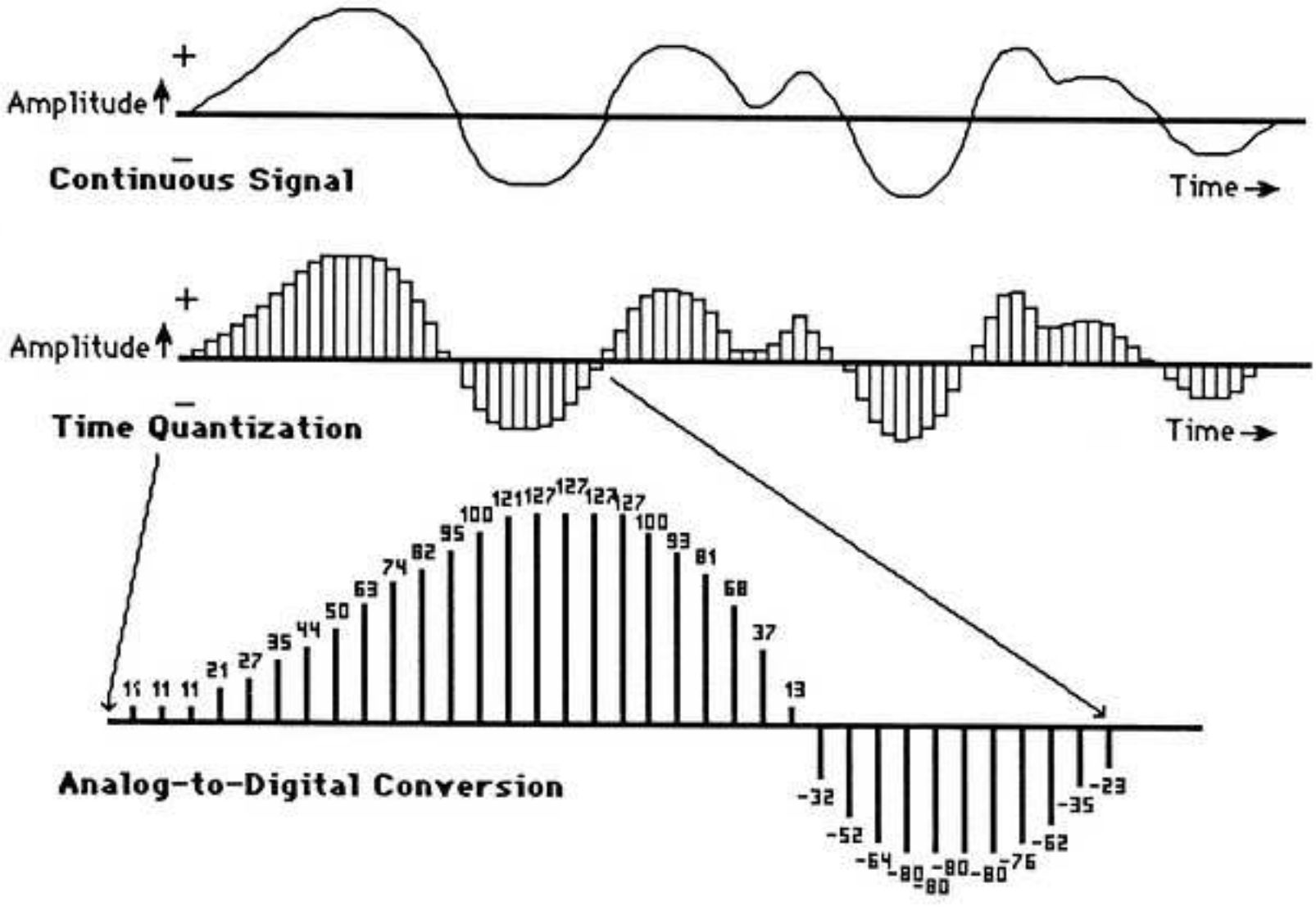
ABOUT DIGITAL SAMPLING

The key feature of the Mirage is its ability to *sample* any sound and play it back on the keyboard. The word 'sample', in this sense, means to record, much like a tape recorder. Anything which provides a standard audio output can be used as a source: a microphone, a record player, a tape recorder, an electric guitar, etc.

The idea of sampling is very simple. Sound is vibration; the exact details of the vibration give each sound its unique character. A phonograph represents these vibrations by a series of hills and valleys inside the groove of a record. Sampling is a way of representing vibrations with a series of numbers.

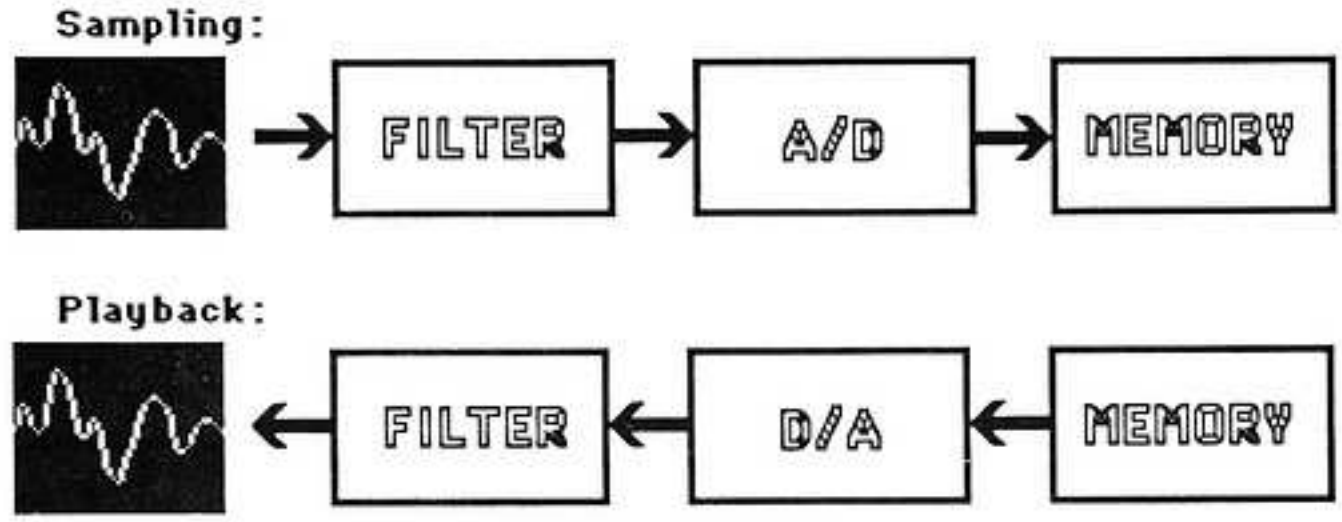
Almost everyone has seen a representation of sound as a squiggly line across a page. Time goes across the page, from left to right, and the 'hills and valleys' are the amplitude of the signal at each particular instant in time.

The whole idea of sampling is that time can be broken up into thousands of tiny steps; this is called *Time Quantization* . If we make these steps smaller and smaller, we eventually get to a point where we can't tell the difference between all of these steps and a smooth, continuous sound waveform.



Each of these time steps is represented by a number. The number is the amplitude of the waveform at that instant. This is accomplished with what is, for us, a black box, called an *analog-to-digital converter* (A/D). The analog signal goes in one end, and numbers come out the other. We can control how fast these numbers come out of the A/D--this is called the *sample rate.* For each step, the A/D produces a number by measuring the current amplitude of the signal; it "samples" it.

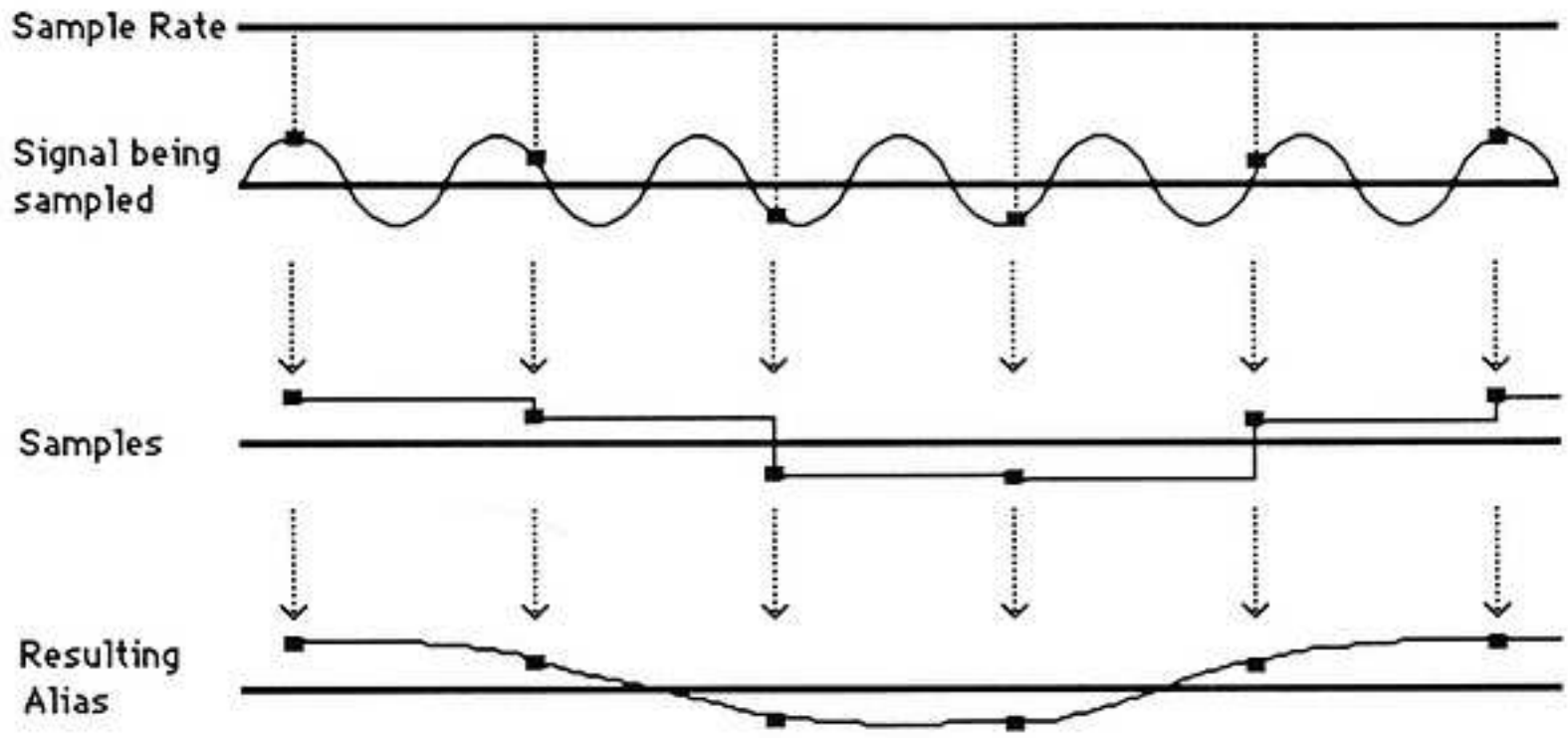
In the Mirage, we store all of these numbers, or samples, in a computer-type digital memory, where we can fetch them easily. Each time a key is pressed, the Mirage reverses the process to play a sound (see the following figure). The numbers are taken from memory, in the same order they went in, and sent to another black box: a *digital-to-analog converter* (D/A) which converts the numbers back into an analog signal. The rate at which these numbers are converted is called the output sample rate. The recreated analog signal is smoothed out with a filter, and becomes sound when it reaches a loudspeaker.



Aliasing

When the signal comes out of the D/A converter, it has square edges because the signal amplitude stays fixed until the next sample comes out, at which time it jumps to that value. These square edges create a roughness to the sound, as they create additional high harmonics which were not in the original signal. This generation of false frequencies is called *aliasing* (one frequency is masquerading as another - an alias). One purpose of the output low-pass filter is to remove these aliases. It does this by smoothing the sharp edges created by the D/A converter.

Aliasing can also occur at the input when a sound is being sampled. This happens when the sound contains frequencies which are higher than one-half the sample rate. To prevent input aliasing, the input signal must also be filtered. If a high frequency creeps in, it gets converted into a low frequency automatically and cannot be seperated from the real signal. This is illustrated graphically in the next figure.



Sampling too slowly causes high frequencies to be misrepresented as low frequencies!

A common example of aliasing occurs in motion pictures; specifically Westerns. Movies are filmed and projected at 24 frames per second; which means that every 1/24 of a second the camera "sampled" the scene. During projection, our eyes act as filters by blurring one frame into the next, giving the illusion of continuous motion. When a wagon wheel starts to turn, the camera has no problem capturing it in enough different positions so that we see it turning. As the wagon starts to speed up, we see a curious thing. The wheels seem to slow down, stop, and then go backward. This is aliasing; we are sampling one frequency and getting a different one on film. Suppose, for example, there are twenty four spokes in the wheel. When the wheel is going around exactly once per second, each frame will show a different spoke in the twelve o'clock position. To the untrained eye, of course, it looks like the same spoke, still in the same position: it appears as if the wheel isn't moving at all.

Sampling in the Mirage

The Mirage is designed to make sampling easy for the novice, yet provide great flexibility for the advanced user. One simple software switch, parameter 77, **User Multisampling**, opens the door to a world of creative sampling.

Normally, with the Multisampling switch set to OFF, the Mirage will use a set of default parameter settings when sampling. In this case, sampling a sound is as easy as plugging in a sound source and pressing two buttons: **SAMPLE** and **ENTER**. When the Multisampling switch is ON, however, the defaults can be altered: the Mirage will sample where, when, and how you want it to. You can start with the defaults and try changing one thing at a time: sample rate, sample length, loops, tuning, keyboard divisions, etc. Rather than assume that it knows the best way to do things, the Mirage gives the musician complete control over every detail of the sound.

THE MASOS DISK

The *Operating System* is the computer program that controls the Mirage from the inside. It determines what happens when any key or button is pressed on the Mirage. One of the advantages of the Mirage is that the Operating System is stored on diskette along with the sounds. The Operating System is loaded, or *booted* when the machine is first turned on (The verb "to boot" is computer-eze for "to load the very first program". It comes from the paradoxical aphorism of "lifting one's self up by one's bootstraps".). This means that the diskette you boot from determines what features the Mirage will have. Normally, you should boot from your sound diskette with the highest version number written along the bottom of the label, e.g. "**Mirage OS Version** *n.n* ".

The *Mirage Advanced Sampling Operating System* (MASOS) is an alternative master disk for the Mirage. To use MASOS, the machine must boot up with the MASOS disk. (That means that the MASOS disk should be the first disk inserted in the machine after you have turned the Mirage on).

The sequencer will not work when you are using MASOS; instead the program implements new features which are useful to the advanced sampler. Any sounds developed using MASOS are completely compatible with the normal Mirage Operating System, (which does have the sequencer). Therefore you can create sounds with the MASOS disk and store them on standard Mirage Formatted Diskettes.

MASOS will enable you to:

- copy a wavesample to another part of memory, upper or lower
- copy any section of memory
- fade in or fade out any waveform
- scale the amplitude of a waveform with a smooth ramp
- add two waveforms together
- invert, reverse, and replicate any waveform
- rotate a wavesample by any number of samples in one operation

The operation of these new features is described in Appendix B. MASOS retains all the features of the standard Mirage Operating System except the sequencer. Most of the examples in this book do not require MASOS; those that do will remind you of this fact.

MASOS is required in order to use the optional ***ENSONIQ Input Sampling Filter,*** *which provides sample rates up to 50 Khz and a 7 pole input filter for enhanced sampling capabilities. MASOS also provides additional* ***MIDI*** *features. These allow a personal computer to access wavesamples and change individual program parameters .* ***ENSONIQ's*** *Apple II *-based* ***Visual Editing System*** *utilizes these System Exclusive* ***MIDI*** *commands to create a powerful digital sound development system.*

PART II - MIRAGE PARAMETERS

The *Mirage Musician's Manual* explains how to execute commands and change parameters on the Mirage. There are several different kinds of parameters: those contained in programs, those affecting wavesamples, and global parameters. All of these are defined in the reference section in the back of the Musician's Manual. This chapter is an in-depth look at certain parameters and how they relate to one another; particularly those needed by the advanced sampler.

Throughout this book, parameters will be referenced by their parameter number in square brackets and, where appropriate, their name; e.g. **[36] Filter Cutoff Frequency**. *The number in brackets is the number used to access that parameter on the Mirage front panel.*

In Mirage terminology, a *sound* is a set of four Programs and eight Wavesamples. There is always an upper and a lower sound present in the machine. It is sometimes misleading to refer to upper and lower keyboard, since certain lower sounds can cover the entire keyboard. Nevertheless, the upper and lower sounds are usually each available on half the keyboard.

When dealing with parameters, one thing to keep in mind is which set of parameters you are working with. First, are you currently on upper or lower? Within that half, you can modify only the currently selected program and the currently selected wavesample. Pressing the **0/PROG** key shows the current half and program; the value of **[26] Wavesample Select** is the current wavesample.

Now let's look at the parameters that are most important to the advanced sampler. We will continue to use square brackets to show the number of a parameter when it is mentioned. For example, **[26] Wavesample Select**.

SELECTING WAVESAMPLES

Select Upper Wavesample	**SEQ REC**	(MASOS only)
Select Lower Wavesample	**SEQ PLAY**	(MASOS only)

When using MASOS, the Sequencer **REC** and **PLAY** buttons have new functions. It is frequently necessary to switch between wavesamples when sampling. To speed this task, the **REC** button becomes a **Select Upper Wavesample** key and the **PLAY** button becomes **Select Lower Wavesample**. Pressing one of these buttons, followed by a number key between **1** and **8**, will select that wavesample. For example: if you are on lower wavesample #2, pressing **Select Upper Wavesample** (**REC**), followed by the **8** key, selects upper keyboard wavesample #8. To accomplish this with the standard (non-MASOS) operating system, requires more effort: press **0** twice to switch to upper, hit **CANCEL**, select **[26]** and **VALUE** to view the current upper wavesample number, then use the "up-arrow " key to push the value up to eight.

OSCILLATOR CONTROLS

[28] Mix Mode

Each time a key is pressed, two oscillators start playing. Each oscillator is assigned a wavesample. In Chorus Mode (**[28]** off), they are both assigned the same wavesample. When Mix Mode is on, however, they play two different wavesamples. Oscillator 2 will always get the next higher numbered wavesample. If Oscillator 1 uses wavesample #5, then Oscillator 2 gets wavesample #6, etc. In either mode, there are still two oscillators sounding, and the other mix parameters will control their relative volumes. When the mix **[34]** is set to 0, we hear only Oscillator 1; when it is set to 63, we hear only Oscillator 2. In the middle, around 32, we hear both.

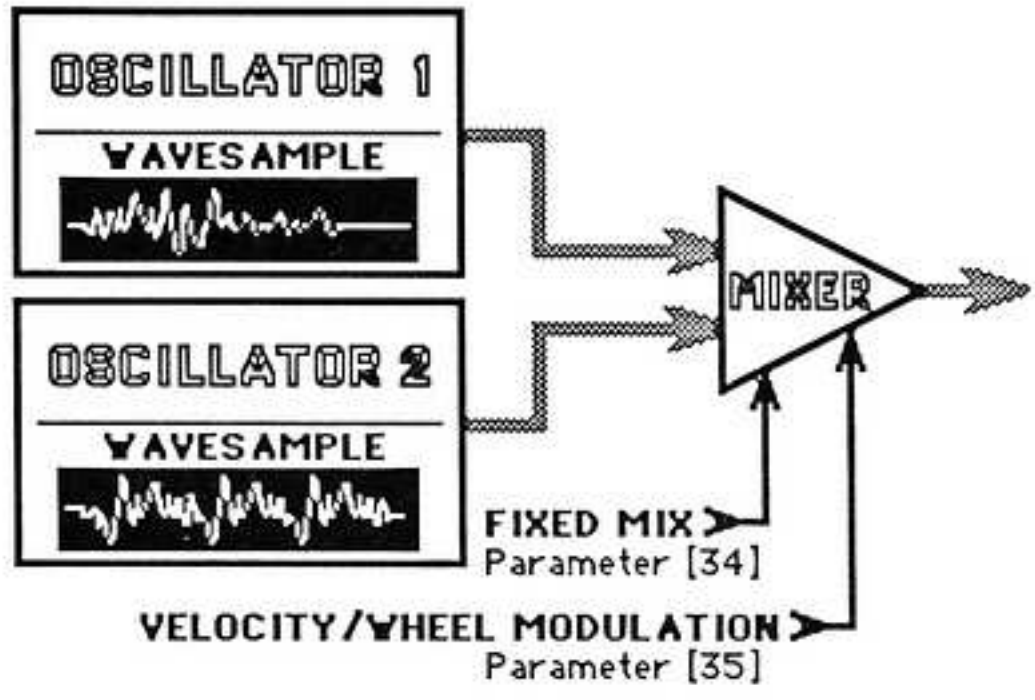
In Mix Mode, a voice will use two wavesamples, but not all the parameters of the second wavesample affect the sound. The second wavesample's parameter settings for **[67]** and **[68]** the **Relative Tuning** parameters, **[69] Relative Amplitude**, **[70] Relative Filter Freq** and **[72] Top Key** are ignored and the settings for the first wavesample are used for both.

[33] Osc 2 Detune

The detune parameter always raises the pitch of the second oscillator. Lower values create a chorusing or phasing effect. The effect can only be heard when the mix **[34]** is near the middle (around 32).

[34] Osc Mix
[35] Osc Mix Velocity Sensitivity

Osc Mix **[34]** and Mix Velocity Sens. **[35]** work together. A fixed mix between the oscillators is set by **[34]** and any modulation from **[35]** is added to this (see the following figure). In the simplest configuration, **[35]** is set to 1, giving no modulation. **[34]** will now be the only control of the mix between the two oscillators. If **[35]** is increased, it will allow the velocity to bias the mix toward the second oscillator. This modulation occurs in addition to the fixed mix amount set by **[34]**. The one special case is when **[35]** is set to 0. Now **[34]** is ignored and the mix is entirely controlled by the mod wheel.



There are four basic ways to use the mix controls, as shown in the next figure.

1) In the single oscillator mode, the mix **[34]** is set to 0, so that only Oscillator 1 will sound. Mix velocity sensitivity **[35]** is set to 1, giving no velocity sensitivity.

2) In chorusing mode, the mix **[34]** between the oscillators controls the amount of chorusing; the maximum effect is heard when it is in the middle, a value of 32. The proper setting of detune **[33]** will depend on the octave of the sound and the effect desired. Very low values produce more of a "flanging" effect.

3) In velocity mixing, the mix **[34]** will set the balance for a note with minimum velocity. Higher velocities will send the mix towards Oscillator 2. Obviously if the mix **[34]** is already set to 63, the modulation will have no effect. The mix velocity sensitivity can be adjusted to give the desired amount of mix for the highest velocity note.

4) In mod wheel mixing (**[35]** set to zero), the mix **[34]** has no effect.

An interesting variation on the above uses method 3) or 4) but with Mix Mode **[28]** turned off. In this case the chorusing of the two oscillators will only be heard when a key is hit hard or when the mod wheel is turned.

MODE	FUNCTION	MIX-MODE Param [28]	DETUNE Param [33]	MIX Param [34]	MIX VEL MOD Param [35]
1	Single Oscillator	OFF	---	0	1
2	Chorusing	OFF	1-20	32	1
3	Velocity Mixing	ON	0	0	1-31
4	Wheel Mixing	ON	0	---	0

KEYBOARD WAVESAMPLE ASSIGNMENT

It is not always easy to determine which wavesample is being played by a particular key. This is because each wavesample has some say in how it is used, and it may be necessary to look at several wavesamples to get the whole story.

The basic scheme is that the wavesamples are used consecutively, and each wavesample controls the highest key that it will use. This arrangement allows lower numbered wavesamples to "hide" higher numbered ones. This can cause confusion, so take a few minutes to understand how the Mirage picks a wavesample for each key.

[72] Top Key (of each wavesample)

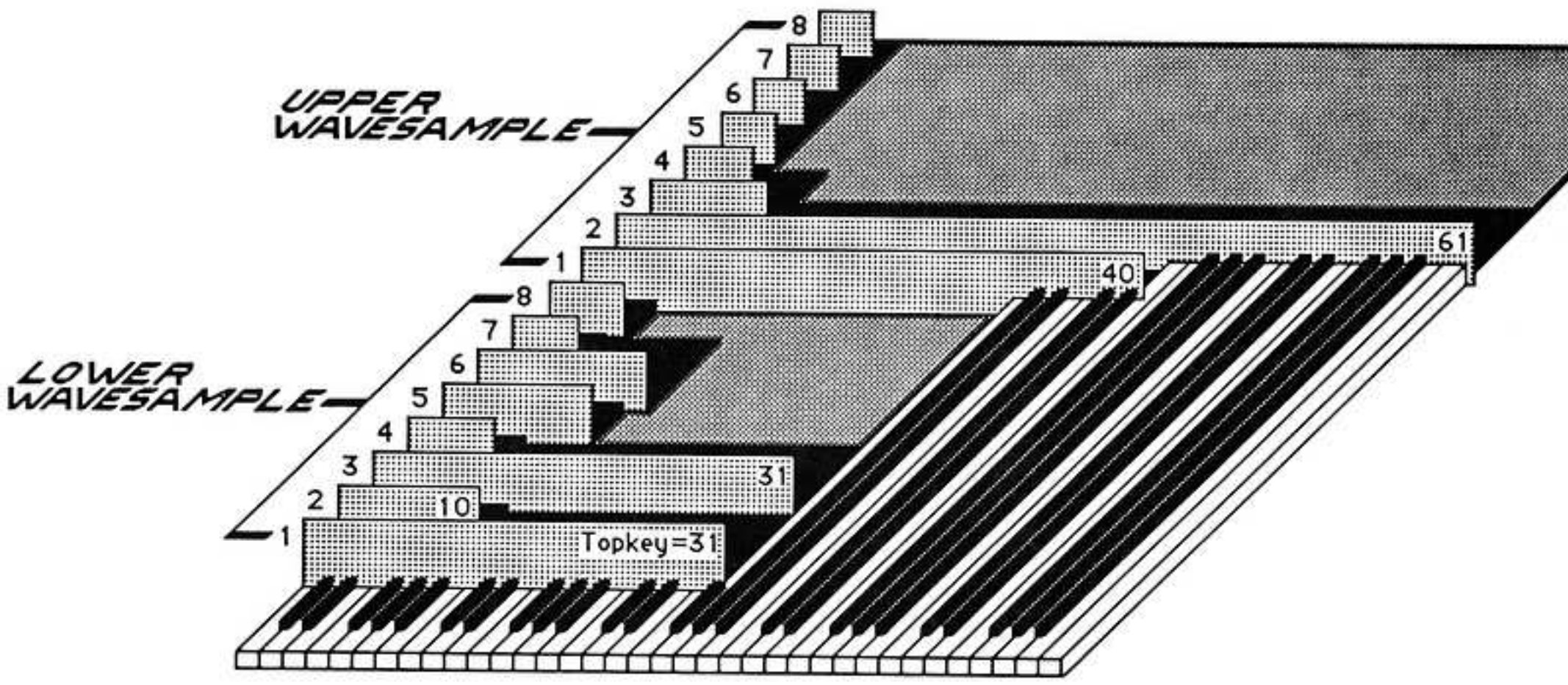
Top Key **[72]** is the most important parameter in the keyboard assignment. There is a Top Key for each of the 16 wavesamples--this is the number of the highest key claimed by that wavesample. Each wavesample tries to claim <u>all</u> the keys from the lowest one on the keyboard up to its Top Key value. Each key normally gets assigned to only one wavesample. The wavesamples are considered in order, and the first one to claim a key 'wins' it. That means if the first wavesample claims all 61 keys, then it gets them all, regardless of the other wavesamples.

This is the order in which wavesamples are assigned keys:

first: lower wavesamples 1 through 8
then: upper wavesamples 1 through 8

The lower wavesamples take precedence over the upper wavesamples. On each half, upper and lower, the wavesamples are taken in numerical order.

As an example of how keyboard assignment works, see the following figure. Lower wavesample #1 has a Top Key of 31. Since it gets first pick, it claims all the keys from 1 to 31. Wavesample # 2 has a Top Key of 10, so it will be hidden by #1. The Mirage then looks at wavesample #3 for a Top Key greater than 31. Since it is not greater (it is also 31), #3 will also be ignored. All 8 of the lower wavesamples will be searched, looking for one that wants to play key 32. When all the lower wavesamples are assigned (or ignored), the same process begins for the upper keyboard. Upper wavesample #1 has a Top Key of 40, so keys 32 through 40 will use it. Wavesample #2 has a top-key of 61, so keys 40 through 61 use #2. At this point it really doesn't matter what the rest of the Top Keys are; they can't be higher than 61 as this is the highest key on the keyboard.



Split Point

Attempting to change the upper/lower split point can cause some confusion. In the above example, we would probably be able to hear that the split occurs between keys 31 and 32. Suppose we wanted the split to be lower, at, say, key 25. Looking at wavesample #1, we see it has a Top Key of 31, so we bring it down to 25. What we didn't realize is, lurking behind #1 was #3, also with a Top Key of 31. Instead of hearing the upper keyboard sound, keys 26, 27, 28, 29, 30 and 31 have a sound we may have never heard before. Don't be dismayed; it's only Lower wavesample #3 with whatever parameters were left in it when it was last used. Lowering wavesample #3's Top Key to 25 or less will solve the problem.

The key assignment is complicated by two more parameters: **[28] Mix Mode** and **[27] Initial Wavesample** . They both affect which wavesamples will be allowed to claim keys on the keyboard when the keyboard wavesample assignment is made.

[28] Mix Mode - Effect on keyboard assigment

As we have seen, in mix mode the two oscillators play different wavesamples. In this case the wavesamples are assigned in pairs. Oscillator 2 always uses the wavesample of Oscillator 1 plus one. The Top Key of Oscillator 2's wavesample is ignored. The Mirage will gang the wavesamples together in pairs; 1 and 2, 3 and 4, etc. It will only look at the Top Key of every other wavesample: 1, 3, 5, and 7. In short: if Mix Mode **[28]** is on, then wavesamples are assigned in pairs, and only every other Top Key is considered.

[27] Initial Wavesample

The Initial Wavesample **[27]** is a Program parameter. It is the first wavesample to be considered in the wavesample assignment process. In the above example, we assumed that it was set to 1 for both the lower and upper keyboard. If we set Initial Wavesample **[27]** to 3 on the lower keyboard, then the Mirage would look first at Lower wavesample #3, see that it had a Top Key of 31, and assign all the keys from 1 to 31 to wavesample #3. From there it would go on to look at wavesample #4, #5, etc. Wavesample 1 and 2 would be ignored, no matter what their Top Key was. Wavesamples with a number below the value of the initial wavesample parameter **[27]** (on each half) are not considered for keyboard allocation.

This parameter allows the four programs to switch between completely different wavesamples, allowing completely different sounds for each program. For example, lower programs 1, 2, 3, and 4 could be set to have an initial wavesample of 1, 2, 3, and 4, respectively. If the Top Key of each of those wavesamples is 31, they will all try to claim the first 31 keys. On program 1, wavesample #1 gets first dibs, and claims all 31 keys; wavesample #2, #3, and #4 are hidden. On program 2, wavesample #2 will get the same 31 keys instead. Meanwhile wavesample #3 and #4 will be hidden. Each program will have a different wavesample for keys 1 through 31.

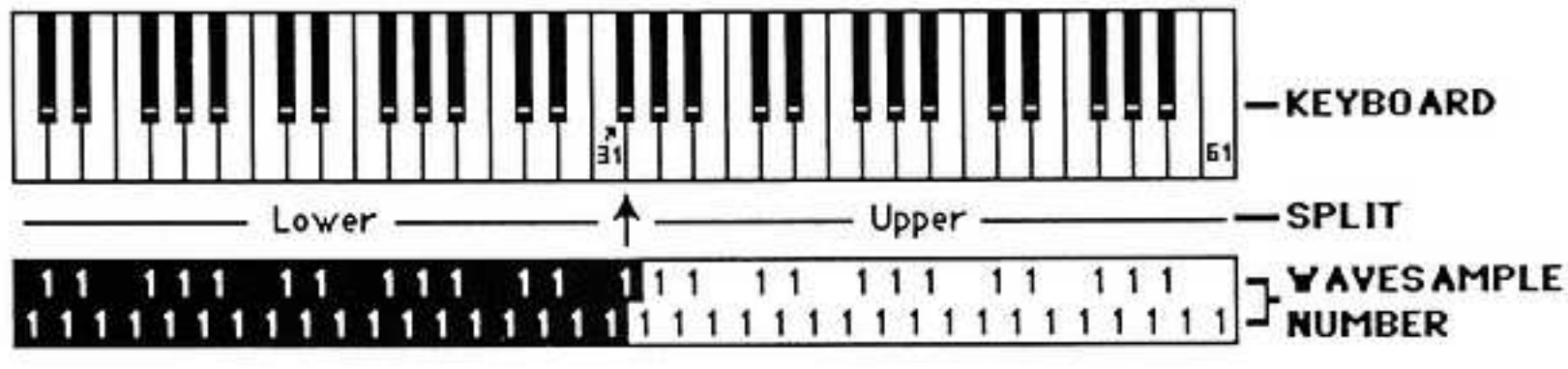
For added fun, those four wavesamples could have <u>different</u> Top Key values, so the split point between upper and lower could vary with the program also.

There are four basic types of keyboard wavesample assignments:

1) Single Wavesample (per half)

This is the simplest arrangement, with a single wavesample on each half of the keyboard. This is the default assignment; It occurs automatically when a sound is sampled and Multisampling **[77]** is off.

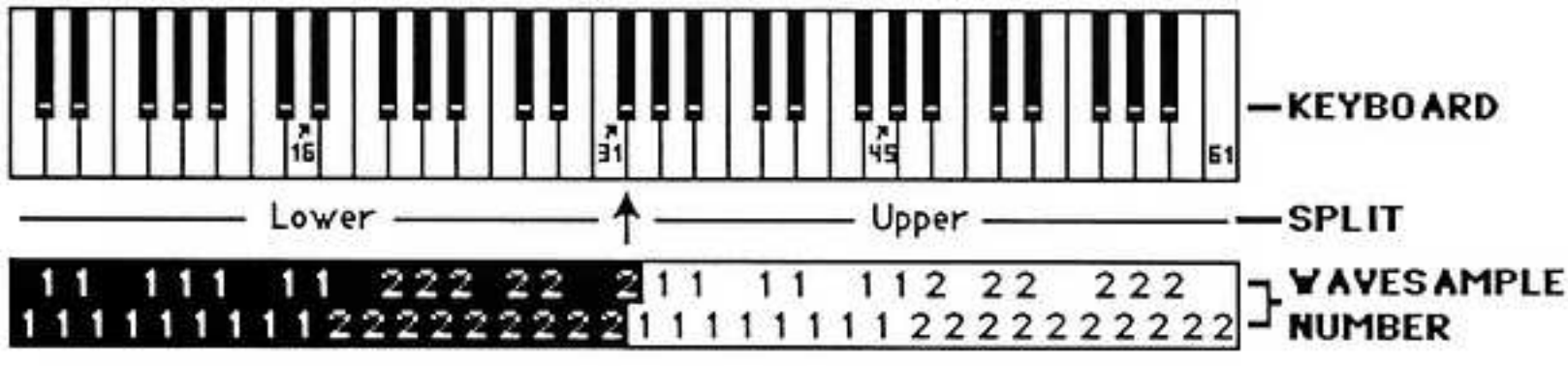
		Lower	Upper
[27]	**Initial Wavesample**	1	1
[28]	**Mix Mode**	OFF	OFF
[72]	**Top Key** - wavesample #1	31	61



2) Multisample

There are a large number of variations on this, depending on how many different wavesamples are desired across the keyboard.

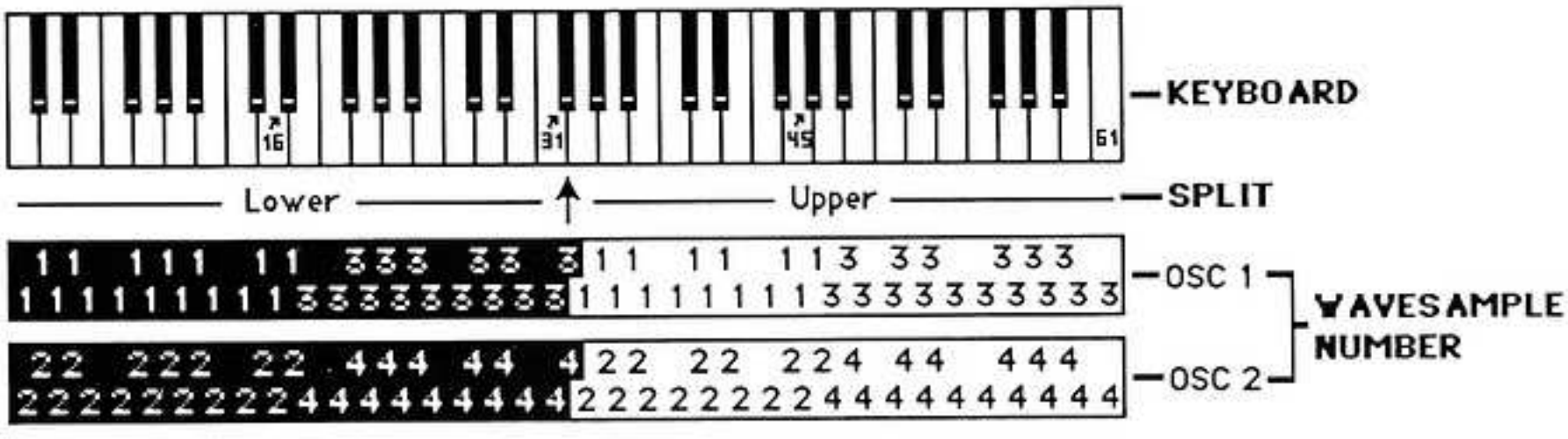
		Lower	Upper
[27]	**Initial Wavesample**	1	1
[28]	**Mix Mode**	OFF	OFF
[72]	**Top Key** - wavesample #1	16	45
[72]	**Top Key** - wavesample #2	31	61



3) Mix Mode Multisample

In mix mode, each key is assigned two wavesamples.

		Lower	Upper
[27]	**Initial Wavesample**	1	1
[28]	**Mix Mode**	ON	ON
[72]	**Top Key** - wavesample #1	16	45
[72]	**Top Key** - wavesample #2	XX	XX (= don't care)
[72]	**Top Key** - wavesample #3	31	61
[72]	**Top Key** - wavesample #4	XX	XX

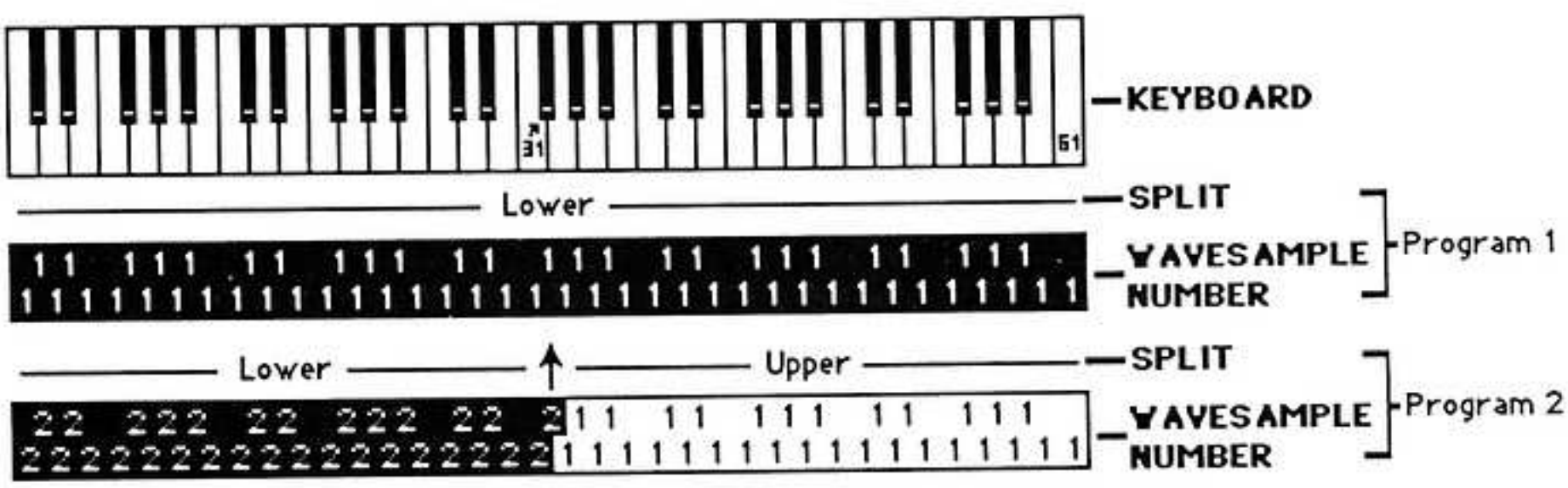


4) Program Variable Assignment

This method takes advantage of Initial Wavesample **[27]**, which allows different wavesample assignments for different Programs.

In this example, lower program 1 dominates the entire keyboard. Lower program 2 only takes half the keyboard, allowing the upper sound to have the top half. Lower program 1 starts with wavesample #1, which goes all the way up to key 61. Lower program #2 has an initial wavesample of 2, so wavesample #1 is ignored. Wavesample #2 only goes up to key 31, and so the upper sound is 'revealed'.

		Lower	Upper
[27]	**Init Wavesample** - Prog 1	1	1
[72]	**Top Key** - wavesample #1	61	61
[27]	**Init Wavesample** - Prog 2	2	1
[72]	**Top Key** - wavesample #2	31	61



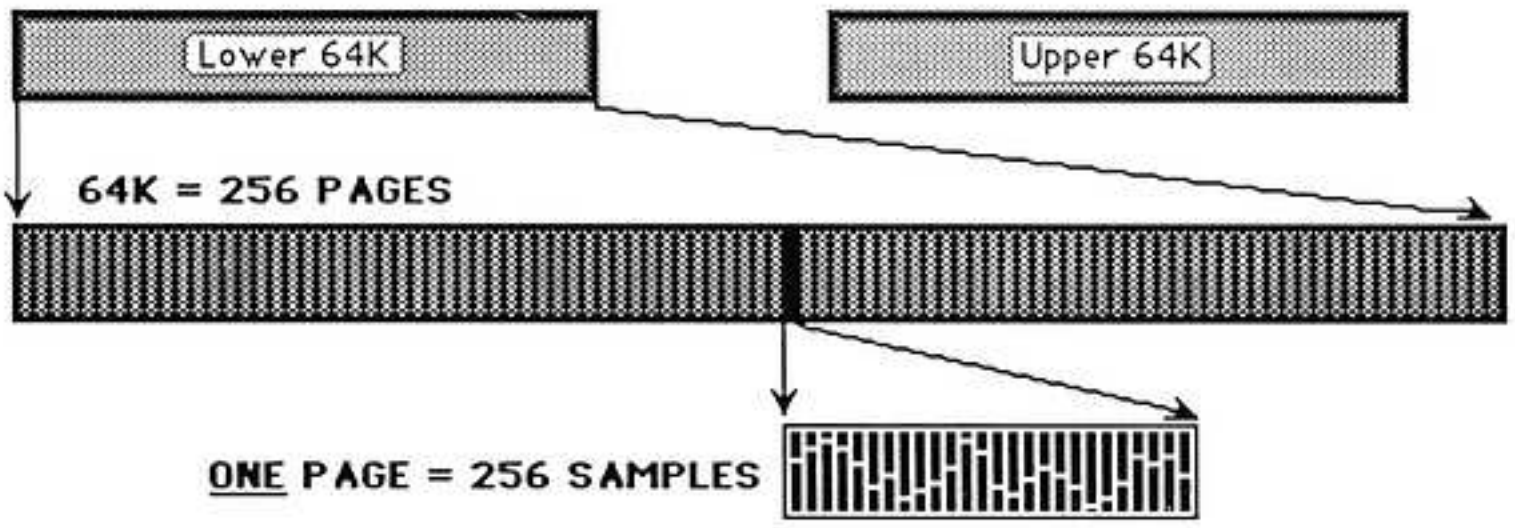
MEMORY ALLOCATION

Each Mirage sound, both upper and lower, has 65,536 bytes of memory to hold the wavesamples. In computer lingo, that number is also called *64K*, which rolls off the tongue a bit better. This 64K bytes is broken into 256 equal units in the Mirage, called *Pages*. A page, in turn, is made up of 256 bytes, or samples. And yes, if you multiply 256 by 256, you get 65,536.

All of this memory is divided up among the eight wavesamples in each sound. The allocation of memory to each wavesample has nothing to do with the assignment of wavesamples to keys on the keyboard. For example, Wavesample #1 could be as short as 2 pages, but be active over the entire keyboard. Wavesample #8 could be a full 256 pages, but only play on one key (not very useful, but possible). A common memory allocation is one where two wavesamples are active and each one gets half of the 64K of memory.

Note that in the Mirage, we count pages in the base-sixteen number system, called *Hexadecimal* (see the Appendix). This turns out to be quite handy for dealing with things that have 256 divisions, like computer memory, but it takes getting used to.

Memory can be thought of as a strip 256 units long. To do multi-sampling, we have to decide how to divide up this strip. For each wavesample we want to use, we must devote a length of memory to it.



[60] Wavesample Start
[61] Wavesample End

To allocate memory to a wavesample, we simply set up the Wavesample Start **[60]** and Wavesample End **[61]** parameters. Wavesample End tells us what the last page of the sample is. If a wavesample starts on page 1 and ends on page 2, then it is exactly two (not one) page long. Likewise if it starts on page 0 and ends on page 2, it is 3 pages long.

For simplicity, it is a good idea to have wavesamples start on nice even Hexadecimal boundaries, like 40, 80, or C0. To get the maximum use of memory, end the wavesample one page before the start of the next wavesample in memory. For example, if wavesample #1 starts at 00 and another wavesample starts at 40, then set the end **[61]** of the first wavesample at 3F.

It is possible, but sometimes dangerous, to allocate the same memory to two different wavesamples, or to overlap wavesamples. The Mirage does not check, for example, that wavesample #1 ends at 40, and will allow you to move the start of wavesample #2 to a number less than 40. If a loop marker is present in either wavesample, strange things can happen (see **Sample Data Format** in Part IV).

The wavesample start and end parameters tell the Mirage where in memory to record the new sound when you sample. They also tell it what section of memory to play when a key goes down. If you move Wavesample End **[61]** after you have sampled a sound, it will not truncate the end of the sound unless you turn the Loop Switch **[65]** on and then off. This inserts an "end of sound" marker in the memory.

LOOPING

[62] Loop Start
[63] Loop End
[64] Loop End Fine Adj.
[65] Loop Switch

Many musical instruments are able to sustain notes over a long period of time. In contrast, a drum sound will always die away quickly. In the Mirage, sustained sounds are usually *Looped*. This means that after the initial attack, a portion of the sound is played over and over until the key is lifted and the sound dies away. This conserves memory, allowing many different wavesamples in the machine at one time.

The Loop pointers - Loop Start, Loop End and Loop End Fine Adjust - determine where in the sound (wavesample) the loop will occur. As with Wavesample Start and Wavesample End, these parameters are in Hexadecimal. Loop Start is the first page of the loop, Loop End is the last. When Loop Start equals Loop End, the loop is one page long. Loop End Fine Adjust is slightly different; it selects which sample (byte), of the 256 in the last page of the loop, will be the last sample. It is normally set to FF, that is, the last sample in the page selected by Loop End. Note that for certain values of Loop End **[63]**, Loop End Fine Adjust **[64]** will not go below 80. This is normal.

[65], the Loop Switch, can be on or off. When off, the sound will play once, from Wavesample Start to Wavesample End, and then stop. When on, the sound will play from Wavesamples Start to Loop End; then begin again at Loop Start and continue playing to Loop End over and over. The amplitude envelope will determine when the sound stops after the you release a key.

The loop switch serves one other function, which has to do with *truncating* sounds. If you have a sound which is not looped, then all of the sound will be played out to the Wavesample End. If you try to make it shorter by moving Wavesample End, the sound will not obey until you turn the Loop Switch on and off again.

*In **Part 4 - Advanced Sampling Techniques**, further details of looping are covered.*

RELATIVE TUNING, AMPLITUDE AND FILTER PARAMETERS

[67] Relative Tuning Coarse (octaves)
[68] Relative Tuning Fine (1/256 octave steps)

The Mirage changes the playback rate of a wavesample as you play different keys. The resultant sound may or may not have a pitch, and if it does, the Mirage does not "know" what it is. For example, if you sang a major scale, starting on C, and sampled it into the upper keyboard of the Mirage, hitting one key will play back the whole scale, made up of eight different pitches. The Mirage will transpose that sound up or down, depending on which key you hit. The Mirage does not know what pitch, or group of pitches, are in its memory; all it can do is blindly shift them up or down.

When you press a key and the sound comes out at the same pitch as the original sound, we say that this is *Unity* playback rate. It means that the samples are coming out at the same speed they went in. Any note below unity rate plays the sound slower (therefore lower in pitch) and any note above unity rate plays faster (and higher).

Note that Unity is when the playback (output) sample rate matches the input sample rate. The default input sample rate for the Mirage is 29.4 KHz. The middle A key of the Mirage upper keyboard produces an output (playback) sample rate of 29.4 Khz. If we sample a sound at the default rate, (and the User Multisampling Switch **[77]** is off) then Unity playback rate occurs when we play the A key. If we change the input sample rate, the A key will no longer produce Unity playback rate. For example, if we sampled our little C scale with a sample rate of 25 Khz (Sample Time **[73]** = 40) then playing it back at 29.4 Khz would cause it to be faster, and two half steps higher in pitch. To play it back at unity we must hit the G key in the middle of the upper keyboard.

The important concept is that the Mirage can only control relative pitch. It is up to the user to adjust each sample into concert pitch. Suppose we had sampled a C major scale at the default sample rate. When we play the A key we hear a C scale. When we play the C three keys above A, the sound is transposed up a minor third, so we hear an E-flat scale. Now we want to tune the wavesample so that when we play the C key, we hear a C scale. Here is where **[67]** and **[68]**, Relative Coarse and Fine Tune, come into play. We simply bring the fine tune parameter down until the sample comes into tune on the C key (Note that you must restrike the key in order to hear the new setting). Use a known-to-be-in-tune sound on the opposite half of the keyboard to compare (If

you prefer to play with numbers, you could, instead, calculate how much to change it; a minor third is one quarter of an octave,which is 64 fine-tune steps (256/4). In Hex that's 40, so move it from 84 down to 44).

Remember the final pitch of a sound is determined by several factors: the original pitch of the sound; the input sample rate; the relative tuning of the wavesample; the key pressed on the keyboard; the pitch wheel and any LFO modulation.

As you learn more about sampling you will find that, for sounds with short loops, its often best to leave Relative Tuning Fine set at 80, and adjust the input sample rate or the pitch of the sound source to bring things in tune. More on that in the section on sample rates in Part IV.

[69] Relative Amplitude
[70] Relative Filter Freq
[71] Maximum Filter Freq

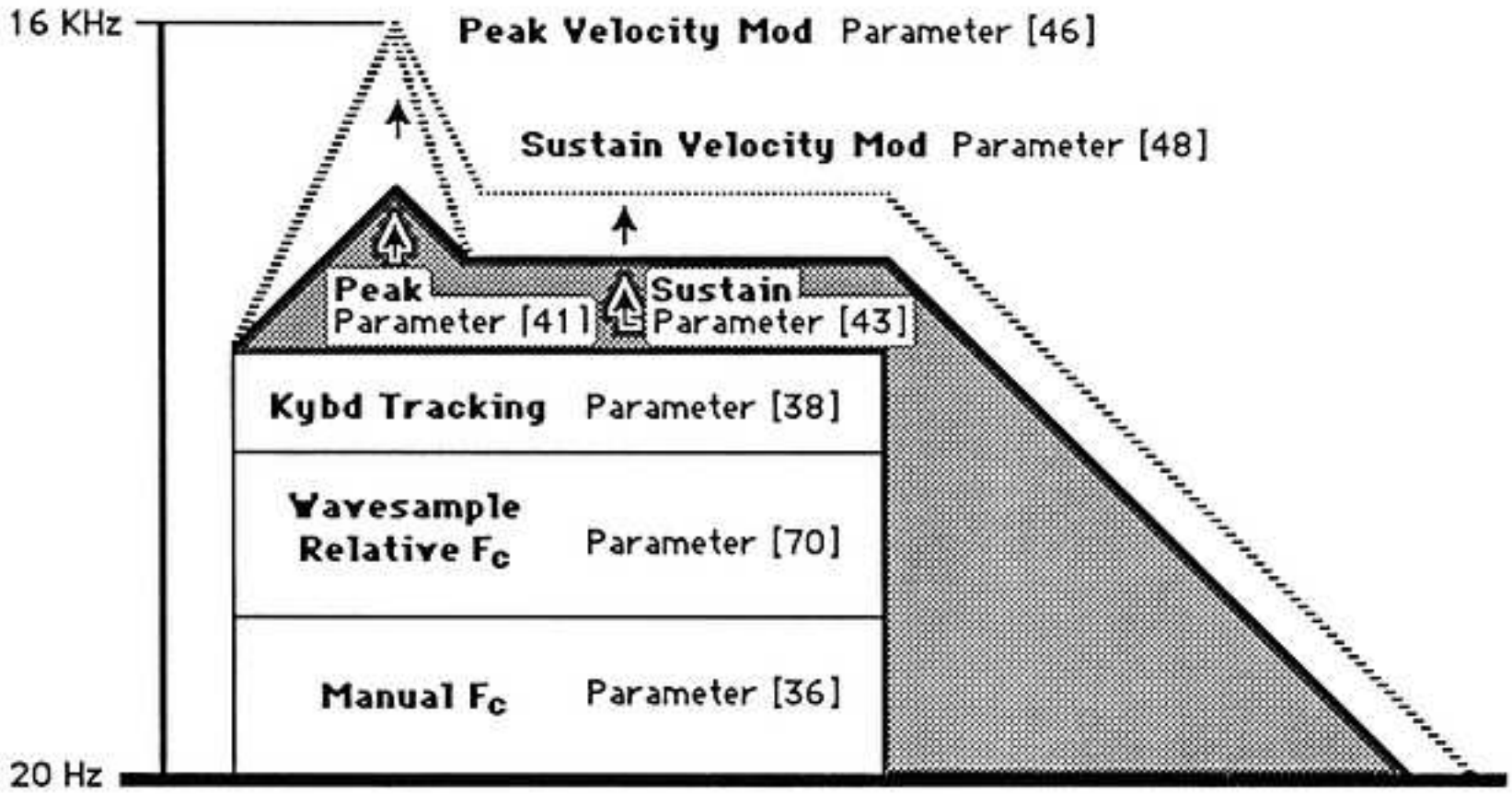
Multisampled sounds are those which have more than one wavesample. Since the program parameters affect all the wavesamples at the same time, we need some way of adjusting individual wavesamples so that they fit-in together. The Relative Amplitude and Filter parameters do just that; they affect only one wavesample.

Relative Amplitude **[69]** is normally set to 63. By bringing it down, you attenuate the selected wavesample with respect to the other wavesamples in that half of the memory.

Relative Filter Frequency **[70]** is normally set to 20. It has the same effect as the program control of filter frequency **[36]**; the two are added together to determine the cutoff frequency of a note. As with **[36]**, it is adjustable in semitone increments.

The Filter cutoff frequency is controlled by many factors (see next figure). The base cutoff frequency for a voice is determined by adding the Program Cutoff Frequency **[36]**, the Wavesample Relative Freq **[70]**, and the Filter Keyboard Tracking **[38]**. From that base frequency, the filter envelope rises, and its height is velocity sensitive. With all of these contributing to the filter cutoff frequency, its hard to know exactly how high the filter may go. Since the Mirage filter has also been charged with the job of keeping out the unwanted high-frequency aliases, it may occasionally be undesirable for it to shoot right up to 16 Khz. **[71] Maximum Filter Frequency** guards against this. It provides a ceiling on the cutoff frequency. No matter what all the modulators add up to, the filter will never go above the Max Freq setting. This parameter is only used if a sound contains undesirable high frequency noises which would become audible if the filter were wide open (for example, if you sample a sound at a rate less than 29.4 Khz, any sound in the output above half the sample rate can only be noise). Often this parameter is just set to its maximum value of 99. Max Freq is in semitone increments.

Note that in Mix Mode, only one set of relative parameters are used; they are applied to both wavesamples. Oscillator 1's wavesample parameters are active; Oscillator 2's are ignored.



The Filter Cutoff Frequency Parameters add to form the actual F_C.

SAMPLING PARAMETERS

[73] Sample Time Adjust

[73] specifies the input sample *time*, which determines the input sample *rate*. Sample rate is the frequency at which we sample. it is a fundamental consideration in all digital music systems. Another measure of the same thing is sample time; that is, the time between samples. Sample time is specified in *Microseconds* (µsec--millionth's of a second), so instead of saying .000034 seconds, we can say 34 µsec. That corresponds to a 29,400 Hz sample rate. Often we use *Kilohertz* (KHz--thousands of Hertz) for frequencies, so we can say 29.4 KHz. Sample rate and sample time are inversely proportional. This means that as sample time increases, sample rate decreases. In the Mirage, we can only specify the sample time; the sample rate follows from this formula:

$$\text{Sample Rate in KHz} = \frac{1{,}000}{\text{sample time in µsec}}$$

for example, when sample time is set to 34:

$$\text{Sample Rate in KHz} = \frac{1{,}000}{34\ \text{µsec}} = 29.4\ \text{KHz}$$

Appendix A lists all sample rates available on the Mirage. Sample rates above 33 Khz are only available when using the ***Input Sampling Filter*** *(available from your dealer) and the MASOS disk.*

[74] Input Filter Frequency

The sampling input to the Mirage passes thrcugh an internal 4-pole filter. This input anti-aliasing filter is a 24 db per octave low-pass filter. Its frequency is adjustable in semitone steps. Appendix A lists the available internal input filter frequency settings. The optimum filter setting may vary for different sounds using the same sample rate.

[93] External Input Filter Frequency (MASOS only)

This parameter has been added for use with the optional external **Input Sampling Filter.** The value set for this parameter will determine the setting of the 7-pole low-pass input filter according to the values listed in Appendix A. When it is set to a value of 0, the Mirage will sample through its internal filter and A/D converter. When set to any other value, the Mirage will sample through the cartridge filter and

A/D. If you are not using the external filter cartridge, you must leave this parameter set to 0 or you will not be able to sample.

[75] Line/Mic Level Input

[75] is the line/mic switch. The Mirage has two input paths which both use the same audio input jack on the rear panel. The line/mic switch contols which path is in use. The Mic level setting (off) increases gain and also puts a compressor on the signal. This is intended to allow fool-proof quick sampling without too much worry about overloading the input. The Line level setting (on) should be used for sampling any electronic instruments, or when connected to the output of a mixer or preamp. This setting is generally better for percussive sounds because the compressor is not in the circuit.

Digital systems are sensitive to amplitude levels in two ways. When the signal is too high, the A/D converter will 'clip', giving hard distortion to the sound. When the signal is too low, it will get buried in quantization noise. The Mirage does not have a variable input level control, so it is advised that the serious sampler use the Line level setting and a mixer or preamp with an output level control on the signal source.

[76] Sampling Threshold

[76] sets the level which the input signal must cross in order to trigger sampling. During level detect mode, the center LED bar will light when the signal crosses the Sampling Threshold. Pressing **ENTER** activates the sampling input and sampling will begin when the signal crosses the threshold.

The threshold level should be adjusted to suit each individual sound so that the sample starts at the desired spot, without trimming off any of the attack transient. When set at zero, the signal is always above the threshold; therefore sampling will begin immediately after pressing the **ENTER** button. Note that in level detect mode, when the threshold is set to zero, the middle bar will always be lit.

[77] User Multsampling Switch

The Mirage makes it easy to sample by using a number of default parameter settings. These parameters are automatically installed whenever a sample is taken with the User Multisampling switch off. When on, the User Multisampling switch prevents the Mirage from installing these parameters, leaving them at the values they were prior to sampling. If you adjust these parameters and wish to retain the new settings, you must turn multisampling on before you sample or all wavesamples will be reset to the defaults. The normal setting for serious sampling is on.

The default parameter values are:

[26]	**Current Wavesample**	1
[27]	**Initial Wavesample**	1
[28]	**Mix Mode**	OFF

and for all eight wavesamples:

[60]	**Wavesample Start**	00 (Start of memory)
[61]	**Wavesample End**	FF (End of memory)
[62]	**Loop Start**	FE
[63]	**Loop End**	FE
[64]	**Loop End Fine Adj**	FF
[65]	**Loop Switch**	OFF
[69]	**Rel Amplitude**	63
[72]	**Topkey**	31 for lower, 61 for upper

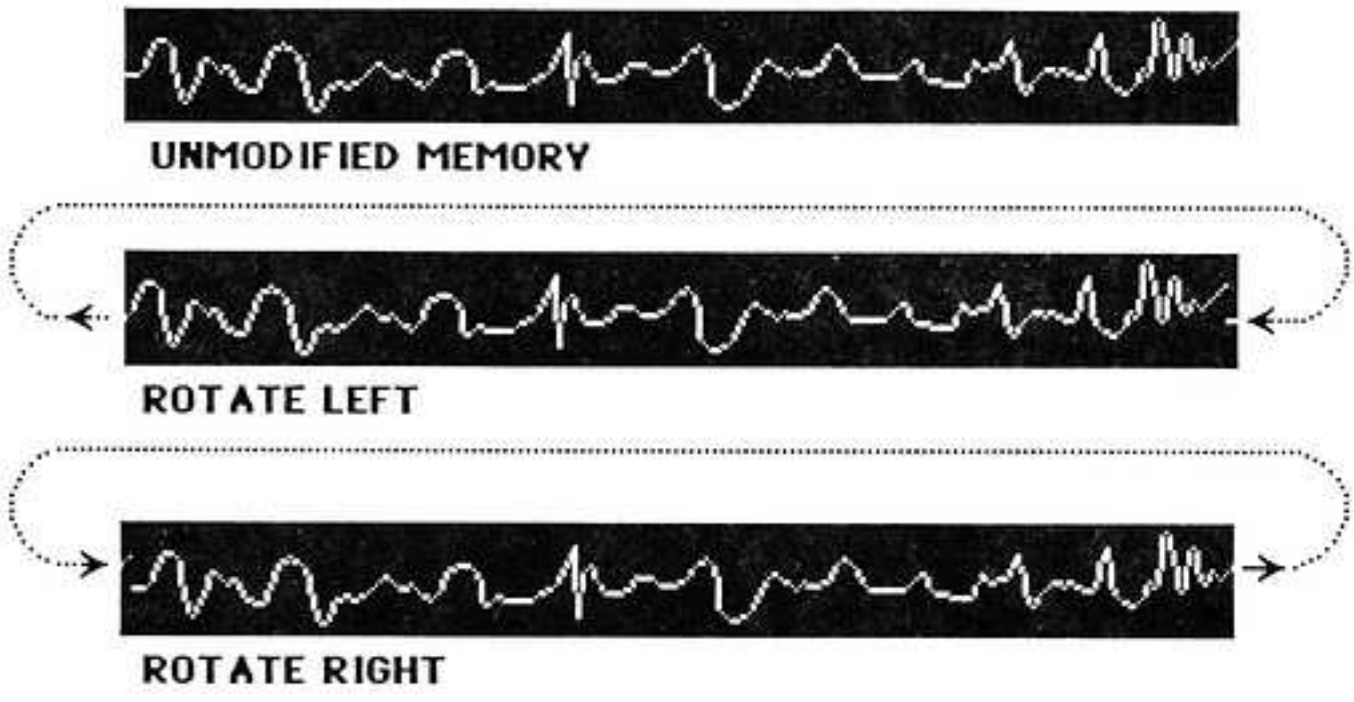
Sampling once with the Multisampling switch off is a quick way to 'reset' the wavesamples before doing multisampling. There are subtle problems that arise when you start working on one wavesample and don't know that some other wavesample claims it has a loop in the same territory. To avoid this, turn **[77]** off and do a quick sample before starting a multisampling project. Then turn **[77]** on so that the parameters will not be reset when you begin sampling for real.

Regardless of the setting of the multisampling switch, the wavesample relative amplitude parameter **[69]** will be reset to it's maximum value (63) each time you sample

WAVESAMPLE OPERATIONS

[66] Wavesample Rotate Command

Rotation is the sliding of the waveform sample data to the right or left with respect to its location in memory. For example, to rotate left by one, the sample that was in location 02 is put in location 01; the sample that was in 03 is placed in 02, and so forth. Finally, the last location is filled with the sample that was in location 01. This is called rotation because the wavesample is treated like a circular buffer; what falls out of one end is put back in the other.



Why would you want to rotate? Loops are very sensitive to the particular sample values at the splice point. Since a loop must always begin on a page boundary, there are usually only a few appropriate places for the loop start point. If the sample at the loop start point doesn't splice cleanly, the particular sample at the start of the page may be changed by rotating, yielding more possibilities for succesful loop splices.

The MASOS rotate command works differently from the standard Mirage operating system. If you booted from a standard sound disk, then the command works as follows:

For each press of the "down arrow" key, the wavesample rotates left by one sample; for each press of the "up arrow" key, the wavesample rotates right by one sample.

The rotation may take up to 8 seconds, depending on how big the wavesample is. The displays will go blank during this time (or leave a segment lit) but don't despair; the Mirage will return when rotation is complete. To help relieve the slowness of the one sample rotate, MASOS provides the ability to rotate by any number of samples, up to 255 (FF Hex). There are two new commands, Rotate Left and Rotate Right, which operate on the current wavesample.

[19] Rotate Current Wavesample Left by (n) (MASOS only)
[20] Rotate Current Wavesample Right by (n) (MASOS only)

The MASOS rotate commands work as follows:

After selecting a rotate command **[19]** or **[20]**, the display will flash "rL" or "rr" for a moment to indicate the direction of rotation, then the number of locations of the previous rotate will appear. This value can be changed using the up and down arrow keys. When the correct value is set, pressing the **ENTER** key will initiate the rotate function. The display will go blank briefly, then will show "dr" indicating that the wavesample has been rotated.

Rotating a sound too far to the left will cut off the attack portion of the sound. You can fix it by rotating it right again, since the data was not discarded but moved to the end of the wavesample.

Rotating a sound too far to the right may cause a gap before the beginning of the sound. Actually, the space in front of the sound is filled with the data from the end of the wavesample. This may or may not be what you had in mind.

[17] Copy Current Wavesample to Lower (n) (MASOS only)
[18] Copy Current Wavesample to Upper (n) (MASOS only)

These commands allow you to copy wavesamples from one half of the Mirage memory to the other. They can be used to transfer a sound on the lower keyboard to the upper, or vice versa. To use the Copy command,verify that the wavesample you would like to copy is currently selected. Next, select the desired Copy command **[17]** or **[18]**. The display will show "U" or "L" and a flashing number. Press the number key corresponding to the wavesample you wish to copy the current wavesample into. Finally, press the **ENTER** key to begin copying. The parameter number will appear after copying is complete.

MASOS DATA MANIPULATION FUNCTIONS

The following functions are available when using MASOS. They provide a way of manipulating wavesample data after it has been sampled, but are not restricted to operating on wavesample boundaries. Since most of these functions will change the samples in memory, it is strongly advised that you save any sound on a diskette before attempting to alter it so you can reload it if you make a mistake. You can always save overtop of this "backup" sound when you are finished.

Each function operates on a section of waveform memory. This section will be called the *Source*. Two of the functions, **Copy** and **Add**, also use a second section of memory, the *Destination*. The Source is determined by the setting of the Source Start and Source End parameters. The Destination will always be the same length as the Source, so it does not have an end parameter; only a Destination Start parameter.

The following parameters are pointers which control the range of memory that will be manipulated. Each pointer is made up of two numbers, which are both displayed in Hexadecimal. The first half of a pointer is the page number; the second half is the sample number on that page.

[85]	**Source Start**: page number	(normally 00)
[86]	**Source Start**: sample number	(normally 00)
[87]	**Source End**: page number	(normally 01)
[88]	**Source End**: sample number	(normally FF)
[89]	**Destination Start**: page number	(normally 00)
[90]	**Destination Start**: sample number	(normally 00)
[94]	**Destination Bank Select**	(normally LO)

[94] Destination Bank Select sets the upper or lower sound memory bank as the place to store the results of a Copy or Add function.

[95]	**Scale Function Start Factor**	(normally 00)
[96]	**Scale Function End Factor**	(normally FF)

These scale factors are used to control the scaling of the sample data by the Scaling Function. The range of both variables is from 00 to FF (hex).

MASOS Function Key (Load Seq key)

The MASOS special functions are accessed by pressing the **MASOS Function** key (which is the **Load Seq** key when not using MASOS), then pressing the number which corresponds to the function you wish to perform. A flashing mnemonic message will appear until you either press the **ENTER** key to initiate the function, or the **CANCEL** key to abort it. When the requested function has been completed, the "Fc" or Function Complete message will appear.

The following table lists the key numbers and their corresponding MASOS functions:

MASOS Functions

Function (key) #	Mnemonic	Description
1	Cd	Copy Data from source to destination
2	Fi	Fade in from source start to source end
3	Fo	Fade out from source start to source end
4	Sc	Scale the source data with a linear ramp function which ramps between scale start factor and scale end factor.
5	Ad	Add source data to destination data; leave result in destination
6	In	Invert the source data
7	rd	Reverse data from source start to end
8	rP	Replicate first page of the source data on each page from source start to source end.

These functions are described in more detail in **PART IV - ADVANCED SAMPLING TECHNIQUES** (page 66) and in Appendix B.

PART III - SAMPLING TUTORIAL

EXAMPLE 1: SAMPLING AN ORCHESTRA "HIT" FROM A RECORD

Recorded music provides a limitless collection of final chords, introductory splashes, heavy hits and quick punches which can be reproduced on the Mirage. These sound "events" don't usually need loops and are quite easy to capture.

For this first example, we'll go through all the details of pushing buttons and setting levels. Later, we'll assume that you know how to initiate sampling and concentrate on the variations possible with multisampling.

1) Load Upper and Lower sound #1 from your MASOS disk.

Unlike the programs on a Sound disk, the sounds on the MASOS disk (or a blank Formatted disk) are "templates" for sampling. Sampling into a sound from a Sound disk (piano, for example) can produce unusual results as your new sampled sound is processed through the envelope, filter and modulation parameters of the previous sound. For this reason, you should always load in a sound from the MASOS disk or a blank Formatted disk before sampling. The parameters are set up in a "plain vanilla" configuration on all Programs, with a quick Attack, full Sustain, quick Release, no chorusing, limited filter modulation and velocity control. When you sample, the Program of the last sound loaded remains in effect for the new sample.

For reference purposes, the values of all parameter settings on the MASOS disk sounds are listed in Appendix E. You can also use a copy of this form for notating your own patches.

2) Set the sampling parameters as follows:

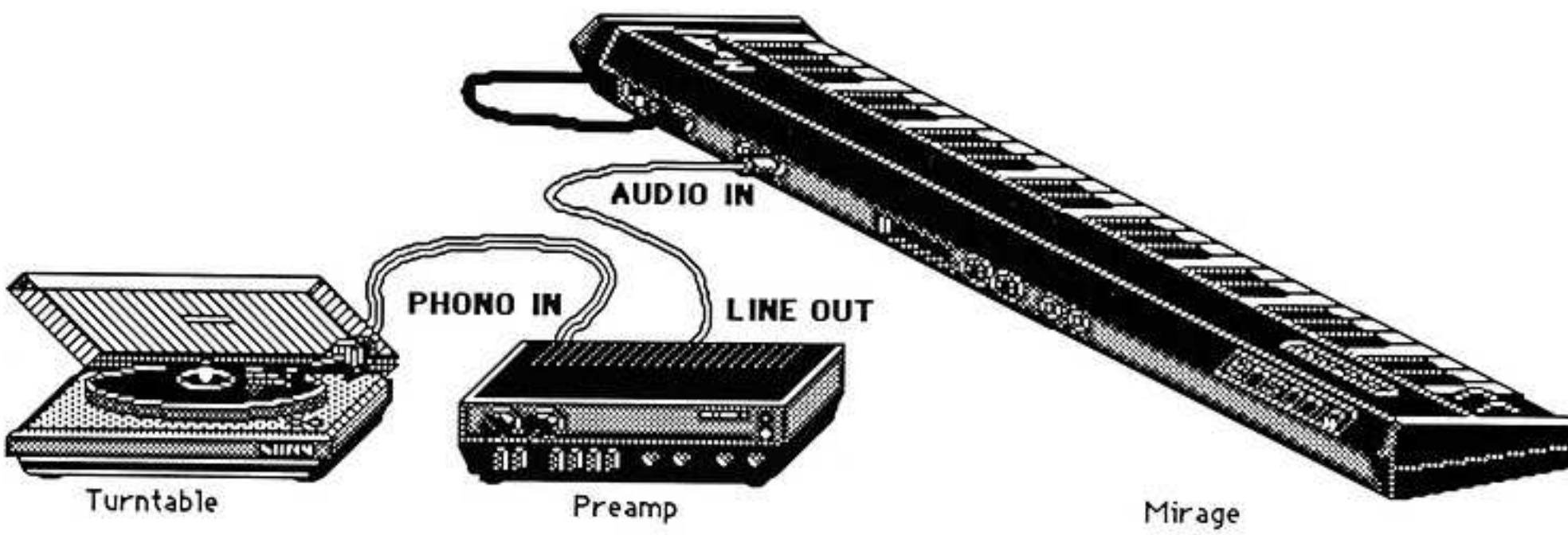
PARAMETER		VALUE	MEANING
[73]	**Sample Time Adj**	34	29.4 KHz sample rate
[74]	**Input Filter Freq**	90	12 KHz cutoff
[75]	**Line/Mic Level Input**	ON	line level
[76]	**Sampling Threshold**	10	trigger at +/- 10
[77]	**User Multisampling**	OFF	default wavesample settings

3) Connect the sound source to the Mirage Audio Input jack on the rear panel.

You will need some way of controlling the output volume of your sound source. If your source is a Phono Turntable, it will need a preamp with a volume control. You should take the signal from the "Line (Main)" output, if there is one. The "Tape Out" jacks on most preamps are not affected by the master volume control or tone controls. If the preamp has a rumble filter, turn it on (the frequency response of the Mirage extends to the sub-audio range and it can easily reproduce record warp noise and motor rumble). Also, put the preamp in Mono, if possible, so the sound from both stereo channels is captured. You can monitor what you are sampling by plugging the remaining preamp output into a power amp, but you will be using the preamp volume control to set the sampling level, which could be excessively loud when amplified, so be sure to use an amp with a separate level control.

Tape Decks are ideal sources as you can easily repeat the sound over and over (if you are a novice sampler, you will almost certainly need a number of "takes" before you get what you want) and tape decks often have an output volume control as well. In fact, if you want to take a sound off of a record, it is advisable to tape the record, then sample the tape. One way to hook up is to take the "Line (Play) Out" of one channel of the tape player directly into the Mirage. The other channel can go to your normal amplifier so you can hear what's on the tape as you sample it (remember, it might be loud!). The headphone output of most hi-fidelity portable tape players should work, but you must have the right connectors to convert the stereo mini-jack output to a mono phone plug (don't short the outputs directly together!).

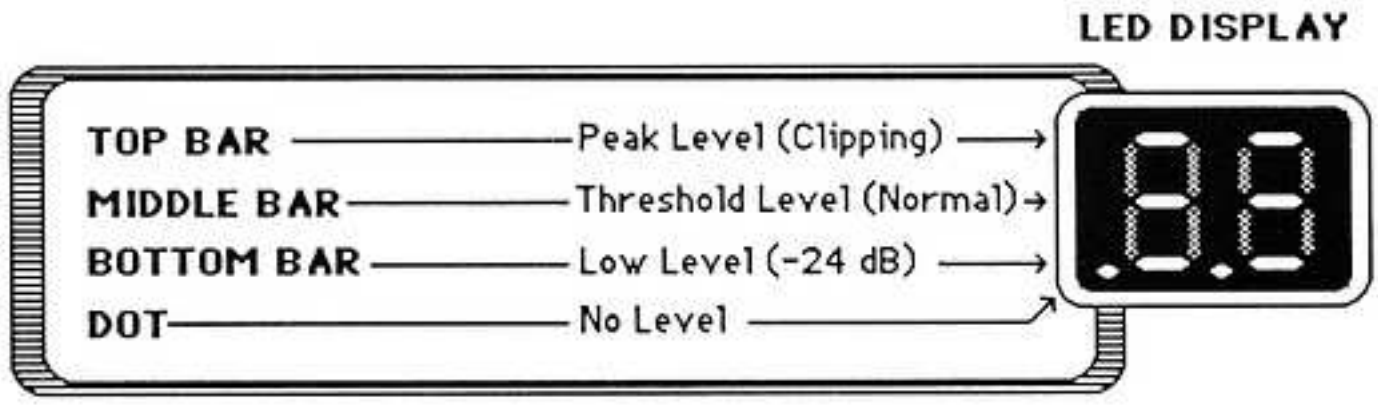
If you are using a microphone, you can use the Mic level setting and plug the unbalanced mic output directly into the Mirage, however you will get better quality by running the mic through a mixer first. The mixer gives you tone controls, a low-impedence, low-noise mic preamp, more gain, a volume control, and basically, more control. You can then add external effects such as compression, EQ, reverb, etc. In fact, any sound you sample will benefit from the extra control and flexibility provided by running it through a mixing board.



4) Press SAMPLE LOWER (or UPPER) on the Mirage keypad.

The display will flash SL or SU for several seconds, and then enter 'level-detect mode'. The LED display has become a bar- graph, showing the level of the signal.

5) Set the output level of your sound source.



Adjust the level using the LED display just as you would a tape recorder, to achieve occasional, but never sustained, peak level. You may have to replay the section you intend to sample several times to get a good reading (which is why you will find tape more convenient). If the display does not move at all, check your connections. If you can't get a high enough level, press **CANCEL** to leave level-detect mode and check **[75] Line/Mic switch**, and **[74] Input Filter Freq** . You may need to switch to mic level if your preamp doesn't have enough gain. Be forewarned that the mic level setting compresses the signal and is about 20 times more sensitive than the line level setting so you will need to back off the gain. If the input filter frequency is set extremely low, it may completely filter out the signal.

6) To sample, press the ENTER button on the keypad.

Start your sound source a few seconds ahead of time, and press **ENTER** just before the section you want to sample. One of two things will happen in the display, depending on your signal.

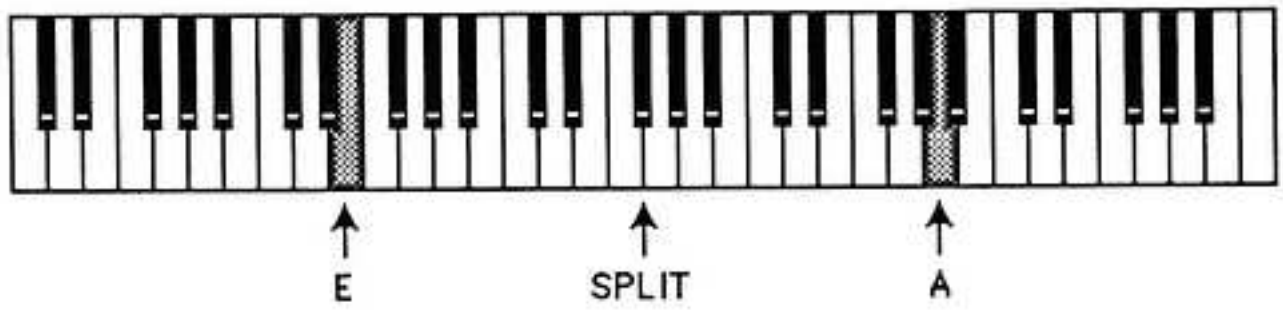
If the 'hit' is preceeded by silence then the display will go blank: the signal has not crossed the Sampling Threshold yet. When the sound starts, the middle bar should go on (indicating that sampling has begun), and stay on for about two seconds after which the display will revert to normal operation. If the sound never crosses the threshold, then the display will stay blank for about six seconds, then flash "nS" (No Signal).

If you are sampling a sound which is preceeded by other sounds, the middle bar will go on as soon as you press **ENTER**. Sampling starts immediately as the signal level is already over the threshold. It's up to you to press **ENTER** at exactly the right moment. If you really wanted the Mirage to wait for a certain sound and it didn't, then

[76] Sampling Threshold should be set to a higher value. If you are good at "punching in" with a tape recorder, you should have no problems. If you always seem to miss the beginning of the sound, you can press **ENTER** a little early, then trim out the undesired sound by rotating the wavesample left, or moving the wavesample start point. Of course you'll waste some memory on this undesired sound, so your sample time may need to be adjusted in order to capture the entire sound event you want.

7) When "SF" (Sample Finished) appears in the display, play the keyboard to hear the result.

Since the Multisampling switch **[77]** was off, you have taken a single sample over half the keyboard. The center key on the selected half (E for lower, A for upper) will play back the sound at the same pitch that was sampled. On that key, the sound will play for 2.2 seconds.



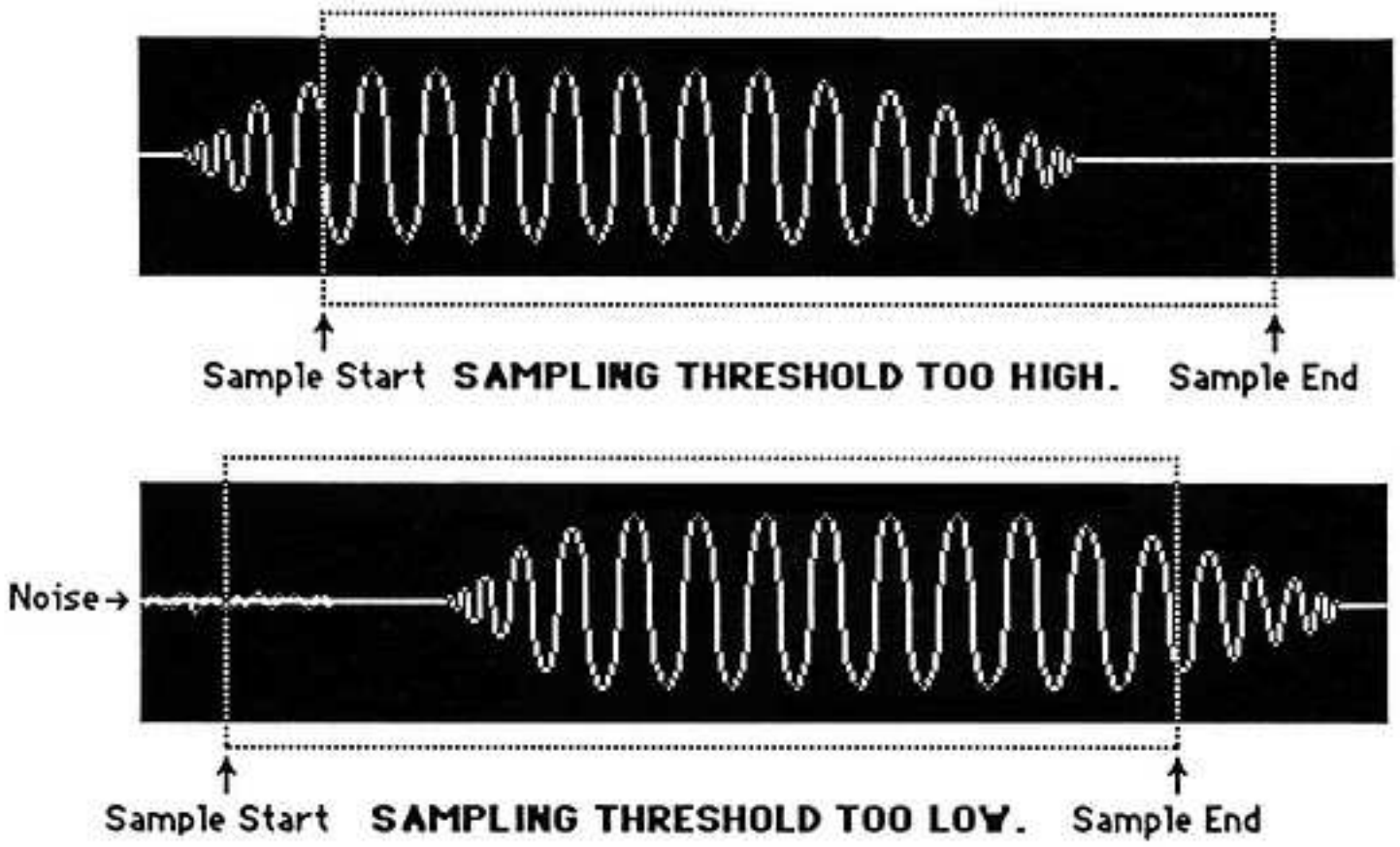
UNITY PLAYBACK FREQUENCY OCCURS IN THE CENTER OF EACH KEYBOARD HALF.

Now that you have sampled a sound, chances are, its not exactly the way you wanted it. Let's look at some of the ways you can get closer, either by resampling it or by adjusting the program parameters.

Readjusting Sampling Parameters

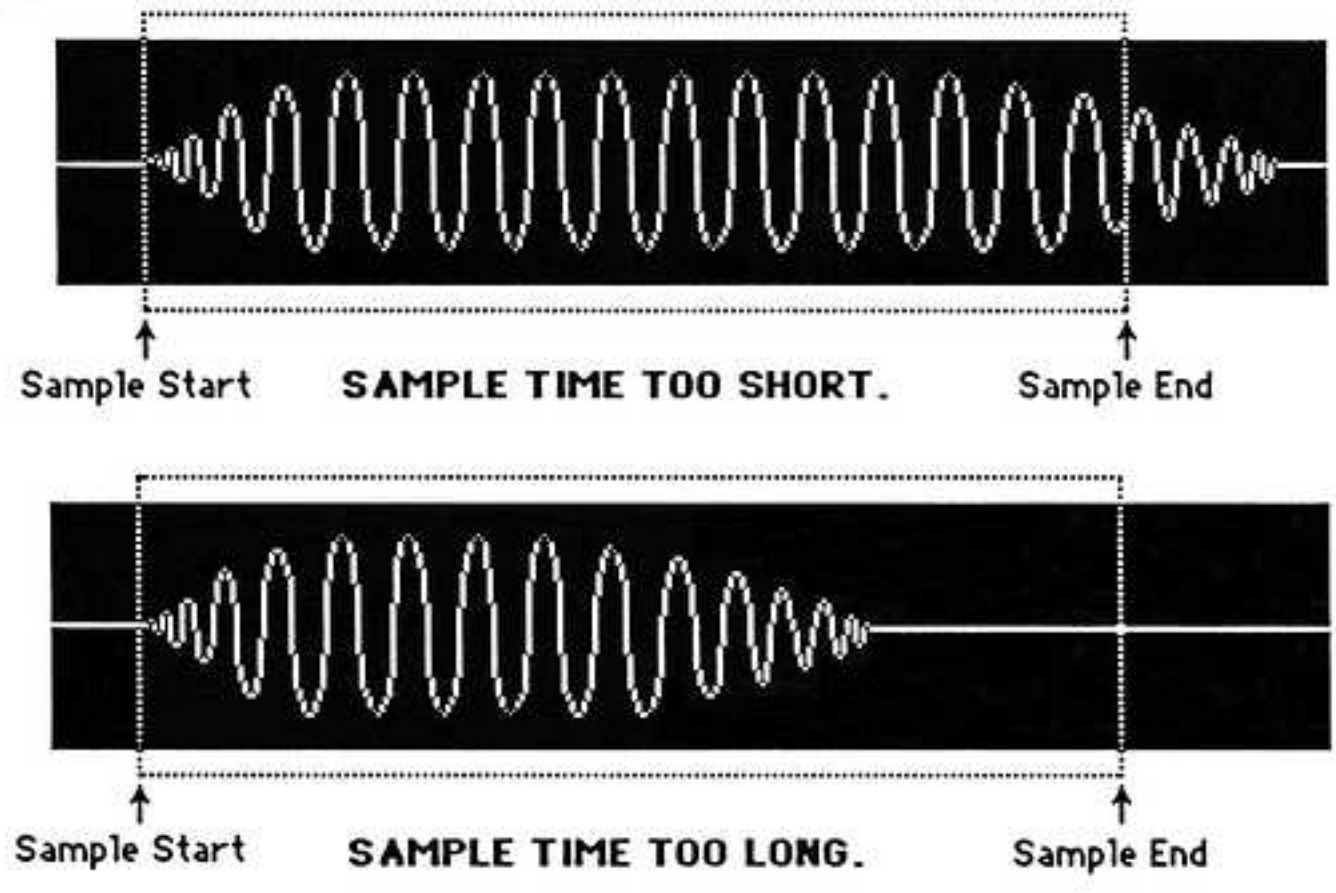
Problem: Attack is cut off or delayed

If you sample a sound with a slow attack and it plays back with a quick attack, then you may have cut off the beginning of the sound. This can occur when **[76] Sampling Threshold** is set too high. On the other hand, if there is a pause between when you hit a key and when the sound starts, you have sampled a bit of silence before the attack because the Sampling Threshold value was too low. The threshold level is usually best at a middle ground, between having it too sensitive, where it will trigger on any small random noise, and not sensitive enough, where it kicks in too late and misses the beginning of the sound. A setting of 10 is usually safe, but for best results you may want to adjust it for each sound you sample.



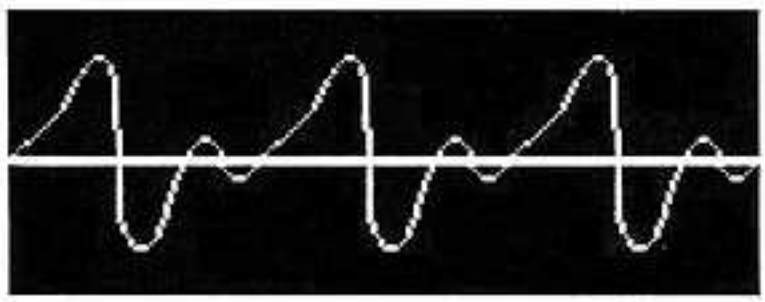
Problem: Decay is cut off or there is noise after the decay.

The Mirage can sample 2.2 seconds of sound at the default sample rate. If that's not enough time to capture a certain event, you can make it longer by increasing **[73] Sample Time** and resampling the sound. Remember this will decrease the sample rate, therefore decreasing the bandwidth, so there is a trade-off to be made. Usually you can adjust the filter and amplitude envelopes to cover up the abrupt end of a sample. If the sound doesn't last for the entire sampling window, you can adjust it in many ways. You can resample at a higher sampling rate (decrease **[73]**), giving better fidelity. You can move back the Wavesample End **[61]** which leaves more memory for other samples when multisampling. Finally, you can adjust the envelopes to end the sound sooner.

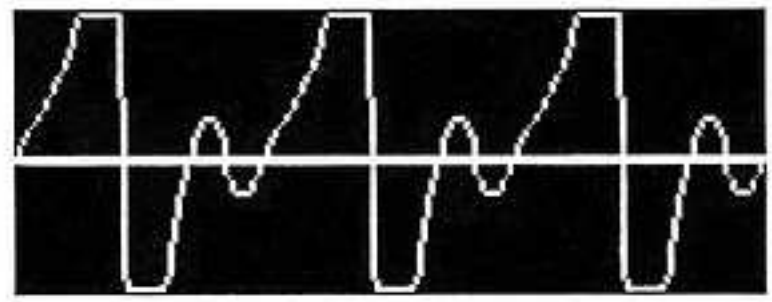


Problem: Distortion

The most common form of distortion is clipping. With a computer graphic display (provided by the **Visual Editing System**), it's easy to see. Even without a display, the sound should be fairly familiar to most musicians who have worked with electronic equipment. Unlike analog equipment which may overload in a smooth manner ("soft clipping"), an A/D converter clips in a very hard fashion, producing nasty distortion and alias noise. Obviously, the solution is to turn down the level of the signal going into the Mirage. In level-detect mode, the top LED bar lights when the signal reaches the overload level. You should see occasional peaks, but seeing the top bar lit a lot indicates an overload condition.



NORMAL



CLIPPED

Aliasing is another form of distortion. Input aliasing makes the signal sound rough or "ratty" and, at extremes, can produce audible frequencies which never appeared in the input signal. If you sample a sound with a downward sweep in pitch and hear a rising pitch when you play it back, that's aliasing! There are two solutions to the problem: either limit the high frequencies coming in, or increase the sample rate (by decreasing the Sample Time **[73]**). In general, use as high a sample rate as possible which still gives you enough time to capture the whole sound. For any given sample rate, adjust **[74] Input Filter Freq** to get the best sound. Here are some guidelines:

The filter setting is related to the sample rate, but not in an absolute way. While a general rule of thumb is that the filter should be set to a frequency one octave below the *Nyquist* frequency (in other words, one quarter of the sample rate--see Part IV), it really depends on the signal. Signals with very strong ultra-high harmonics will need a lower setting to eliminate aliasing. Sound sources with inherently limited frequency response, such as records and tapes, may need very little filtering. The sharp cutoff slope of the **Input Sampling Filter** will allow you to sample sounds with strong high frequency content without excessively limiting the frequency response.

It's worth spending some time training your ear to the effects of aliasing. Try leaving the Input Filter **[74]** wide open and bringing the sample rate way down. Aliasing will make complex sounds raspy or distorted. Aliasing of simple sounds can create strange low or mid frequency tones. Some aliasing can even be quite acceptable on certain sounds with high noise content, such as drums and cymbals.

Adjusting Playback Parameters

Explaining how to use envelope generators is beyond the scope of this book. They are just as important in creating useful sounds on the Mirage as they are on a synthesizer. The Mirage envelope generators provide a great deal of flexibility through modulation of the individual stages. Familiarize yourself with them thoroughly and you will be able to take any old sample and turn it into a musically viable sound.

Before you sample, there is always some sound present in the Mirage, such as the piano if you had just loaded Sound Disk #1. When you sample, the Program parameters are not changed; they remain behind, even though the piano Wavesample is gone. When you play your newly sampled sound, you will hear it with the piano envelopes, the piano filter setting, whatever is set up in the current program. This may not be what you had in mind! In particular, the envelope shapes and chorusing may be changing the sound of your sample drastically. If the sound dies away, sounds too muffled, or doesn't have a sharp attack, suspect the playback parameters. That is why its usually best to load the "plain vanilla" sound from the MASOS disk (or a blank Formatted disk) before sampling.

You can also use this feature to your advantage. As with a synthesizer, it's often easier to modify an existing patch than to start from scratch. Before you sample, load in a sound which has a Program similar to the sound you want. For example, if you are sampling a french horn, you might want to try sampling into the trombone Program. Consider what type of velocity feel you want, whether the filter should be kept high or swept down, etc. Eventually you may want to save a sound which is specially designed for the types of sounds you like to sample; a personal template. Load this template in before you sample, and most of the work will already be done for you. Also, remember that you can copy Programs from one sound to another, after you have a wavesample you like.

EXAMPLE 2: SAMPLING A SYNTHESIZER OR GUITAR WITH A SHORT LOOP

Sustaining simple sounds is a matter of looping on a single cycle of the input waveform. The key to looping is to sample the right pitch at the right sample rate. The Mirage can only sample at certain fixed rates, so we will need to tune our sound source to meet the Mirage's demand (If an instrument is not tunable, it may be necessary to record it on a variable speed tape recorder and tune the playback speed.).

The first instrument you try to loop should be one which is clearly pitched, tunable, and has no vibrato. Guitar, bass guitar, organ, synthesizer, or other keyboard instruments are all good choices. It is much more difficult to loop instruments where a fallible human is involved in sustaining the pitch, such as woodwinds, bowed strings, or voice, since the pitch isn't likely to remain constant throughout the sustain. If you have a synthesizer, find an electric piano patch, or something else percussive with a clear sustain. If it has a built in chorusing effect, turn it off. Chorusing will only make looping more difficult and you can add the chorusing back later using detuning on the Mirage.

In this example you will sample the note A, 110 Hz, an octave and a third below middle C. This is the open A string on a guitar. The Mirage expects a specific pitch, which, at the nearest available sample rate, happens to be slightly higher than a perfectly tuned 'A'. For proper looping, retune the instrument to be sampled as follows:

1) Set **[21] Master Tune** to 55 (25 cents sharp). This will allow the A key on the Mirage to be used as a pitch reference for retuning the guitar.

2) Load the Drums/Bass-Synth sound (lower sound #3) from **ENSONIQ** Sound Diskette #1. This provides a clear tone as a tuning reference.

3) Tune the instrument to be sampled to the A on the Mirage Bass-Synth.

4) Load lower sound #1 (plain vanilla) from the MASOS disk.

The default tuning of the "plain vanilla" sound will cause unity playback to occur on E in the middle of the lower keyboard. Since you'll be sampling an A, you should change the wavesample relative tuning parameters as follows:

5) To tune lower wavesample #1:

- Select lower wavesample #1 (**[26]**= 1).
- Set **[67] Coarse Tune** to 4.
- Set **[68] Fine Tune** to 80.

This provides a tuning offset of a half octave (80 Hex=128 and Fine Tune provides 256th octave steps, so 128/256ths of an octave is a half octave), the difference between A and E. With this offset, the A you sample will play back as an A on the A key, rather than the E key.

6) Turn **[77]** User Multisampling on. This will prevent the Mirage from changing the tuning back to the default settings when you sample.

7) Set the sampling parameters as follows:

Parameter	Value	Meaning
[73] Sample Time Adjust	35	28.5 KHz sample rate
[74] Input Filter Freq	80	7KHz cutoff
[75] Line Level Input	ON	line level
[76] Sampling Threshold	10	trigger at +/- 10
[77] User Multisampling	ON	no defaults

The crucial parameter here is the Sample Time (sample rate). For sounds with short loops, we must use the right sample rate so the waveform lines up with the loop boundaries. If the waveform doesn't line up, the loop pitch and sound quality will not match the input. The table in Appendix A gives the proper sample rates for all of the equi-tempered scale pitches.

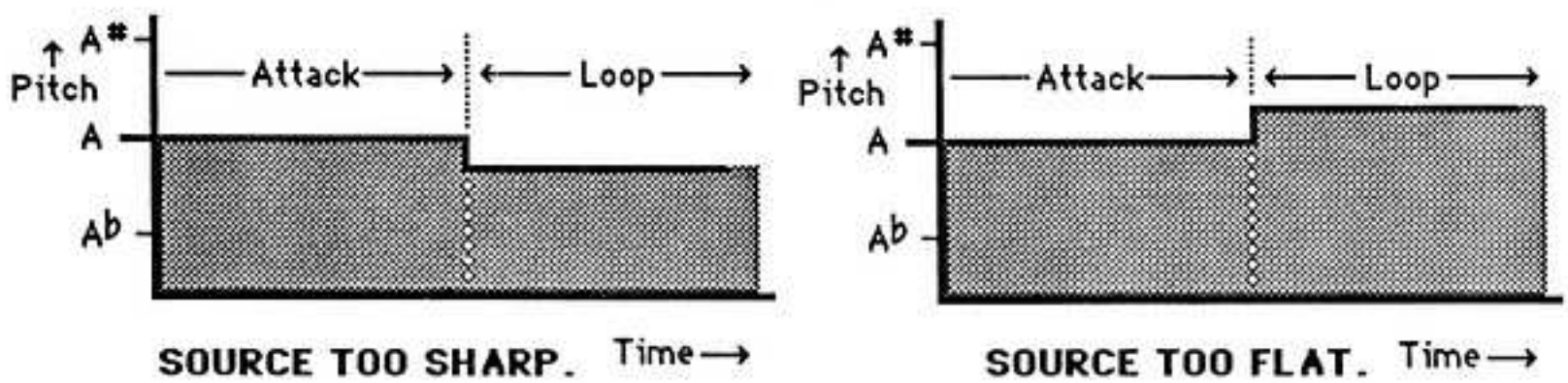
8) Do some trial samples to adjust the output level of your signal source and the Sampling Threshold level **[76]**.

9) Turn the Loop Switch **[65]** ON. This will provide a one page loop starting at FE. You can sample a sound while leaving the loop switch ON, as long as the Multisampling switch is also ON (the Loop Switch will automatically be turned OFF if you sample with Multisampling OFF).

10) Take a sample of the open A string on a guitar, or the A a tenth below middle C on a keyboard.

11) Correct the tuning of the sound source, as follows:

Play the low A on the Mirage keyboard. It should be the same pitch as the sound you sampled. Listen for the point where it drops into the loop. This should be about two seconds after the attack and it will probably not be the same pitch as the rest of the note. If the sound seems to sustain forever, at the same pitch, then you're done! If, instead, the sound drops off into silence, then perhaps you didn't hold the note long enough; the Mirage is looping on the silence after the note. Most likely, when you get to the loop there will be a small change in the pitch, and tone, of the sound. Think of the loop portion as the "correct" pitch. If the first part of the sound is lower in pitch than the loop, raise the pitch of your sound source. Likewise, if the first part is higher in pitch than the loop, your source is too sharp. Tune it down.



Keep re-sampling and re-tuning until the pitch of the first part of the sound (the part before the loop) matches up exactly with the pitch during the loop. It should become a smooth transition, with no tonal change at the "splice".

12) Now that you're all tuned and set up, you may want to try to get the right "performance". If you're sampling a guitar, try different playing styles: muted, plucked, harmonics, etc. Save the ones you like on a spare disk; you can always erase them.

13) Once you have a sample you like, try different program settings: release times, filter cutoff, chorusing, etc. The Program can make the difference between an ordinary sample and a great sound. Remember to set Master Tune **[21]** back to 50 when you are finished to bring the Mirage back to concert pitch.

EXAMPLE 3: MULTISAMPLING A DRUM SET

One of the outstanding features of the Mirage is its multisampling flexibility. Up to sixteen different wavesamples can be spread over the keyboard. As you increase the number of active wavesamples, it gets harder to keep track of all the parameters. Pencil and paper become necessary tools. To explore multisampling, it pays to plan out the keyboard assignment and memory allocation ahead of time. In this example, we'll go through splitting up the keyboard and assigning wavesamples to the keys, and then splitting up the memory and allocating memory to each wavesample.

First, you must make some decisions about what you want to do. In this example, we've decided that we want the drums on the upper keyboard, so that we can play a bass sound along with them on the lower keyboard. There is a maximum of eight wavesamples on the upper sound, and we'll use them all.

Drums are difficult to sample. Unless you have access to a recording studio and a tireless precision drummer, don't try to sample live drums. It's much easier, for learning purposes, to sample drums that have already been processed and recorded. One good source is a digital drum machine--the best one you can get. If you are really interested in recording your own unique drum sounds, get them on tape first, and sample them from the tape. In this way, you can do numerous retakes and adjust or process the sound to your liking.

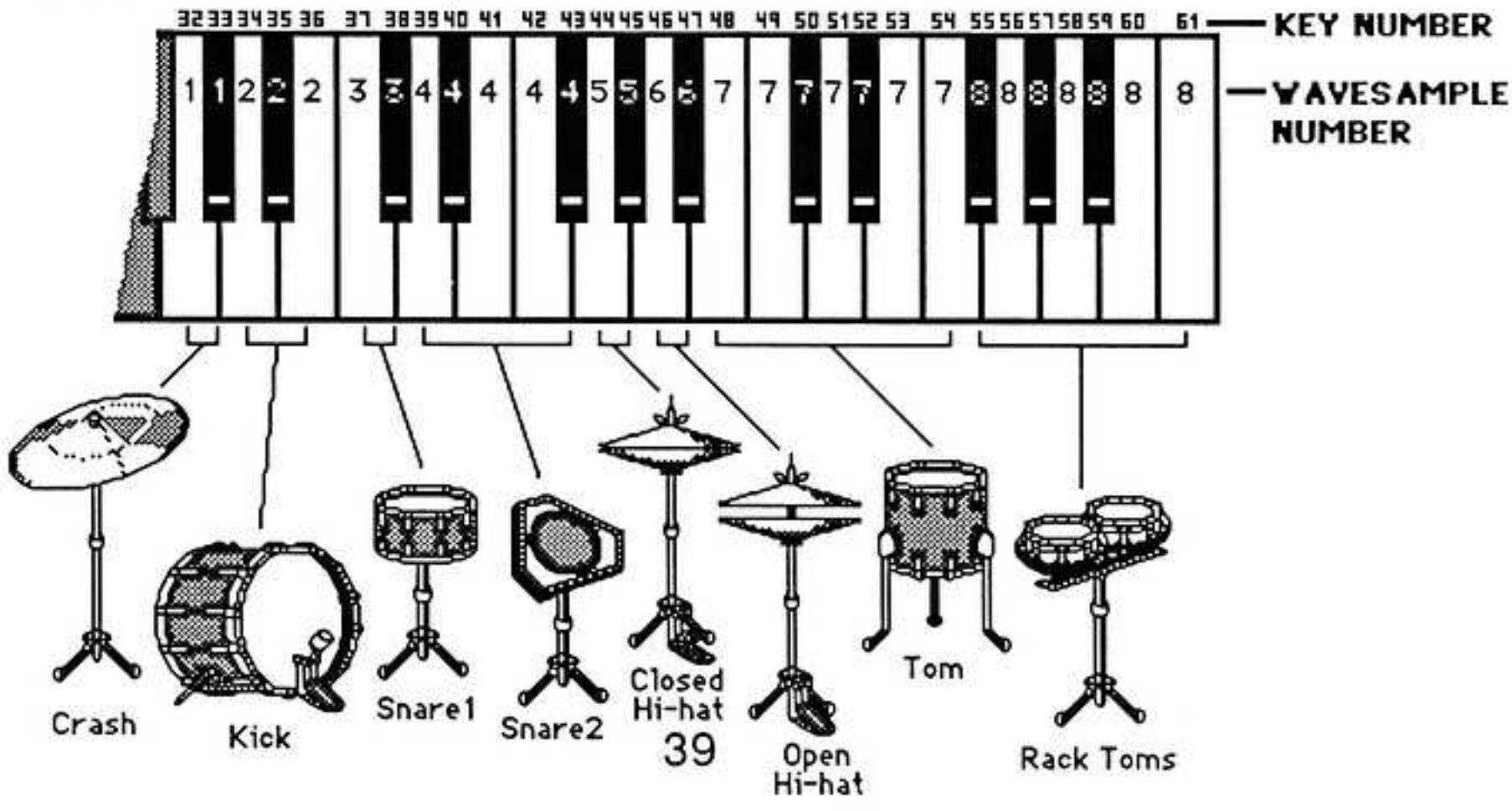
Besides choosing what sounds you want, you must decide where you want them to appear on the keyboard. There are several considerations here.

1) Playability - group the kick drum and snare together, since they will be played the most; group the closed and open hi-hats together; etc.

2) Range - a kick drum doesn't usually transpose very well and you don't need two octaves of it. On the other hand, tom-toms cover a wide range and transpose very well.

3) Compatibility - certain drum machines may put out MIDI note information which could drive the Mirage very nicely. This would allow the Mirage to be played from the drum machine, with all of its features (quantizing, swing, etc.). If the format of the drum machine isn't programmable your Mirage drum layout will have to conform. We'll assume this isn't important at the moment.

After deciding what layout is suitable, make a diagram of the keyboard, and plan out the locations and ranges of the different sounds. Then make a chart of the Top Key **[72]** values for each wavesample.



Wavesample	Sound	Keys	Topkey
1	Crash cymbal	32-33 (g-g#)	33
2	Kick drum	34-36 (a-b)	36
3	Snare 1	37-38 (c-c#)	38
4	Snare 2	39-43 (d-f#)	43
5	Closed Hi-hat	44-45 (g-g#)	45
6	Open Hi-hat	46-47 (a-a#)	47
7	Toms	48-54 (b-f)	54
8	Rack Toms	55-61 (f#-c)	61

What we've done is give most of the sounds two keys for easy rolls, and grouped them together so that you can play kick drum, snare and hi-hat with one hand. If you slide your thumb over, you can hit the crash cymbal and the kick-drum at the same time. The tom-toms have 15 keys, so you have 15 tuned drums at your fingertips. Note that the key names (c,c#,etc.) have nothing to do with the "note" that each drum plays. The tuning of each wavesample is independent.

Once you make up a map like this, you can read off the value of Top Key **[72]** for each wavesample. To implement the plan, you just have to do some fingerwork.

Setting Up

1) Load upper sound #3 from your MASOS disk which has the "plain vanilla" sound template for Multisampling with eight wavesamples.

2) Turn User Multisampling **[77]** on since we will be multisampling.

3) Select the upper sound using the **0/PROG** key (be sure the display shows a "U", you wouldn't want to spend all of your time setting things up only to find later that you've set up the wrong half of the keyboard!).

4) Set up the Keyboard-Wavesample assignment. You will be setting which keys are assigned to each wavesample according to the drum list. Do this, from left to right:

Wavesample			Topkey		
Select **[26]**,	set it to 1	then	Select **[72]**,	set it to 33
Select **[26]**,	set it to 2	then	Select **[72]**,	set it to 36
Select **[26]**,	set it to 3	then	Select **[72]**,	set it to 38
Select **[26]**,	set it to 4	then	Select **[72]**,	set it to 43
Select **[26]**,	set it to 5	then	Select **[72]**,	set it to 45
Select **[26]**,	set it to 6	then	Select **[72]**,	set it to 47
Select **[26]**,	set it to 7	then	Select **[72]**,	set it to 54
Select **[26]**,	set it to 8	then	Select **[72]**,	set it to 61

You can select wavesamples using either **[26]** or the MASOS **Select Upper Wavesample** key (in which case **[72]** remains selected, making things easier).

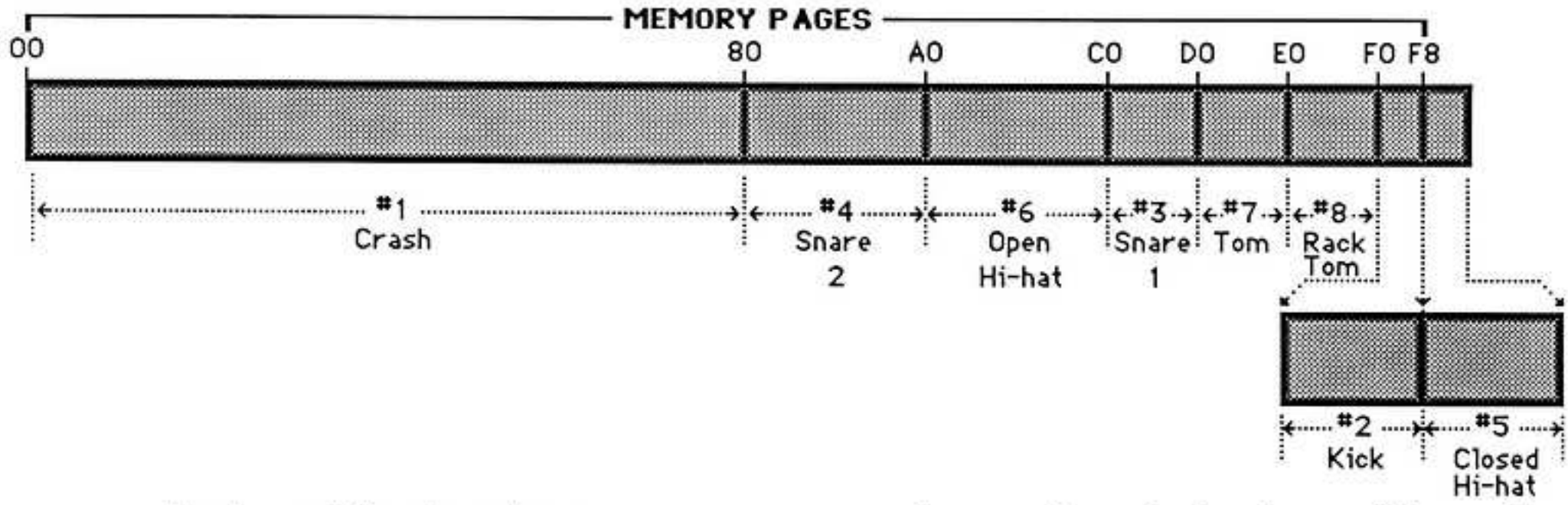
5) Now we have our drum set organized on the keyboard, but we still haven't put any sounds into memory. Since we have no way of knowing in advance exactly how long each sound will be, we have to experiment. We can make some guesses, though. The Crash Cymbal will take the most memory; it has both a long decay and high frequency content. The Kick Drum and the Closed Hi-hat, on the other hand, will use very little memory; they are short sounds. The other sounds, the Snares and Toms, fall in the middle; they're about a quarter to a half second long.

As a first guess, we might divide memory up this way:

Wavesample	Sound	# of pages	# of pages in Hex
1	crash cymbal	128	80
2	kick drum	8	08
3	snare 1	16	10
4	snare 2	32	20
5	hi-hat closed	8	08
6	hi-hat open	32	20
7	low toms	16	10
8	rack-toms	16	10
	Total	256 pages	100 pages Hex

We've juggled the space allowed for each sound so that they all fit in 64K (256 pages) of memory. Now we have to set up all the Wavesample Start **[60]** and Wavesample End **[61]** parameters to put this format into effect. When deciding where to put each wavesample, it's best to put the biggest samples in first, starting them on nice round number boundaries. That is, try to start the largest wavesamples at 0 or 80; then put the next largest at 40 or C0; then fit the smaller ones into leftover memory. The process might go like this:

Start wavesample #1 (128 pages) at 0, which will take all the memory up to 80 (it will end at 7F). Next put wavesample #4 (32 pages) at 80 and wavesample #6 (32 pages) at A0, since they're the next largest. Then, fill in the rest of the 16 page wavesamples, #3, #7, and #8. The two 8 page wavesamples fill in the remaining memory space. The most difficult part is getting used to the idea of round numbers in Hex. This is rather difficult to explain; its a little easier to grasp in pictures.



Again, a little planning on paper can save time and confusion later. When all eight wavesamples are being used, and they are not all the same size, its hard to remember all of those parameters.

Note that it is not necessary for the wavesamples to be consecutive in memory; wavesample #2 does not have to follow wavesample #1. It's also not necessary that the wavesamples butt up against one another. If we decide to shorten the end of wavesample #1 to less than 7F, we don't have to move the next wavesample (#4) down to take up the space; let it start at 80 anyway.

6) The process of setting up these memory divisions is just like setting up the keyboard assignment divisions. Go through each wavesample and set Wavesample Start and Wavesample End. Start with #1:

Wavesample		Wavesample Start		Wavesample End
Select **[26]**, set to 1;	then	select **[60]**, set to 00;	then	select **[61]**, set to 7F
Select **[26]**, set to 2;	then	select **[60]**, set to F0;	then	select **[61]**, set to F7
Select **[26]**, set to 3;	then	select **[60]**, set to C0;	then	select **[61]**, set to CF
Select **[26]**, set to 4;	then	select **[60]**, set to 80;	then	select **[61]**, set to 9F
Select **[26]**, set to 5;	then	select **[60]**, set to F8;	then	select **[61]**, set to FF
Select **[26]**, set to 6;	then	select **[60]**, set to A0;	then	select **[61]**, set to BF
Select **[26]**, set to 7;	then	select **[60]**, set to D0;	then	select **[61]**, set to DF
Select **[26]**, set to 8;	then	select **[60]**, set to E0;	then	select **[61]**, set to EF

Again, if you are using MASOS, you can select the desired wavesample directly.

7) The other parameters in the Program also need to be set up, but there's a sneaky way to do that. Load the Drums/Synth-Bass (lower #3) from Sound Diskette #1 into the lower keyboard. We will copy lower Program 1 of that sound into Program 1 of the upper keyboard since lower Program 1 is already set up with drum-like envelopes on the filter and amplitude. The copy procedure is:

Select lower Program 1 with the **0/PROG** key (if you have just loaded in the sound, lower Program 1 will already be selected, but it can't hurt to check). Select **[16] Copy Program to Upper** and press the **1** key. Finally, hit **ENTER**. This copies all of the Program parameters (but not the wavesample parameters) from the lower drum Program to our new sound. Should you be sampling sounds for which you can't find a suitable Program to copy, use the MASOS (or blank diskette) Program settings. You can go in and adjust the Program to your liking when you're finished sampling. Just remember, all of the wavesamples will be processed through the same Program, so keep it subtle or you may create undesirable effects on other sounds

Now, finally, we can get to the business of sampling the sounds. We need high frequencies, so set the Sample Time **[73]** to 30 (a sample rate of 33KHz). Make sure the Multisampling switch **[77]** is on and select wavesample #1, for the Crash Cymbal. It really doesn't matter which one we start with; all will follow the same procedure.

Repeat for Each Wavesample

1) Select the desired wavesample (**[26]** or MASOS key) and press the **SAMPLE UPPER** button.

2) Set up the level so that the top bar just faintly blinks on the attack of the sound.

3) Press **ENTER** and play the sound into the Mirage.

4) Using the previous chart, find the keys on the keyboard that play that sound. Adjust the frequency of the wavesample until it plays back at the original pitch. Tune by ear using **[67] Relative Tuning Coarse** and **[68] Relative Tuning Fine**

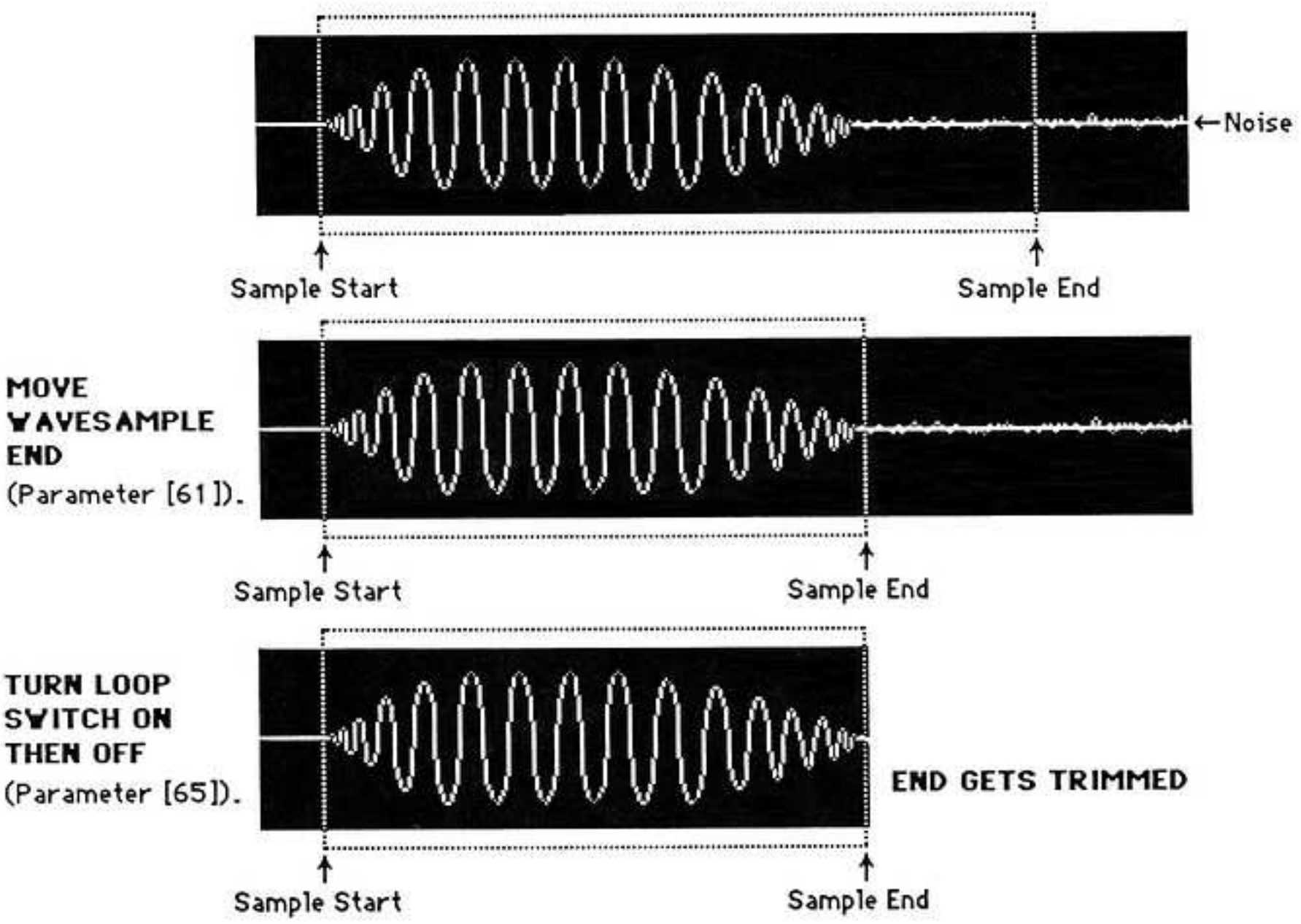
5) If the sound is 'clipped' or not loud enough, adjust the input level and sample it again. If the attack of the sound is cut off, try lowering the threshold level.

6) Trim the end of the sound as necessary. If the sound has a period of silence after it or noise which sounds more like frying bacon, then trim the end as follows:

First, save the sound on disk! The trimming process will permanently alter the sound in memory. Always save a copy of your sound on disk so that you can retrieve it later should your experiments go awry.

Second, move the Sample End parameter **[61]** a few pages lower. This will have no effect until you...

Third, turn the Loop switch **[65]** on and then off again. This truncates the sound to the new Wavesample End location.



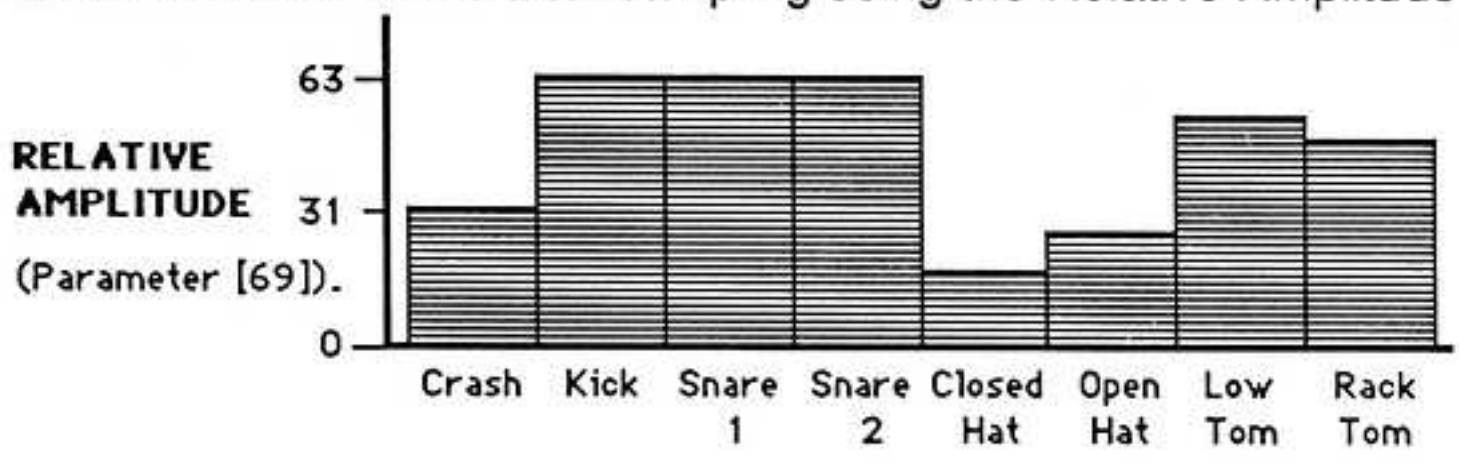
Finally, evaluate the new end point by playing the sound. If there's still some noise, try taking more off. you can do this by going back to the second step. If you cut off too much, the sound will end abruptly (perhaps with a click) and you must reload the untrimmed sound from disk. The trimmed version cannot be restored to its old length. Keep trying different lengths.

Another alternative is to resample the sound with the new Wavesample End pointer. This may be quicker, depending on the sound source you are using. You can also find the end of the sound by moving the Wavesample Start to the point where you hear the sound end. Next, set the Wavesample End to this value, return Wavesample Start to its original value and turn the loop on and off. Moving Wavesample Start doesn't trim the data, so you can zero-in without having to reload if you overshoot.

After Sampling

Adjust the Relative Filter Freq **[70]** and Amplitude **[69]** for each wavesample. Once you have all eight sounds in the machine, you can fine tune them. For most of the sounds, the filter should be set so high that it has no effect. This can be done by setting the wavesample Relative Filter Freq **[70]** to a high number, like 99. Other sounds, like the Toms, may need more filtering. Adjust the Program filter envelope to suit the Tom-Toms; it won't affect the Cymbals if the Relative Filter Freq of the Cymbals is high enough. The important thing is, adjust the filters so everything sounds good. Don't assume that everything will sound best when cranked up all the way--that doesn't usually work on a synthesizer and probably won't work on the Mirage.

The Wavesample Relative Amplitude is important for drums, since the sounds are so different from one another. The Hi-hat, in particular, will need to have a lower amplitude to fit in well with the rest of the drum set. Don't try to balance the levels when you're sampling, you want to record the hottest level you can for optimum sound quality. You can balance levels after sampling using the Relative Amplitude parameter.



A neat trick--If you are sampling a tunable drum machine, you can get the effect of super-high sample rates by tuning the sound down from its normal pitch. For example, tune the Hi-hat down an octave on the drum machine and sample it at 25KHz. After adjusting the playback tuning to bring the sound back up to the original octave, when you play the sound back, the frequency response will be as if it were sampled at 50KHz. This trick can also be used with sounds on a variable speed tape deck.

If you can't allocate enough memory to a Cymbal, it will sound funny if it abruptly cuts off. It is extremely difficult to loop a Cymbal (non-harmonic overtones and all that). One solution is to perform a manual 'fade-out' during the sampling process; One hand to play the sound and the other to turn down the volume. With some practice you can get a smooth fade-out that sounds natural. Another solution is to make the amplitude envelope decay to zero before the end of the sound, however, this may have an undesired affect on the other sounds.

EXAMPLE 4: CROSS-FADE DIGITAL SYNTHESIS

In this example we'll go through some of the more esoteric capabilities of MASOS. The Mirage is actually a flexible digital synthesizer since it can manipulate waveforms to create new electronic sounds of its own.

There are MASOS commands to Copy, Fade in, Fade out, and Add waveforms. By combining these operations we will create a wavesample which is the merging of two different waveforms.

The basic process goes like this:

1) Select two waveforms of one page each. We'll call these Wave1 and Wave2. Practically any page from any sound will do. They can be taken from a sound that you already have on disk, or you can sample them just for this experiment. The operations that we will perform will turn any waveform into a pitched sound. One choice might be a very bright sound for Wave1 (for example a waveform from a saxophone wavesample) and a more mellow, pure tone for Wave2. There is a pure sine tone on Sound Diskette #1, on Program 4 of the wooden flute sound.

2) Load lower sound #1 from the MASOS disk. This will give you plain vanilla envelopes and two wavesamples on the lower keyboard.

		Wavesample #1	Wavesample #2
[60]	**Wavesample Start**	00	7F
[61]	**Wavesample End**	80	FF

3) Copy wave1 into lower page 00. Wave1 is whatever complex wave we would like to use. We can pluck a page out of any upper sound by copying it to the lower. Let's say that we would like page FE (Hex) from the saxophone sound on Sound Diskette #3. Load the sound into the upper memory as usual. Make sure you are currently in upper memory (use the **0/PROG** key). Set **[85] Source Start Page** to FE. Set **[87] Source End Page** to FE. Set **[89] Destination Page** to 00. Set **[94]**

Destination Memory Bank Select to LO (lower). Press the **FUNCTION** Key (**LOAD SEQ**). Press the **1** key (for the Copy function) and **ENTER**.

4) Replicate Wave1 from 00 to 7F. A copy of wave1 is now in page 00 of the lower memory. We will make every page, from 00 to 7F, exactly the same. Make sure you are currently in lower memory. Set the Source Start **[85]** to 00. Set the Source End **[87]** to 7F. Press the **FUNCTION** Key. Press the **8** key (for the Replicate function) and **ENTER**.

5) Fade Out from 00 to 7F. All the Source pointers should still be in place, so just execute the function. Press the **FUNCTION** Key, the **3** key (for Fade out) and **ENTER**. The resulting waveform data will now fade out in level from full volume at 00 to silence at 7F.

6) Copy Wave2 into lower page 80. Wave2 will be the waveform that we will merge into. For example, if we wish to use the sine wave on Sound Diskette #1, load only upper sound 3, the wooden flutes. Upper Program 4, a vibes sound, uses wavesample #3, a sine wave, which is at page 7E in memory. Follow the same process as in step 3 above, but set the Source to page 7E, and the Destination page to 80 instead of 00. This will copy upper page 7E (the sine wave) to lower page 80.

7) Replicate Wave2 from 80 to FF. Make sure you are currently in lower memory. Set the Source Start **[85]** to 80. Set the Source End **[87]** to FF. Press the **FUNCTION** Key, the **8** key (for the Replicate function) and **ENTER**.

8) Fade in from 80 to FF. All the Source pointers should be still in place, so just execute the function. Press the **FUNCTION** Key, the **2** key (for Fade in) and **ENTER**. The resulting waveform data will now fade in from silence at 80 to full volume at FF.

9) Add the fade in waveform to the fade out waveform. The Source pointers will already be set up properly, from page 80 to FF. Set the Destination page to 00, on the lower memory. Press the **FUNCTION** Key, the **5** key (for Add) and **ENTER**. Wavesample #1 will now contain the merged waveforms. Wave1 fades out as Wave2 fades in. In the middle, around page 40, they are evenly mixed together.

Using the MASOS functions, you can create an infinite variety of sounds, far more complex than any synthesizer. It is possible to create unusual timbres that still retain acoustic qualities. You can effectively layer sounds by adding them. You can reverse sounds so they play backwards, add a backward sound to a forward sound, create bi-directional loops to smooth loop transitions and you can process your final sound through the analog synthesizer section. In short, the Mirage is not only a powerful sampling keyboard, it is also an outstanding digital synthesizer and analog synthesizer. Further uses of MASOS are described in the appendix.

PART IV - ADVANCED SAMPLING TECHNIQUES

This chapter starts with an explanation of the basic concepts used in sampling systems, then it moves to the art of sampling; the tools, techniques, and tricks that can be used to get the highest quality sounds into, and out of, the Mirage.

SAMPLING TERMS

The musician who is serious about sampling sounds will need to be familiar with a few terms:

A *Sample* is one number representing the amplitude of a signal at one instant in time. Most sounds will require thousands of samples to be accurately represented. In the Mirage each sample is a byte. This means that it is an eight-bit number, having a value from -127 to +127 (255 different values).

The *Input Sample Rate* is the frequency of the analog-to-digital conversion while recording a sound.

The *Output Sample Rate* is the frequency of the digital-to-analog conversion while playing back a sound.

Aliasing is distortion caused by sampling sounds with frequencies which are greater than one half the input sample rate.

The *Input Filter*, or *Anti-aliasing Filter*, limits the high frequency content of the sound being sampled to prevent aliasing. Its setting will depend on the input sample rate.

In addition, we have invented the term *Wavesample* to mean a single sampled sound, of variable size, with an associated group of 12 parameters which control where the wavesample is in memory and how it should be played. Sixteen wavesamples reside in the Mirage memory; 8 in lower memory and 8 in upper memory. Often they are not all used, but they are always there.

SAMPLE RATE

Digital sampling is, on the simplest level, much like tape recording. As with tape recording, the recording time must be traded off with recording fidelity. The highest quality professional tape recorders run at fast speeds, such as 15 or 30 inches per second. Cassette recorders go at the slow speed of 1-7/8 inches per second.

In sampling, we use digital memory instead of tape. The sample rate in a digital system is like the tape speed. Higher rates can do a better job of reproducing higher frequencies, but use up memory faster just as higher tape speed uses more tape.

Unfortunately memory is considerably more expensive than tape; we have to carefully consider how we use it up. There is always a trade-off between bandwidth and sample duration. At lower sample rates the sound can last longer; at higher rates it will be shorter, but the frequency response will be increased.

In the Mirage, each half of the keyboard has 64K bytes of memory for its sound recordings (that's 64 times 1024, or 65,536 bytes). At the default sample rate of 29.4 KHz, a sound will last 2.2 seconds. It's easy to compute:

$$\frac{\text{65,536 samples}}{\text{29,400 samples per second}} = 2.2 \text{ seconds}$$

That means that we can record 2.2 seconds of sound at the default sample rate. If we transpose that sound up one octave, by playing higher on the keyboard, the sound will only play for 1.1 seconds. Going down an octave will make the sound play for twice as long; down two octaves and it will be four times as long, or 8.8 seconds.

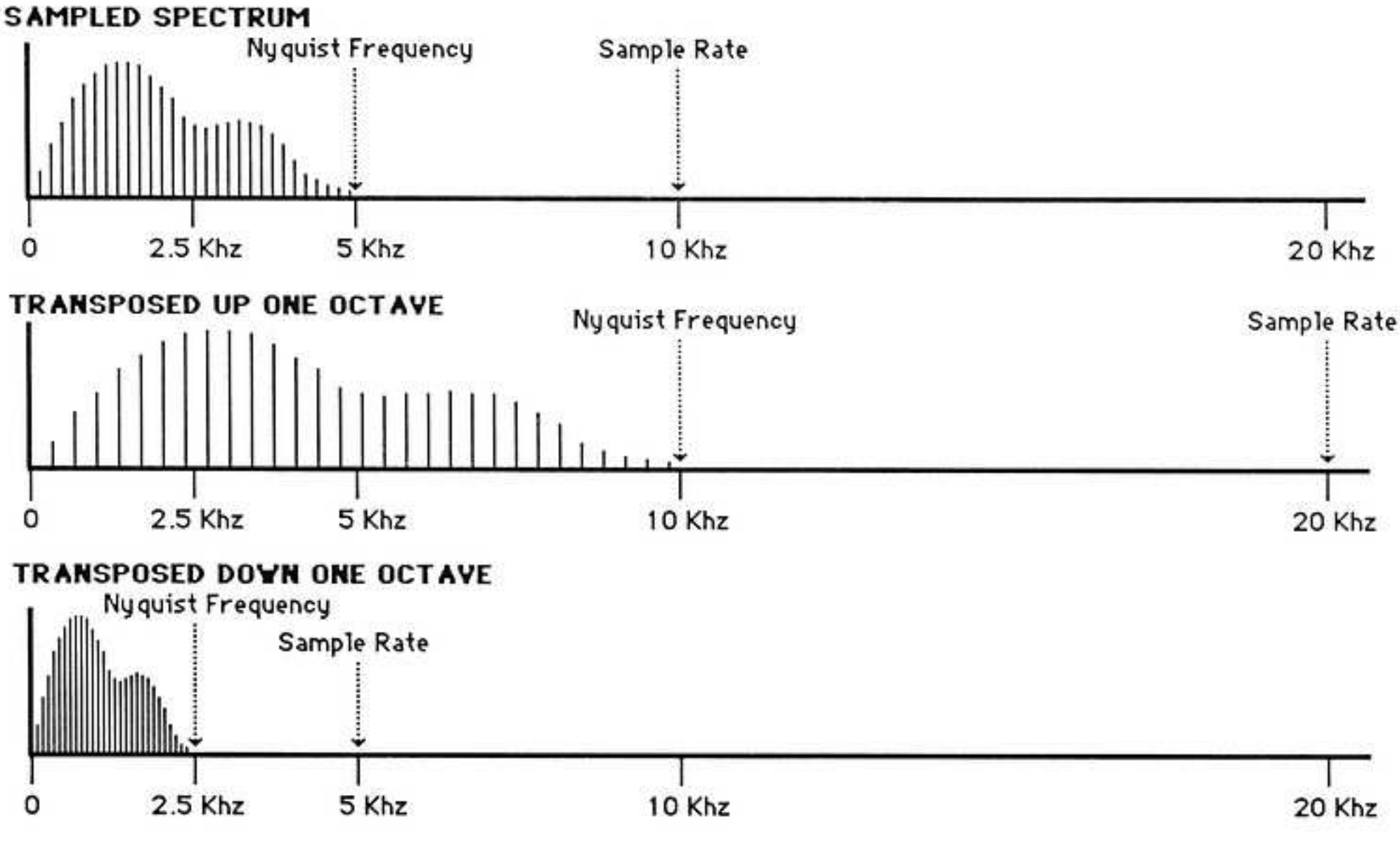
If we wanted to record a sound that was 6 seconds long, we would start with a different sample rate:

$$\frac{\text{65,536 samples}}{\text{6 seconds}} = \text{10,922 samples per second (10.9 KHz)}$$

We need a sample rate of 10.9 Khz to fit 6 seconds of sound in 64K of memory (you would set the Sample Time to 91 µsec). At this rate, the original sound will be 6 seconds long when we play it back at unity. Playing an octave lower, the sound lasts for 12 seconds, and down two octaves stretches it to 24 seconds. Some New Wave songs are shorter than that!

We see that we can change the sample rate to get different recording times in the same amount of memory, but what happens to the frequency response as we do this? The frequency response of a digital system is directly related to the sample rate. The theoretical maximum frequency that can be sampled or played back is one-half the sample rate. This is known as the *Nyquist* frequency.

As we transpose a sound by playing it on the keyboard, we are changing the output sample rate, and thus changing the frequency response. For example, if we sample a sound with a 10KHz sample rate, then it has (possible) spectral content up to 5KHz. Playing it back on the keyboard, we can transpose it up one octave by doubling the playback sample rate. Now the output sample rate is 20KHz, and the Nyquist point has moved up to 10KHz. As we transpose a sound, we transpose its spectrum with it. Likewise, if we played the sound down one octave, the Nyquist frequency would be cut in half; down to 2.5KHz. Go down two octaves and we've cut our frequency response down to 1.25 KHz! When transposing up we could in theory, extend the frequency response indefinitely. In practice, the Mirage imposes a 15KHz limit on the output, even when transposing a sound up many octaves.



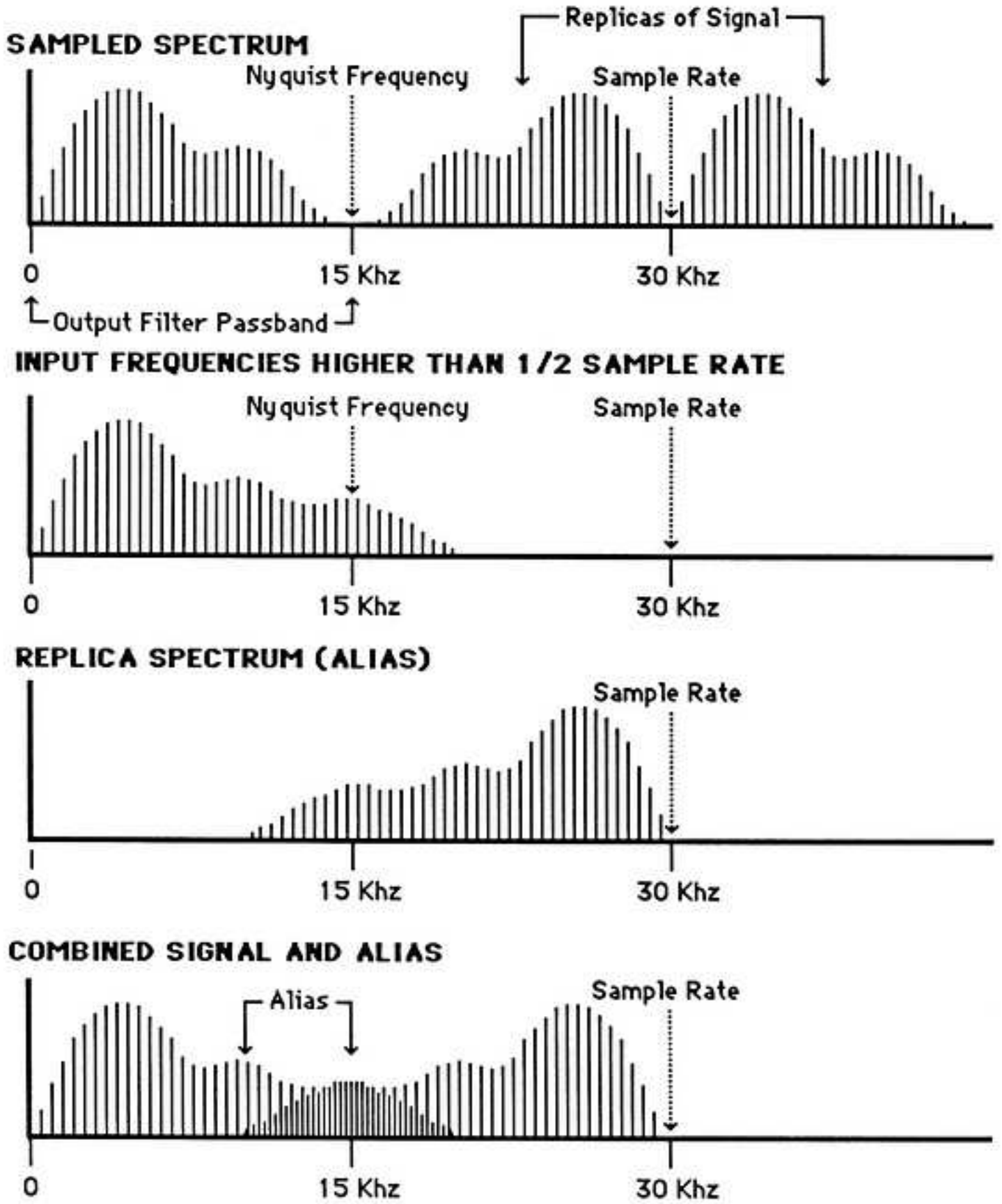
So there is a very simple tradeoff involved in choosing sample rates: sample duration versus frequency response. Sounds with short loops, however, have special requirements for very specific sample rates, as we shall see.

ALIASING

Aliasing is a kind of noise or distortion which can be generated by sampling. It is perceived as a harshness or "rattyness" in a sound. On pure tones, it can sound like odd, almost ring-modulated, undertones. An alias is created when the input sample rate is not high enough to capture the highest frequencies in a sound. If any frequencies above one half the sample rate are present in the input signal, they will 'alias' when they are sampled. The input filter, correctly adjusted, prevents this by removing the offending signals.

Aliasing is also called 'fold-over distortion', a term which is a more descriptive of how it occurs. Whenever any sound is sampled, the sampling process generates many replicas of the sound in different frequency ranges. One of these replicas starts at the sample frequency and goes <u>downward</u> in frequency. This is usually not a problem as the output filter will remove anything above the Nyquist frequency. If, however, the original spectrum had frequencies above one half the sample rate, these

will generate new frequencies below one half the sample rate. The higher the original frequencies were, the lower the new frequencies will be. The high frequencies have 'folded over' and invaded our signal, causing the roughness and noise in our sound. We can't remove them with the output filter because they are actually lower in frequency than the highest frequencies we want to reproduce. Bringing the filter down to remove the alias tones would also filter out the desired signal! Aliasing is prevented by eliminating these high frequencies before they are sampled.



The Mirage uses a custom-designed *VLSI* Integrated Circuit created by **ENSONIQ**, called the **Q-chip**™. This chip contains the oscillators, mixers and DCAs used in the Mirage. The Q-chip makes low-cost digital sampling possible. It does, however, have certain idiosycracies which make it very easy to get very bad sounds if you don't understand how things work.

All digital systems have to deal with aliasing. Input aliasing is caused by trying to sample frequencies which are higher than half the sample rate. Input alias frequencies become part of the signal; there is no way to get rid of them once they are there. They are prevented by filtering the input so that those high frequencies never get sampled. Output aliasing is always present; it is a replica of the signal spectrum, transposed up higher than the Nyquist frequency. All digital systems have a sharp filter on the output to eliminate any audible output aliasing. Most of the frequencies are above 15KHz, which are inaudible, anyway.

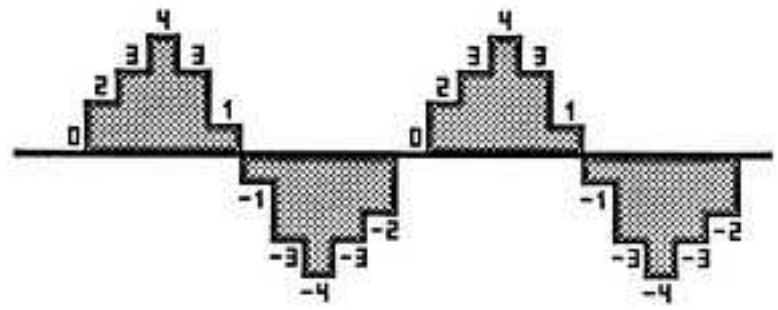
Digital oscillators like those in the Q-Chip also produce a third kind of aliasing, called a 'table-lookup alias'. This noise can be extremely troublesome when it shows up in the audio spectrum. It is a function of the input sample rate, the input signal, the output sample rate and the amount of transposition--that makes it rather hard to predict. The ways to control it are fairly straightforward, however. The strength of the alias is directly proportional to the ratio of the input frequency to the input sample rate, therefore, using a higher input sample rate will always improve the signal-to-noise ratio of a sound with strong high frequencies. The section on virtual sample rates shows how to achieve the effect of high sample rates by using a variable speed tape recorder.

QUANTIZATION

Since a sample is a number representing a signal at a certain point in time, we can ask how well it represents the signal at that point. In any digital system, the signal is *quantized*; that is, there are only a discrete (limited) number of different levels that the samples may assume. In the Mirage, samples can range from +127 to -127, or 255 different values. If the amplitude of the signal being sampled falls between two of the available values, the Mirage chooses the nearest available value. The difference between the actual value and the chosen value is called *quantization error* and it exists in <u>all</u> digital systems. Normally quantization shouldn't concern the Mirage user, but it's good to know what it is. The Mirage uses an *8-bit Floating-point* system to represent sounds. What this means is, there is an 8-bit sample and an 8-bit volume which, when multiplied together, form a 16-bit result for each oscillator. The 16 oscillator outputs are added together to form the final output. This adds the equivalent of another 4 bits of dynamic range. In theory, then, the Mirage has 20-bit dynamic range, or 120 dB. Actually the Mirage will distort if you try to play 16 sine waves, at maximum volume, all in phase, at the same time. Don't do that.

Quantization error results in noise which is most obvious when the signal recorded in memory is at a low level. For example, if a nice smooth sine wave were

sampled at too low a volume, the sample values might only go from +4 to -4. In memory, the wave would look very stepped:



On playback, there would be a "fizzle" sound in the background. This happens because there is not enough vertical (amplitude) resolution to accurately represent the waveform. In normal, full-level recording, quantizing noise is also present, but it is 52 dB quieter than the signal, making it virtually inaudible.

The only time quantization noise may be a problem is on sounds with both quiet and loud parts. The best solution is usually to use a compressor to even-out the signal, and then use the envelope generators to put the dynamics back in on playback. This is effective for getting the "fizz" out of the end of a drum sound, for example.

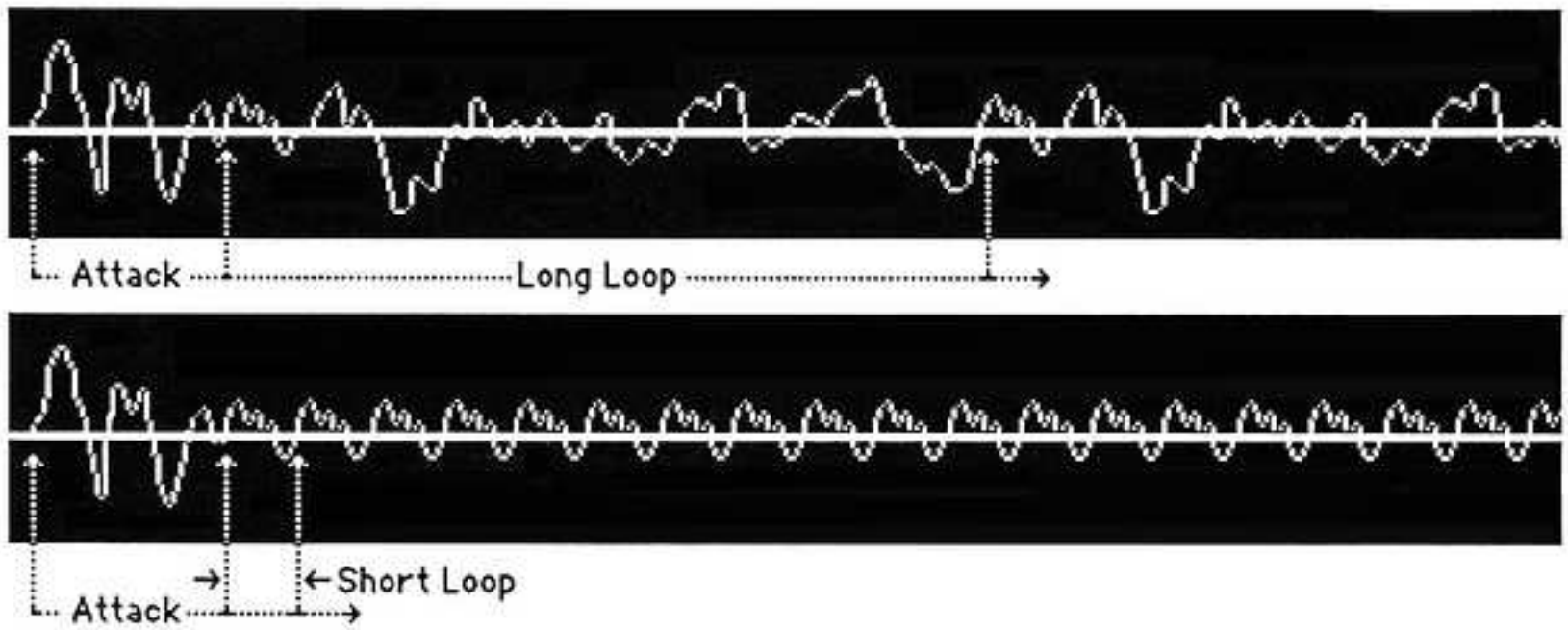
LOOPING

Finding good loops is one of the most difficult parts of sampling. Looping is not a dirty word; every sampling instrument, even the $37,000 ones, use looping to sustain notes. Looping takes advantage of the acoustic property of most musical sounds: after a complex attack portion they settle down into a smooth, predictable sustain portion.

There are really only two useful types of loops: long loops and short loops. A short loop is usually a loop on a single cycle of a waveform. This type of loop will always have pure pitch, since the repetition is absolutely perfect. The harmonic content of the waveform, no matter how complex it may be, is static over the duration of the loop. To add motion to the sound we can use two oscillators slightly detuned, or perhaps sweep the filter. This works remarkably well on most sounds which have very clear pitch content. The Mirage defaults to a short loop.

The other type of loop, the long loop, is better for sounds which have more dispersed pitch content. For example, a string section of 14 violins must use a loop long enough to capture the vibrant chorusing of 14 different notes, since none of the violins will be exactly in tune with each other. The longer the loop, the better.

Your ear is very sensitive to repeated patterns of 2 to 10 per second, since this is the range of beats and tempos. Loops that repeat at that rate are rather annoying and are very hard to hide. Loops of one second or more, however, are less noticable and are easily masked by other activity in a musical piece. That's why there's no middle ground; loops are either short or long. Of course, there are always exceptions.



Long Loops

Long loops are usually used with large wavesamples; those occupying all 64K or perhaps 32K of memory. They are needed for sounds like strings and vocal chorus, in which the sustain section is a rich texture rather than a pure pitch. In these cases you would like the loop "splice" to be as subtle as possible. That means finding two points, the Loop Start and Loop End, that are nearly identical in pitch, volume, and tonal quality. Another use for long loops occurs with sounds which are actually "events". For example, a cuckoo clock might have a loop containing a single "cuckoo". In this type of sound, the loop points are defined by the desired rhythm of repetition.

There are two steps to finding a long loop:

1) Find the general page location of the loop points.

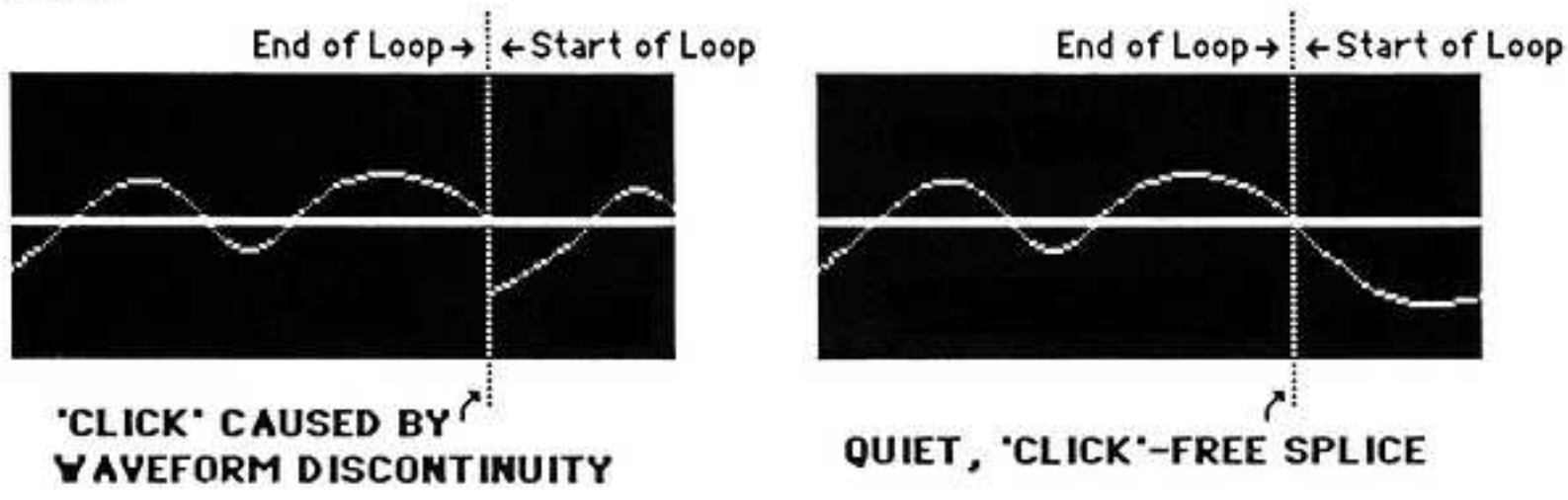
2) Fine tune the loop end to eliminate clicks.

The general location of a loop is estimated first. The decision is based entirely on the type of sound, and what you want to do with it. If you have sampled a measure of a musical piece, and want it to loop in rhythm, then there isn't much choice. A typical instrument sound is more difficult, since we would like the splice to be inaudible. Suppose you have sampled a voice singing the word "blah". You would like to use a long loop to retain the breathiness of the "ah" portion. Since we would really like the loop to be as long as possible, begin by moving the Loop Start **[62]** all the way to the beginning of the sound (so that it is the same as the Wavesample Start **[60]**), then turn the loop on. Now start increasing the Loop Start value, moving the starting point towards the end of the wavesample. As you move it, keep replaying the sound and listen to the result. We can expect there to be a click at the splice point in the loop, but we are listening for changes in the tone (vowel). As we move the Loop Start away from the start of the wavesample, the sound changes from:

blah-blah-blah-blah-blah
to
blah-lah-lah-lah-lah-lah
to
blah-ah-ah-ah-ah-ah-ah

Note that you must restrike the key each time to hear the change. What we are looking for is the place where the initial consonants (the "bl") end and the vowel (the "ah") begins. You can drive back and forth to find a place where the tone of the "ah" at the beginning of the loop seems to match up with the tone at the end of the loop. There will probably still be a click. This process will be similar for most other sounds; you look for the part where the initial transients have died away and the "steady- state" portion begins.

The second step is to adjust the Loop End Fine Adjust **[64]**. Start with the assumption that there is some place in the current Loop End page where the waveform will match up with the waveform at the beginning of the loop. That will be a good loop. Explore different Loop Fine values while listening for the click (you can drive the Loop End Fine back and forth while holding a key. You do not have to retrigger the note each time you make a change, although you may hear some strange glitches while moving the loop). You can develop an ear for when you are getting close to a good match-up. A perfect loop, where the splice is inaudible, is rare. You must also try playing the sound at different pitches; sometimes a loop will click on one pitch but not on another.



It would be nice if you could always find a good loop, but sometimes there is no loop point that works! Even systems with "auto-looping" can't find a good loop if none exists. If you can't find a loop, its best to try again somewhere else. Usually you would move the Loop End parameter down one page and start again. Sound tedious? It can be! The **Visual Editing System** makes this all much easier by allowing you to see the waveform on the computer screen. If you have no luck moving the Loop End, try a different Loop Start and begin hunting for a loop again. If that doesn't help and you can't find a good loop anywhere in the entire memory, you will have to resample the sound and start all over. Good luck.

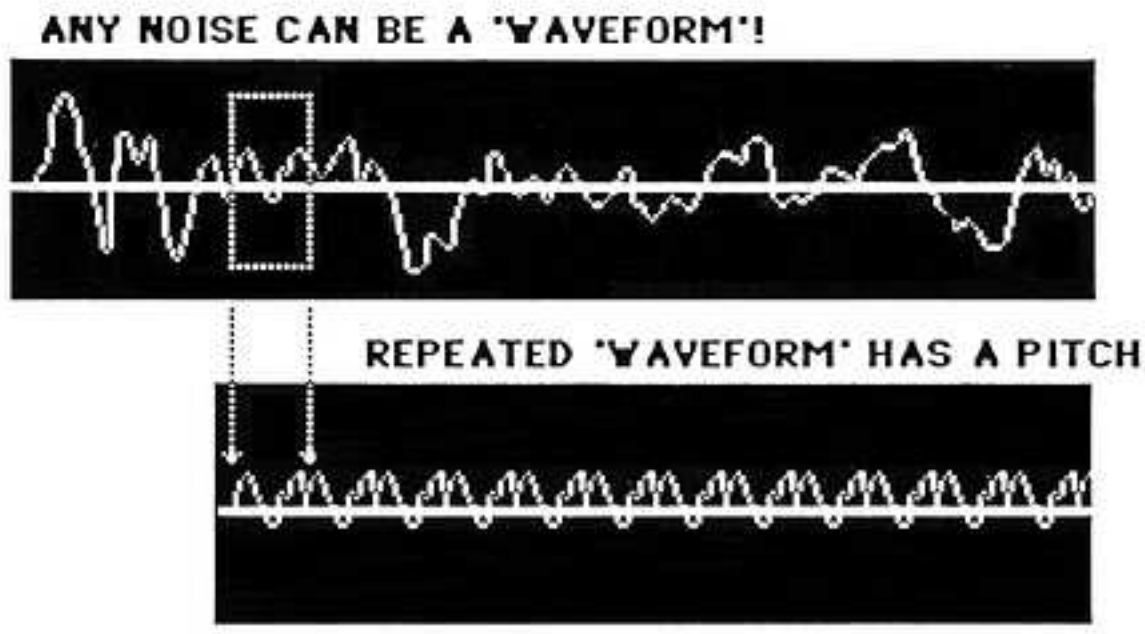
If you do have the **Visual Editing System**, you can look through the waveform to find a good place to start the loop. There are several generally applicable criteria for

calling a spot "good". First of all, it should be a point in the waveform after all the initial transients in the attack have died down. Second, it should be a zero crossing. This means, on the page where the loop starts, the first few samples should have amplitude values of zero or near zero (middle scale). Splicing at zero amplitude will help cover up the splice, reducing clicks. Third, the start page should be a "round" number, if possible. Page numbers ending in 0 or 8 are best; those ending in 4 or C will do; even numbers are better than odd. All this has to do with some subtle internal workings of the Q-Chip. Tiny pitch glitches can occur when the oscillators cross over certain memory boundaries. If you can't find a desirable page that starts on a zero crossing, you can try rotating the data until you line up a zero crossing with the page boundary.

Short Loops

Short loops are quite a different animal. The period of the loop becomes so short that we can't pick out individual repeats. The repitition is so rapid that it becomes a pitch. The cycling of the loop will become the pitch of our sound. With a short loop, a sound has two very distinct parts: the attack portion and the loop portion. This "attack" portion should not be confused with the attack segment of the envelope generator. The attack portion of the sound is the part of the wavesample which is heard only once, immediately before the loop. The pitch of the attack portion is determined by the sound we sampled.

The pitch of the loop portion, however, is decided solely by the length of the loop. You can prove this to yourself by sampling any old noise and turning on the default loop, which is one page long. When the noise plays out and gets to the loop, suddenly you will hear a constant pitch. The loop takes whatever data happens to be in that page of memory, and turns it into a repeating waveform. The spectrum of that waveform, the tone-color, probably has no resemblance to the original sound and the pitch is most likely different from that of the attack.



Usually we would like a short loop to have a great resemblance to the original sound! To do that we have to line up the length of the waveform with the length of the loop. Most short loops are exactly one page long, therefore our task is to sample the

sound in such a way that each waveform cycle in the sound is exactly one page (256 samples) long. This means that our sample rate should be 256 times the frequency of our sound source.

To achieve this relationship we can change either variable: the input sample rate or the frequency (pitch) of our sound source. In some cases, we may need to change both.

The main steps to getting a short loop are:

1) Choose a sample rate which corresponds to the pitch to be sampled. The second sample rate chart in Appendix A gives the nearest sample rate to the equi-tempered scale pitches. When you set one of these sample rates, the Mirage will "expect" a certain, exact pitch for input.

2) Adjust the location of the loop, in pages, to the desired position in the sound where the attack transients have died away and the timbre is relatively constant. The default loop position, on the next to the last page, is often the best. Use a one page loop whenever possible. Do <u>not</u> adjust Loop End Fine **[64]**, leave it set at FF (very nasty things can happen if you make the loop shorter than a page).

3) Fine tune the pitch of the <u>source</u> to match up to the pitch that the Mirage "expects". This requires resampling and listening for a pitch change between the attack sound and the loop, as we saw in the sampling examples.

This last step, fine tuning, is the most laborious. Note that the Mirage does not "expect" the source to be in perfect, concert pitch. The Mirage has a limited number of different sample rates, and each of these rates corresponds to some expected input frequency; those frequencies don't quite line up with the notes that we call "in tune to A440". For example, when the sample time is set to 35 (µsec), the sample rate is 28,571 Hz. The pitch that the Mirage "expects" is 1/256 of this, or 111.6 Hz. True A concert is 440, 220, or 110 Hz. Since we can't adjust the Mirage sample rate any closer, we have to raise our sound source by 1.6 hertz. In that pitch range, a semitone happens to be 6.5 hertz, so we're off concert pitch by about a quarter of a semitone, or 25 cents (that's 5 steps on the Mirage master tune control). You can retune the wavesample after you have sampled the sound to get back to concert pitch.

Lengths of Short Loops

So far we have talked about short loops as single page loops. A single page loop is one where the Loop Start is set to the same page as the Loop End, and the Loop End Fine is set to FF. This means that the loop is exactly 256 samples long. For certain sounds, particularly low-pitched sounds, you may need a longer loop, as one waveform cycle may be more than 256 samples long. The Mirage will try, but do a bad job, of playing short loops that are not exactly one, two, or four pages long (256, 512, or 1024 samples). The preferred loops are:

1) Loops of exactly one page (Loop Start=Loop End, Loop Fine = FF);

2) Loops of exactly two pages, where the first page is an even number (the page number ends in 0,2,4,6,8,A,C, or E);

3) Loops of exactly four pages, where the first page number is divisible by 4 (page ends in 0,4,8, or C).

If a short loop is not one of these cases, the Mirage will have some problems playing it. Problems usually occur when the sound is played at higher pitches. These evil loops may be detected by their ability to cause the display to temporarily blank out completely when several keys are pressed--the Mirage is kept very busy trying to play these loops. More keys or higher notes will cause the condition to deteriorate, until the Mirage decides that enough is enough, and turns the notes off. These loops will also go out of tune as they go up in pitch, and the pitch, in general, will be less accurate.

The bad loops to be avoided are:

1) Any loop less than 8 pages with a loop end fine adjustment other than FF.

2) Any 2 page loop that starts on an odd numbered page.

3) Any loop of 3, 5, 6, or 7 pages.

In short: do not use Loop Fine Adjust **[64]** on short loops (leave it at FF) and start all loops on even boundaries in memory.

Sampling A Harmonic of the Loop Frequency

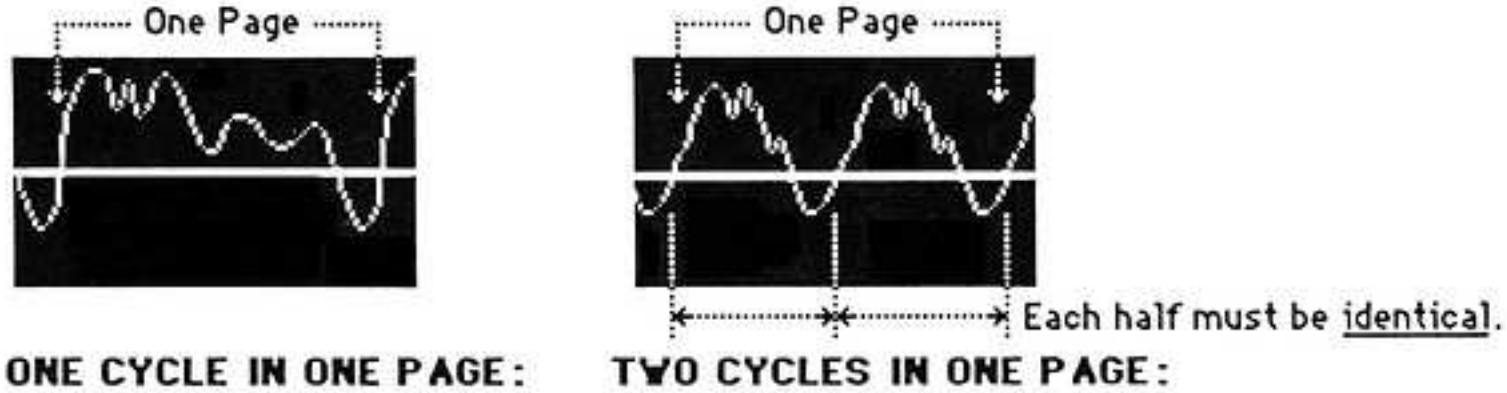
A single page loop is the smallest useful loop on the Mirage. At its highest sample rate, the Mirage "expects" a fairly low note, the C below middle C. To sample middle C, an octave above, we can use the same sample rate but an interesting thing happens. Two complete waveform cycles will fit into a page of memory (each waveform will be 128 samples long). Now when we loop, we are repeating those two waveforms. Similarily, if we sample a note two octaves above the 'expected' pitch, we are putting four waveforms into each page of memory.

In practice, this means that when you sample high frequencies, there will be subharmonics in a short loop unless the source is exactly lined up with the sample rate. This is caused by any differences in the waveform cycles that will repeat at the loop rate (which may be many octaves below the frequency of the signal).

Occasionally you may encounter a waveform, like the piano, which has subtle phase variations from one waveform cycle to the next. This is caused by the fact that there are really three strings vibrating on a piano note. Finding a 'multiple

wave-per-page' loop may be impossible. Even two waveforms, right next to each other on one page, can be different enough to cause a subharmonic when looped. You must force the waveforms to be identical by making the second one an exact copy of the first. Use the Copy function of MASOS, which can copy any size chunk of memory. Set the Source to be the first half of the page: XX00 to XX7F (Hex). Set the Destination to be the second half of the same page: XX80 to XXFF (Hex).

The same procedure (which is identical to waveform replication on boundaries less than a page) can be performed on smaller waveforms, such as waveforms that are 64 samples long (4 waveforms/page). Best results occur if you stick to an even number of waveform cycles in a page (don't try to fit 3 waveforms in a page, it won't work as well, even if it seems like it should). Of course, the more waveforms you cram into a page, the less samples for each waveform can be stored, which can lead to distortion.



ONE CYCLE IN ONE PAGE: THE FUNDAMENTAL

TWO CYCLES IN ONE PAGE: THE SECOND HARMONIC

SAMPLING TECHNIQUES

Virtual Sample Rates

There are two ways to get the effect of a higher input sample rate. One is, obviously, to increase the sample rate. The other is to transpose the input down in frequency; lowering the pitch of the source. The effect is the same: There are more samples taken for each waveform cycle. This gives a better 'picture' of the signal.

Certain electronic sounds, like synthesizer oscillators, can be transposed (by retuning) and still retain exactly the same harmonic content. Most acoustic instruments, however, have markedly different harmonic content when played at different pitches. A multi-speed tape recorder is needed to electronically slow the sound down while maintaining the same relative harmonic content.

Using a Tape Recorder

A variable-speed, or at least a two-speed, tape recorder is an invaluable tool for the advanced sampler. By recording at 7-1/2 ips, for example, and playing back at 3-3/4 ips, we transpose a sound down one octave. Sampling this signal at 25KHz will give the same result as if we sampled the original sound at 50KHz.

Another reason for having a variable-speed tape recorder is for pitch adjustment. Any sound which uses a short loop must be sampled at a rate which is exactly 64, 128, 256 or 512 times the fundamental pitch. This will result in waveforms which are exactly 64, 128, 256 or 512 samples long. The Mirage does not have a 'fine tune' on the input sample rate; only certain rates are available. You must therefore tune your sound to one of the pitches the Mirage will accept. See the section on short loops (page 57) for more on this.

A third reason for using a tape recorder while sampling is repeatability. No matter how expert a flutist may be, one could never play the same note the same way twice. If you are trying to get a looped sound, the exact frequency of the source is critical. There is nothing you can do to a wavesample, once it is in the machine, to fix an incorrect source frequency to sample rate ratio. In acoustic instruments, especially wind instruments,tiny pitch fluctuations occur constantly and the sound may be in tune on the 20th cycle of the waveform, then drift out of tune by the 40th. With a taped sound you can deal with the detail of the sound because it is exactly the same each time you play the tape.

Sample Rate Fine Tuning

Suppose you wish to sample the low C on a gigantic pipe organ. The organ was tuned to concert pitch, so the low C is 32.703 Hz, a very low note indeed. To sample that note at 512 times the fundamental will require a sample rate of:

32.703 Hz	*	512	=	16,744 Hz
(cycles per second)		(samples per cycle)		(samples per second)

That's all well and good, but when we go to our Mirage sample rate chart we find that the nearest available sample rate is 16666.6 Hz; not close enough for a good clean loop. One solution is to tune the organ pipes to the "expected" frequency, which is:

$$\frac{\text{16,666.6 samples per second}}{\text{512 samples per cycle}} = \text{32.552 cycles per second}$$

Now if you are too lazy, or for some other reason don't wish to retune a pipe organ, you could, instead, use a vari-speed tape recorder. Anything that has plus-or-minus a semitone of pitch control will do. Record the organ note with the pitch control at center. Move the pitch control down when playing back, making the sound ever so slightly flat. The rest is trial and error, based on listening to the sampled sound with the loop turned on; retuning the tape until the attack is in tune with the loop.

Equalization

Sampling is more similar to recording than to synthesis. The skills and techniques used by recording engineers are just as important in achieving a good sample as they are in achieving a good sounding record. At first you may think that if you put a microphone in a room with a piano and sample middle C, you will have an accurate piano sound for the Mirage. In fact, there is no one "true" piano sound, and recording engineers still argue over the best places to put which microphones to get the "best" sound. To make things worse, a "true" piano sound on one particular note doesn't sound as good when transposed up or down; things like "action noise" or "frame noise" cause problems; and all the high strings ring when you hit one note.

The moral is: to get a "true" sound out, you don't (necessarily) put a true sound in.

Recording engineers are used to adding equalization to sounds to help them work in a musical way. Many sounds need to be exaggerated or distorted in some way so they sound right when mixed in with other sounds. A kick bass drum is often equalized on records to bring out the "kick" part of it: a boost at 4KHz. It doesn't sound like a drum anymore, but it sounds good in the context of the music. In any case, the same creative approach must be used when sampling sounds. In this case sounding good "in context" means "when a sound is played on the Mirage keyboard". Equalization can greatly enhance a sound by bringing out the essence of its character and by suppressing noises which, although they exist in the real instrument, don't work for a digitally sampled imitation of it (for example, action noise in a piano is part of the sound, but, in a real piano the action noise is constant. If you transpose a piano sample which contains action noise, the noise will transpose in pitch as well, producing a strange thump that tracks the pitch of the note).

Equalization is achieved by putting an equalizer between the signal source and the Mirage sampling input. The settings for the EQ must be derived experimentally. Here's a convenient way to do sample equalization:

1) Sample your sound flat, with no EQ.

2) Get most of the Mirage parameters set up the way you expect to use them.

3) Put an equalizer, preferrably parametric, on the Mirage output.

4) Play the key on the Mirage that plays the sound back at the pitch it was sampled. Adjust the equalizer while playing the key to get closer to the sound you would like. Boost the highs, take out some midrange, notch out a nasty harmonic; whatever.

5) Remove the equalizer from the Mirage output and put it on the input; between the sound source and the Mirage.

6) Resample the sound. It will now have the desired EQ 'built-in' to it.

One of the difficult aspects of multi-sampling is matching up the different samples across the keyboard. Equalization can help smooth out the differences between adjacent samples.

Another use of equalization is to help out the input anti- aliasing filter. If you have a sharp low pass filter available, you can increase your effective bandwidth by using two filters at a higher frequency instead of the single internal input filter. The additional filter increases the roll-off slope at the input, allowing the same attentuation of high frequencies at a higher cutoff frequency than is possible with the lesser slope.

In addition, the Mirage is not a "flat" reproduction system. There is a filter on the input, and a filter on the output. Both filters must usually intrude on the signal's high frequency components somewhat. If this is a problem, the solution is to "pre-emphasize" the signal before sampling, much as tape decks and other signal processing equipment do.

Dynamic Filtering

Dynamic input filtering can be useful for certain sounds. This means changing the external filter setting during the course of the sample, either by hand or automatically. For example, If you want to let the bright attack of a plucked string through, but want to loop on a more mellow waveform, you can manually sweep the EQ during sampling so the sound loses its high frequencies faster than it would normally.

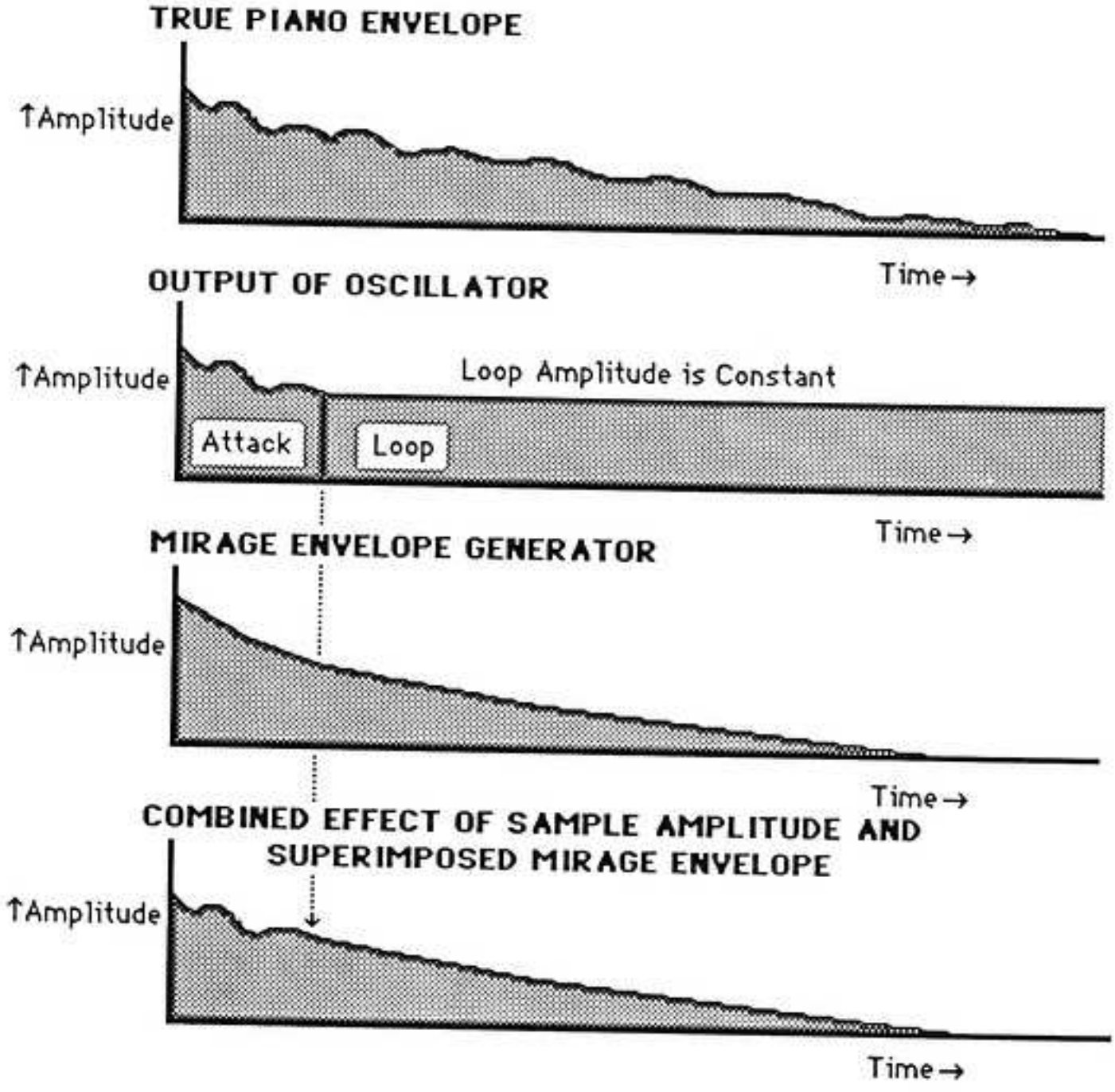
Dynamic output filtering is created with the Mirage filter envelope. You can simulating the natural muting of tone quality as a sound decays even if you're looping on a "bright" waveform. You can create the acoustic effect of increasing brightness with increasing key velocity (provided, of course, the sound is bright to begin with, the filter can't add brightness, only remove it). In some cases you can use the filter as a "noise gate" to hide quantizing noise as a signal dies away. Don't be afraid to use to filter to achieve the desired sound--in most cases the best sound will <u>not</u> be achieved with the filter "wide open". In all cases, any sound above half the sample rate can <u>only</u> be noise and should be filtered out.

Volume Dynamics

The Mirage amplitude envelope performs different functions for different kinds of sounds. In the piano, for example, the envelope attack is instantaneous and the duration of the sound occurs in the decay stage of the envelope. An actual piano envelope is more complex than that, particularly in the first second of a note. In the Mirage, the wavesample itself reproduces the complex attack, then starts to die away.

When the wavesample hits the loop, its amplitude becomes fixed, and the envelope generator takes over to finish the sound. The envelope generator creates the smooth decay to silence. In a drum sound, on the other hand, the decay is part of the wavesample, the envelope does little more than turn the sound on and off. Using the envelope generator it is possible to create sounds which occur only while the key is held, or which continue on long after a key is released. The sound itself will determine how the envelope should be used. In most cases it is desirable to loop on a high amplitude portion of the wavesample to insure good signal-to-noise. In these cases, the envelope generator must be used to make the sound decay to silence.

Typically, it is a combination of the envelope generator and the dynamics within the wavesample which contribute to a realistic sound. Careful use of the envelope generator can give a better signal-to-noise ratio to a sound. For example, if you wanted to create the sound of a crowd cheering, you may want it to swell slowly. One approach is to sample a crowd sound as it begins to swell. The beginning of that sample will be at a low level and some quantization noise could be audible. A better approach is to sample the crowd at full tilt; maximum level for the whole sample. You can then synthesize the swell of the crowd by increasing the attack time of the envelope generator. You can even make the attack time velocity sensitive. This approach gives you both a better signal-to-noise ratio and a longer portion of sound to loop on. Use the envelope to model the contour of the sound.



A compressor or a peak limiter will automatically compensate for changing levels in a signal. You can use one to effectively eliminate any envelope from the sound you are sampling making it easier to loop (rapid fluctuations in amplitude are almost as noticable as pitch fluctuations). The envelope generator can then be used to recreate the shape of the sound. This is not always realistic, since the envelope generator may not reproduce all the subtle fluctuations that the compressor took out.

SAMPLE DATA FORMAT

If you are using a computer to look at the Mirage waveforms, you may be interested in the data format. Inside the Mirage, each sample is represented by a byte. The value 80 (Hex) corresponds to a zero 'DC level' in the waveform; it is mid- scale. The amplitude can go from a level of +127 (FF Hex) to a level of -127 (01 Hex). The value 00 has a special function: it acts as a marker which tells the oscillators where to stop. If the loop is on, 16 zeros are inserted into the wavesample memory immediately following the loop. Otherwise the 16 zeros are put at the very end of the wavesample data. This is why loops cannot extend to the last byte of the last page of a wavesample. Normally, you should turn the loop off before transferring data from the Mirage to the computer as you will end up with undesirable zeroes in your data.

QUICK ADVICE

If you can't hear a particular wavesample and don't know why: The wavesample may be hidden by another--check Top Key. Mix-mode or Initial Wavesample may be affecting the key assignment too. Wavesample Relative Amplitude may be too low, or at zero. Some evil sorcerer may have set the wavesample Max Filter Freq parameter to some low number.

Turn off all loops before starting a multi-sample. One quick way to do this is by sampling with the user Multisampling switch off. This effectively "cleans out" all of the wavesamples and resets them (for upper or lower).

To trim the end of a non-looped sound, it is necessary to turn the loop on and off once after moving the Wavesample End parameter.

Short loops should be exactly one page long whenever possible. Avoid using Loop End Fine on short loops. Try to start loops on even page boundaries.

Start Wavesamples on large boundaries in memory. It's better to start a sound at 80 than at 81 or 82.

Use the line input setting for sampling, since the microphone input has a compressor built in. You can always add external compression, but you can't remove the effect of the internal compressor if you use it.

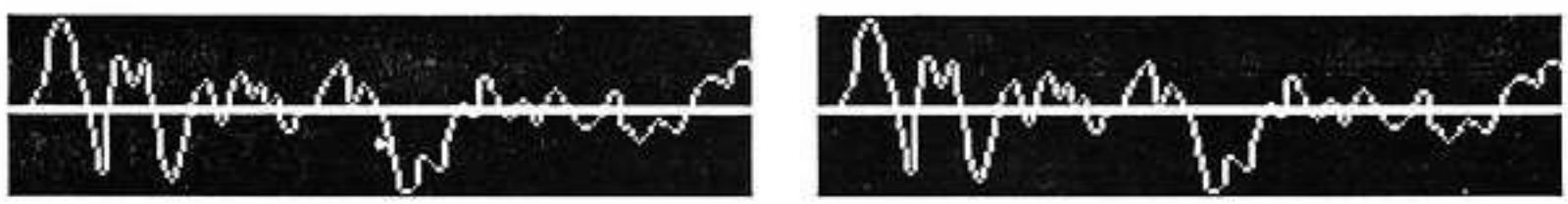
DIGITAL SIGNAL PROCESSING

Once a sound has been sampled, there are many techniques for manipulating it, which can be lumped under the label of digital signal processing. These include simple operations like moving samples from here to there, or sophisticated processes like digital filtering. MASOS includes a small set of operations which are very useful for sampling and creating loops. This section will discuss those and some other processing techniques which are not available in the Mirage, but which could be implemented in an external computer.

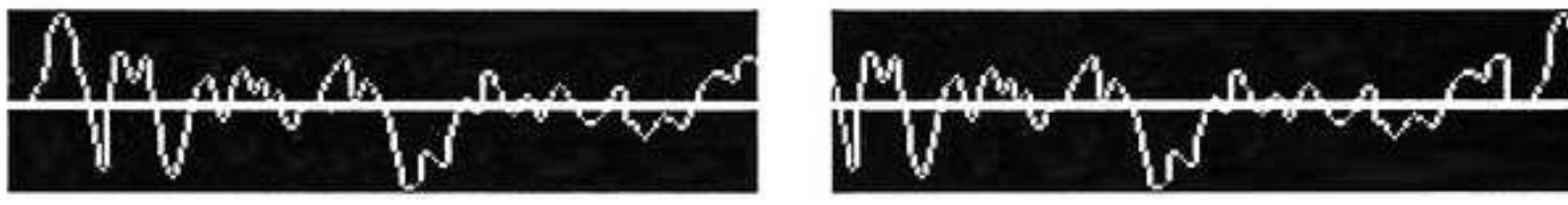
Time Domain Manipulations

The simplest thing we can do with samples is to move them around in memory. We can treat our sampled sound as a string of data, without knowing much about where it came from. The basic operations we need are provided by MASOS. They are:

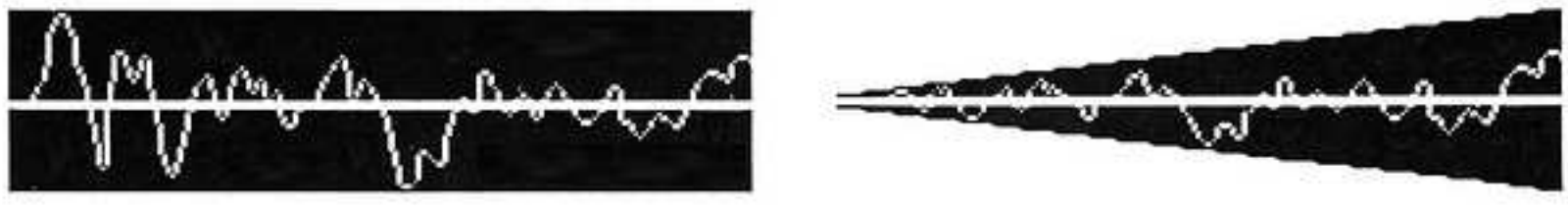
Copy - copy samples from one section of memory to another. Used for making copies or back-ups of waveform data. Also good for making multiple identical waveforms in a single page, as explained in the section on Short Loops (page 60).



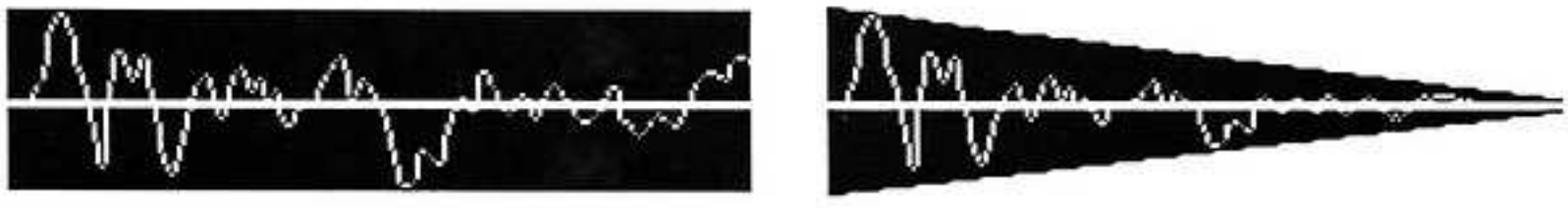
Rotate - shift the samples up or down in memory, but keep all the data. Take the samples which fall out one end and insert them in the other end. Rotate left to get rid of silence or noise at the beginning of a wavesample. Rotate data to line up loop points with zero crossings in waveform data.



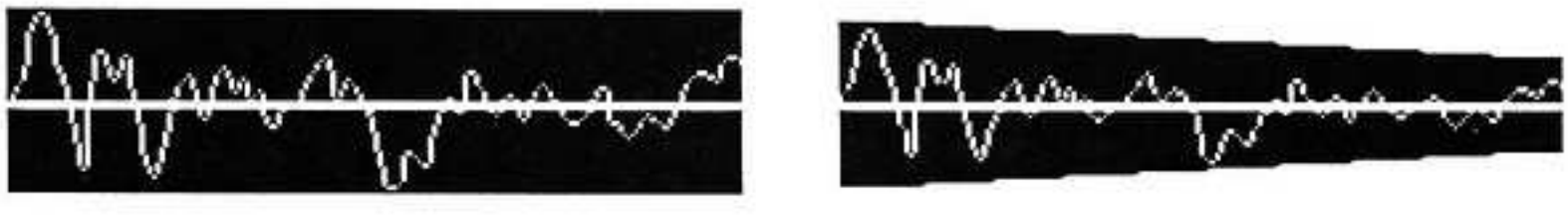
Fade in - multiply the sound by a ramp going from zero to one. Create a fade-in, or one part of a cross-fade.



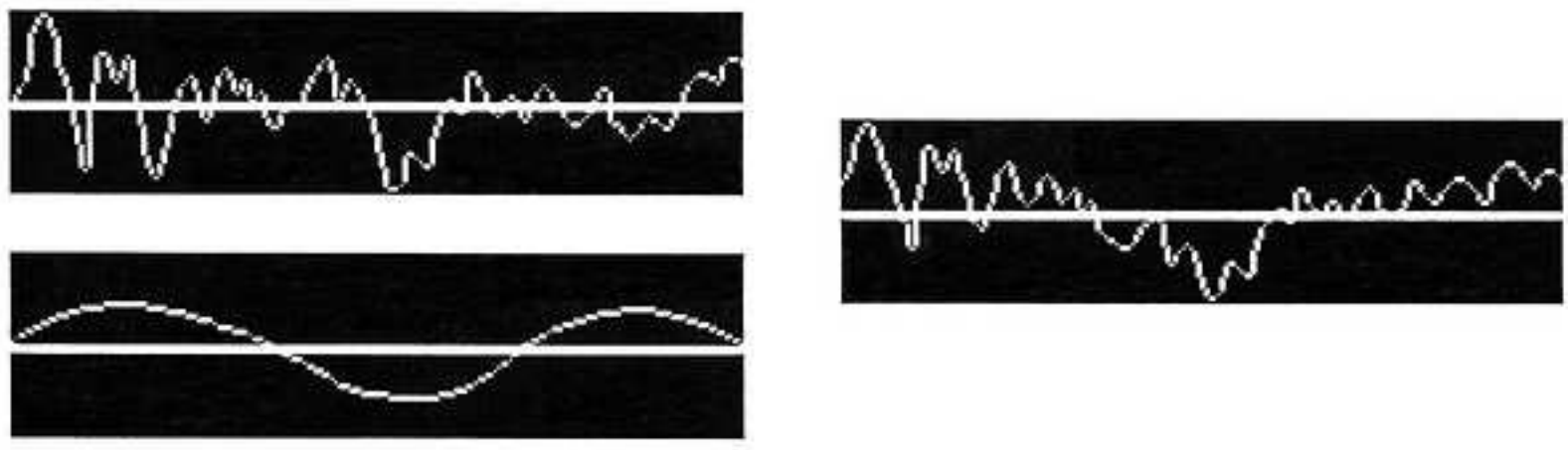
Fade out - multiply the sound by a ramp going from one to zero. Fade out non-looped sounds (such as orchestra hits) or create one part of a cross fade.



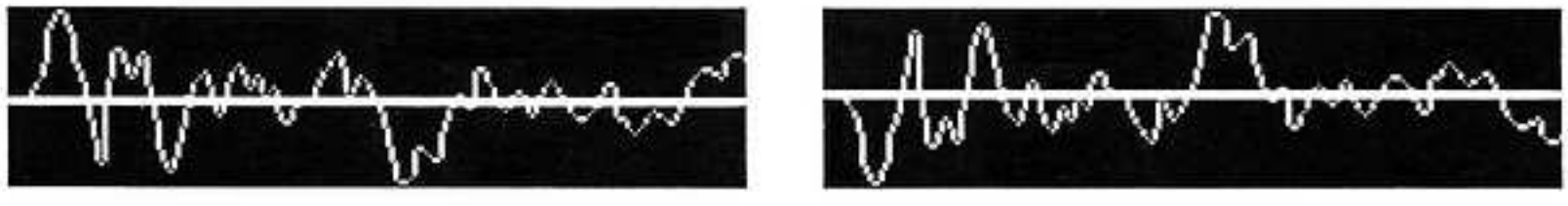
Ramp scaling - multiply each sample by the corresponding point on a linear ramp, causing a gradual change from one volume to another. A general fading function used to create non-linear fades by combining many waveform sections with different fade ramps. When the ramp Scale Start and End Factors are equal, the sound is attentuated by a fixed amount. Also used to balance the volume of the beginning and end of a loop.



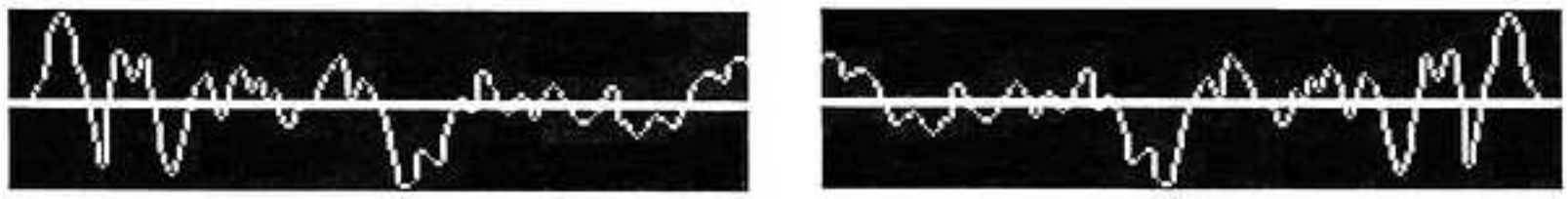
Add - add two sounds together, sample by sample. This is the equivalent of mixing two sounds at equal volumes.



Invert - reverse the sign (polarity) of each sample. Positive numbers become negative numbers, and vice versa. Can be used to splice a waveform with a Reversed copy of itself, allowing bidirectional looping.



Reverse - reverse the order of the samples. Plays a sound backwards or can be used to create bidirectional loops.



Replicate - make repeated copies of the same waveform. Eliminate subharmonics in loops by making all pages the same.

By combining these basic operations, you can achieve various other manipulations. For example:

Cross fade - fade from one sound into another. This is accomplished by adding one sound which fades out to another which fades in.

Waveform merging - create blends of two waveforms. Replicate each waveform seperately and perform a cross fade between the resultant wavesamples.

Amplification - a waveform can be increased in level by adding it to itself. For smaller amplifications, add it to a scaled copy of itself.

Envelope correction - correct for differences in level between loop start and loop end points. Arbitrary level control can be achieved by stringing together many small scaling ramps.

Bidirectional looping - the first half of a loop can be copied, reversed and inverted so the splice points at both ends line up perfectly.

Frequency Domain Manipulations

Since we usually think of music in terms of frequency, it makes sense to be able to deal with sampled signals as frequencies. Unfortunately, most of the operations are an order of magnitude more complex than waveform manipulations and require more sophisticated software than is provided with MASOS.

Envelope detection - In the analog world, this is usually accomplished by rectifying a signal and passing it through a low-pass filter to separate the envelope

from the signal. A simulation of this effect can be performed on the digital data. The resulting function can be used to modify the original signal, as in a compressor, expander, or limiter. It can be used, for example, to remove the decay of a piano note so the loop end is the same volume as the loop start.

Sample Rate Conversion - Data that has been sampled at one rate can be converted to another. For example, the data from a digital recorder is sampled at around 44 KHz. This data could be used in the Mirage, but for looping purposes, you may need to convert the data so the sample rate is a multiple of the pitch. When converting, new samples are created by interpolating between the original samples.

Digital filtering - You can impose an arbitrary filter response onto a sampled sound. This is useful for removing high harmonics which are below Nyquist, but causing a problem with playback aliasing. Also, a high-pass response is sometimes helpful for removing sub-audio frequencies which may become audible when transposed up.

Analysis and resynthesis - *Fourier analysis* breaks up a complex waveform into its component harmonics. It makes the assumption that the waveform is periodic, which it often is not. When used with the reverse process, *Fourier synthesis*, it provides a way of cleaning up loop waveforms. Even more sophisticated analysis tools have been developed, such as the *Phase Vocoder*, which can analyze arbitrary harmonic content and break it into sine waves with complex frequency and amplitude envelopes. Most of this work has been done at Stanford with large computers, but it should be possible on a micro, just slower.

Software synthesis - Many techniques have been developed to create sound waves algorithmically: Additive Synthesis, Frequency Modulation (FM), non-linear waveshaping, etc. All of these techniques can be employed to create sounds which can be used as wavesamples in the Mirage.

In short, the Mirage is one of the most flexible electronic music systems ever created. Since the Mirage is a general-purpose digital data player it has the ability to reproduce any sound via sampling. It can also create the sound of analog synthesizers, both by sampling and by processing with the internal synthesizer section. The Mirage can also create the sounds of digital synthesizers, both by sampling and by the algorithmic generation of digital waveforms. Complete access, via an external computer, to the digital data used in creating sounds provides a degree of creative freedom never before available to the average musician. **ENSONIQ** is committed to developing and enhancing the Mirage system into the most powerful and affordable digital music system ever offered--stay tuned for further developments!

APPENDIX A - TABLES

MIRAGE SAMPLE RATES

Sample Time [73]	Sample Rate	Nyquist Rate	Sample Rate/256
20	50000.00	25000.00	195.31
21	47619.05	23809.52	186.01
22	45454.55	22727.27	177.56
23	43478.26	21739.13	169.84
24	41666.67	20833.33	162.76
25	40000.00	20000.00	156.25
26	38461.54	19230.77	150.24
27	37037.04	18518.52	144.68
28	35714.29	17857.14	139.51
29	34482.76	17241.38	134.70
30	33333.33	16666.67	130.21
31	32258.07	16129.03	126.01
32	31250.00	15625.00	122.07
33	30303.03	15151.52	118.37
34	29411.77	14705.88	114.89
35	28571.43	14285.71	111.61
36	27777.78	13888.89	108.51
37	27027.03	13513.51	105.57
38	26315.79	13157.90	102.80
39	25641.03	12820.51	100.16
40	25000.00	12500.00	97.66
41	24390.25	12195.12	95.27
42	23809.52	11904.76	93.01
43	23255.82	11627.91	90.84
44	22727.27	11363.64	88.78
45	22222.22	11111.11	86.81
46	21739.13	10869.57	84.92
47	21276.60	10638.30	83.11
48	20833.33	10416.67	81.38
49	20408.16	10204.08	79.72

MIRAGE SAMPLE RATES

Sample Time **[73]**	Sample Rate	Nyquist Rate	Sample Rate/256
50	20000.00	10000.00	78.13
51	19607.84	9803.92	76.59
52	19230.77	9615.39	75.12
53	18867.92	9433.96	73.70
54	18518.52	9259.26	72.34
55	18181.82	9090.91	71.02
56	17857.14	8928.57	69.75
57	17543.86	8771.93	68.53
58	17241.38	8620.69	67.35
59	16949.15	8474.58	66.21
60	16666.67	8333.33	65.10
61	16393.44	8196.72	64.04
62	16129.03	8064.52	63.00
63	15873.02	7936.51	62.00
64	15625.00	7812.50	61.04
65	15384.62	7692.31	60.10
66	15151.52	7575.76	59.19
67	14925.37	7462.69	58.30
68	14705.88	7352.94	57.44
69	14492.75	7246.38	56.61
70	14285.71	7142.86	55.80
71	14084.51	7042.25	55.02
72	13888.89	6944.44	54.25
73	13698.63	6849.32	53.51
74	13513.51	6756.76	52.79
75	13333.33	6666.67	52.08
76	13157.90	6578.95	51.40
77	12987.01	6493.51	50.73
78	12820.51	6410.26	50.08
79	12658.23	6329.11	49.45

MIRAGE SAMPLE RATES

Sample Time [73]	Sample Rate	Nyquist Rate	Sample Rate/256
80	12500.00	6250.00	48.83
81	12345.68	6172.84	48.23
82	12195.12	6097.56	47.64
83	12048.19	6024.10	47.06
84	11904.76	5952.38	46.50
85	11764.71	5882.35	45.96
86	11627.91	5813.95	45.42
87	11494.25	5747.13	44.90
88	11363.64	5681.82	44.39
89	11235.96	5617.98	43.89
90	11111.11	5555.56	43.40
91	10989.01	5494.51	42.93
92	10869.57	5434.78	42.46
93	10752.69	5376.34	42.00
94	10638.30	5319.15	41.56
95	10526.32	5263.16	41.12
96	10416.67	5208.33	40.69
97	10309.28	5154.64	40.27
98	10204.08	5102.04	39.86
99	10101.01	5050.51	39.46

Note: Sample Time settings below 30 are only available with the optional ***Input Sampling Filter*** *Cartridge.*

SAMPLE RATES FOR EQUI-TEMPERED PITCHES

Note Name	Freq	Freq*128	Sample Time	Sample Rate	Nyquist
a	110	14080	71	14084	7042
b flat	117	14917	67	14925	7462
b	123	15804	63	15873	7936
c	131	16744	60	16666	8333
c sharp	139	17739	56	17857	8928
d	147	18794	53	18867	9433
e flat	156	19912	50	20000	10000
e	165	21096	47	21276	10638
f	175	22350	45	22222	11111
f sharp	185	23679	42	23809	11904
g	196	25087	40	25000	12500
g sharp	208	26579	38	26315	13157
a	220	28160	35	28571	14285
b flat	233	29834	34	29411	14705
b	247	31608	32	31250	15625
c	262	33488	30	33333	16666
c sharp	277	35479	28	35714	17857
d	294	37589	27	37037	18518
e flat	311	39824	25	40000	20000
e	330	42191	24	41666	20833
f	349	44700	22	45454	22727
f sharp	370	47358	21	47619	23809
g	392	50174	20	50000	25000

INTERNAL INPUT FILTER CUTOFF FREQUENCY

Setting [74]	Frequency (in Hz)
50	1250
51	1324
52	1403
53	1487
54	1575
55	1669
56	1768
57	1873
58	1984
59	2102
60	2227
61	2360
62	2500
63	2649
64	2806
65	2973
66	3150
67	3337
68	3536
69	3746
70	3969
71	4204
72	4454
73	4719
74	5000
75	5297
76	5612
77	5946
78	6300
79	6674

INTERNAL INPUT FILTER CUTOFF FREQUENCY

Setting [74]	Frequency (in Hz)
80	7071
81	7492
82	7937
83	8409
84	8909
85	9439
86	10000
87	10595
88	11225
89	11892
90	12599
91	13348
92	14142
93	14983
94	15874
95	16818
96	17818
97	18878
98	20000
99	21189

EXTERNAL FILTER CUTOFF FREQUENCY

Setting **[93]**	Frequency (in KHz)
0	Off - uses internal filter
1	3.37
2	3.47
3	3.57
4	3.68
5	3.79
6	3.91
7	4.03
8	4.17
9	4.31
10	4.46
11	4.63
12	4.81
13	5.00
14	5.21
15	5.43
16	5.68
17	6.25
18	6.94
19	7.81
20	8.33
21	8.93
22	10.4
23	12.5
24	13.9
25	20.8

APPENDIX B - MIRAGE ADVANCED SAMPLING OPERATING SYSTEM (MASOS)

MASOS

The two diskettes supplied with this book contain the Mirage Advanced Sampling Operating System (MASOS). MASOS is an alternative to the standard Mirage Operating System which is provided on all Mirage Sound Diskettes and Mirage Formatted Diskettes. MASOS provides special functions which are useful to anyone sampling sounds. The Sequencer will not operate while using MASOS as the additional MASOS functions replace the Sequencer functions. The upper and lower sounds on the MASOS disk are sound "templates" which should be used when sampling your own sounds. They provide simple envelopes with velocity control and remove all modulation and effects to minimize confusion.

BOOTING MASOS

To use MASOS, the MASOS diskette must be the first diskette inserted into the Mirage after turning it on. The Mirage will load MASOS, then load the first sound on the MASOS diskette. After booting, you may load any sound into the Mirage from another diskette.

COMPATABILITY

MASOS provides all of the functions of the standard Mirage Operating System except the Sequencer. You may load and save sounds, change parameters, sample sounds and play the keyboard. Any sounds created or saved while using MASOS are completely compatible with standard Mirage sounds. MASOS can load or save sounds from any Mirage diskette.

When creating sounds with MASOS and saving to other non-MASOS disks, never save the configuration parameters **[14].** *The configuration parameters of the MASOS disk have a different format and saving the MASOS configuration on a standard disk (or vice versa) will produce unpredictable results.*

FEATURES

MASOS provides three key features. First, it contains new commands for manipulating sound sample data. Second, the expanded MIDI interface supports the optional Apple IIe*-based **Visual Editing System**. Third, higher input sample rates are possible when used with the optional **Input Sampling Filter** Cartridge.

The following section lists the new command parameters which are used in MASOS. Refer to **Chapter II - Mirage Parameters** for more information about how to use these new functions.

New Front Panel Key Functions

Since the Sequencer is not available when using MASOS, the keys in the **SEQ** section of the front panel are assigned new functions for convenience.

Upper Wavesample Select Key (**REC SEQ**)
Lower Wavesample Select Key (**PLAY SEQ**)

Pressing either of these keys will select the corresponding Mirage memory bank (upper or lower) and display the current wavesample number as a flashing digit. If no change is required, press the **CANCEL** key. If you wish to select a new wavesample, press the number key (**1** through **8**) which corresponds to wavesample number you wish to select.

MASOS Function Key (**LOAD SEQ**)

When this key is pressed, the flashing "Fn" or Function select prompt appears. Pressing the number key associated with the desired function will result in a flashing prompt for that function. If you wish to execute the function, press the **ENTER** key. If you wish to abort the function, press the **CANCEL** key.

System Reset Command (**LOAD ALL 0**)

Performing a "Load All 0" (**LOAD UPPER**, **LOAD LOWER**, **0**, **ENTER**) will reboot the system from whatever diskette is in the drive, without requiring you to turn off the power. Pressing the **CANCEL** key prior to **ENTER** will abort the reset. The contents of the sound memory will be lost and replaced with Sound 1 from the disk that is in the drive when the reset is initiated. This command is used to switch to the standard Mirage Operating System after using MASOS without turning off the power.

Caution: this command will erase the current contents of the memory! Be sure to save any sounds you want to keep before initiating this command.

The System Reset Command (LOAD ALL 0) is also available with the Standard Mirage Operating System on diskettes with version numbers higher than 1.7.

New Command Parameters

Caution: these commands will alter the current contents of the memory! Be sure to save any sounds you want to keep before initiating any of these commands.

[17] Copy Current Wavesample to Lower Wavsample (n)
[18] Copy Current Wavesample to Upper Wavsample (n)

After the display shows "U" or "L" and a flashing number, press the number key for the wavesample which the current wavesample is to be copied into. Press the **ENTER** key to execute the copy. The parameter number will appear after copying is complete.

[19] Rotate Current Wavesample Data Left by (n) Locations
[20] Rotate Current Wavesample Data Right by (n) Locations

The display will flash "rL" or "rr" for a moment to indicate the direction of rotation, then the number of locations of the previous rotate will appear. This value can be changed using the up and down arrow keys. When the correct value is reached, press **ENTER** to initiate the rotate function. The display will go blank briefly and then show "dr" indicating that the data has been rotated.

[93] External Input Filter Frequency

This parameter has been added for use with the optional **Input Sampling Filter**. The value of this parameter determines the cutoff frequency of the 7th-order input filter according to the values listed in Table 4 of Appendix A. When set to a value of 0, the Mirage will sample through it's internal filter and A/D converter. When set to any other value, the Mirage will sample through the cartridge filter and A/D.

Data Manipulation

MASOS provides a set of eight functions which may be performed on any sound data in the Mirage. To control what data is modified by these operations, some new parameters have been created which act as pointers. A few of these parameters are normally used to control the Sequencer, which has been removed from MASOS. The values of these parameters are used by the routines which perform the data manipulations and should always be set or checked before initiating any of the functions accessed by the MASOS **Function** key. These operations act directly on the data memory areas of the Mirage and are independent of Program and Wavesample control parameter settings.

The following parameters are pointers which control the range of memory that will be manipulated. Each pointer is made up of two numbers, which are both shown in Hexadecimal. The first half of a pointer is the Page number and the second half is the Sample number on that page.

[85] Source Start: page number (normally 00)
[86] Source Start: sample number (normally 00)

[87] Source End: page number (normally 01)
[88] Source End: sample number (normally FF)

[89] Destination Start: page number (normally 00)
[90] Destination Start:sample number (normally 00)

These pointers must be set before initiating any of the MASOS functions. The values of the pointers are set on bootup to the defaults listed above and are not changed by any of the functions.

[94] Destination Bank Select (normally LO)

This parameter selects the upper or lower sound memory bank as the place to store the results of those MASOS function which require a destination.

[95] Scale Function Start Factor (normally 00)
[96] Scale Function End Factor (normally FF)

These scale factors are used to control the scaling of the sample data by the Scaling Function. The range of both variables is from 00 to FF (Hex).

The loop should be turned off in any memory area being operated upon to avoid inserting zeroes in the waveform data. Additionally, when the loop is off, the last sixteen bytes (samples) of any wavesample are not waveform data, but are, instead, zeroes used to halt the oscillators. These zeroes are always present and should not be included when operating on a wavesample. Reversing a wavesample, including these zeroes, will put zeroes at the beginning of the waveform data, causing erratic results on playback. For this reason, always set the Source End (sample) **[88]** *to EF, not FF, if the Source End (page)* **[87]** *is the last page of the wavesample*

MASOS Data Manipulation Functions

These special functions are accessed by pressing the MASOS **Function** key , then pressing the number which corresponds to the function to be performed. A flashing mnemonic message will appear until you either press the **ENTER** key to initiate the function, or the **CANCEL** key to abort it. When the requested function has been completed, the "Fc" or Function Complete message will appear.

Function #	Mnemonic	Description
1	Cd	Copy Data from Source to Destination.
2	Fi	Fade in from Source Start to Source End
3	Fo	Fade out from Source Start to Source End
4	Sc	Scale the Source data with a linear ramp function which ramps between Scale Start Factor and Scale End Factor.
5	Ad	Add Source data to Destination data; leave result in Destination.
6	In	Invert the Source data.
7	rd	Reverse data from Source Start to End
8	rP	Replicate the first page of the Source data on every page of the Source data.

Further information and graphical depictions of the MASOS functions can be found starting on page 26 (**MASOS DATA MANIPULATION FUNCTIONS**) and page 66 (**Time Domain Manipulations**).

APPENDIX C - MIRAGE MASOS MIDI IMPLEMENTATION 1.1

This is the Mirage MIDI implementation available after booting from the MASOS disk. The Real Time messages are eliminated and additional commands are added to support the MASOS functions when used with the **Visual Editing System**.

1.0 TRANSMIT DATA.

1.1 Channel Information.

status	second	third	Description
1000NNNN	0KKKKKKK	0VVVVVVV	Note off: K[36..96], V[1..127].
1001NNNN	0KKKKKKK	0VVVVVVV	Note on: K[36..96], V[1..127].
1011NNNN	01000000	01111111	Sustain pedal on (depressed). *1,*2
1011NNNN	01000000	00000000	Sustain pedal off (released). *1 ,*2
1011NNNN	00000001	0VVVVVVV	Mod wheel amount: V[0..127]. *2
1110NNNN	00000000	0VVVVVVV	Pitch bender: V[0..127]; V=64:center, V=0:max flat, V=127:max sharp. *2

1.2 Notes

*1 *Only sent if pedal is in Sustain Pedal Mode, parameter* **[89]**.
*2 *Only sent if enabled by Midi Controller Enable, parameter* **[84]**.

On power up Mirage defaults to Omni on (mode 1)
and will transmit on Channel number 1 (NNNN=0).

Mirage is always in poly mode.
Mirage can be set to Omni off Poly mode (mode 3).

2.0 RECOGNIZED RECEIVE DATA.

2.1 Channel Information.

status	second	third	Description
1000NNNN	0KKKKKKK	0VVVVVVV	Note off: K[36..96], V[1..127]; V=0 will be treated as V=64.

1001NNNN	0KKKKKKK	0VVVVVVV	Note on: (V≠0) K[36..96], V[1..127]; V=0 treated as note off with V=64.
1011NNNN	01000000	01111111	Sustain pedal on (depressed).
1011NNNN	01000000	00000000	Sustain pedal off (released).
1011NNNN	00000001	0VVVVVVV	Mod wheel amount: V[0..127]. *1
1110NNNN	0XXXXXXX	0VVVVVVV	Pitch bender: v[0..127]; V=64:center, V=0:max flat, V=127:max sharp. *1 X is ignored.
1011NNNN	01111011	00000000	All notes off command. *2
1011NNNN	01111100	00000000	Omni mode off. All notes off. *2
1011NNNN	01111101	00000000	Omni mode on. All notes off. *2
1011NNNN	01111110	00000000	Mono mode on. All notes off. *2 Mono mode off is ignored but all notes are turned off.
1011NNNN	01111111	00000000	Poly mode on. All notes off. *2

1 Only recognized when enabled by Midi Controller Enable, parameter* **[84].
**2 These function as a key release for all notes. The notes will not stop sounding instantly unless the envelope release is zero.*

The Mirage powers up in Omni on, Poly mode. In this case, channel data is recognized on all channels.
When the Mirage is in Omni off mode, channel data is only recognized for the channel selected by the MIDI Channel Select, parameter **[82]**.

Controllers only affect notes played by MIDI key commands. A MIDI Pitch Bend or Mod Wheel command will only affect notes which sounded as a result of MIDI key commands. Notes played on the local keyboard or from the sequencer will not be affected.

3.0 SYSTEM EXCLUSIVE FORMAT.

3.1 **Receive Data.**

3.1.1 Mirage Command Code (front panel commands).

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000001	Mirage command code.
0NNNNNNN	Up to 5 bytes of front panel message
.	(Table 3.3). Must be a complete
.	message, i.e. terminated with
.	ENTER if the command requires it.
01111111	End of Command marker.

3.1.2 Program Dump Request.

This command instructs the Mirage to dump its current Programs for the upper or lower keyboard.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
000N0011	Program dump request;
	N=0: lower, N=1:upper.

3.1.3 Configuration Parameters Dump Request.

This command instructs the Mirage to dump its current configuration parameters.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000000	Configuration parameters dump request.

3.1.4 Wavesample Dump Request.

Used to ask the Mirage to dump the current wavesample as selected by Wavesample Select, parameter **[26]**.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000100	Wavesample dump request.

3.1.5 Wavesample Dump Absolute Request.

Used to ask the Mirage to send waveform data from the absolute addresses specified in the message.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage Code.
00001011	Wavesample dump absolute request.
000NLLLL	N=0: lower, N=1: upper.
0000LLLL	LLLLLLLL = Low byte of start address.
0000HHHH	
0000HHHH	HHHHHHHH = High byte of start address.
0000LLLL	
0000LLLL	LLLLLLLL = Low byte of end address.
0000HHHH	
0000HHHH	HHHHHHHH = High byte of end address.

3.1.6 Wavesample Dump Absolute Data.

Used to send or receive data to or from the Mirage from the pointers passed in the message.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage Code.
00001100	Wavesample dump absolute data message.
000NLLLL	N=0: lower, N=1: upper.
0000LLLL	LLLLLLLL = Low byte of start address.
0000HHHH	
0000HHHH	HHHHHHHH = High byte of start address.

0000LLLL	
0000LLLL	LLLLLLLL = Low byte of end address.
0000HHHH	
0000HHHH	HHHHHHHH = High byte of end address.
.	Each data byte sent as 2 nybbles with
.	the MS nybble clear. Low nybble is sent
.	first, i.e. 0000LLLL then 0000HHHH.
.	
0CCCCCCC	Checksum: formed as a modulo 128 add of all data nybbles plus the nybbles of the start and end addresses.

3.1.7 Program Dump Data.

Used to send upper or lower programs to the Mirage.

11110000	System Exclusive.
00001111	ENSONIQ code.
00000001	Mirage code.
000N0101	Program dump data message; N= 0: lower,N=1: upper.
.	
.	Program data (Table 3.4).
.	
.	

3.1.8 Wavesample Dump Data.

Used to transfer data into the currently selected wavesample on the Mirage.
Note: A Wavesample should not be transmitted and received simultaneously, as this will have unpredictable results.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000110	Wavesample dump data message.
0000CCCC	Page count LS nybble.
0000CCCC	Page count MS nybble.
.	

.	Each data byte sent as 2 nybbles with
.	the MS nybble clear. Low nybble is sent
.	first, i.e. 0000LLLL then 0000HHHH.
0CCCCCCC	Checksum: formed as a modulo 128 add of each nybble and the page count.

3.1.9 Perform Wavesample Manipulation Function Command.

Used to perform MASOS wavesample manipulation functions. When the function is completed an acknowledge is transmitted by the Mirage.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage Code.
00001111	Manipulation function command code.
000N0FFF	N=0: lower,N=1: upper. FFF = Command function code. *1
0000LLLL	Source Start address
0000LLLL	LS nybble through MS nybble.
0000HHHH	Address = HHHHHHHH LLLLLLLL.
0000HHHH	
0000LLLL	Source End address
0000LLLL	LS nybble through MS nybble.
0000HHHH	Address = HHHHHHHH LLLLLLLL.
0000HHHH	
0000FFFF	Scale Start Factor (FFFFFFFF).
0000FFFF	Always transmit whether used or not.
0000FFFF	Scale End Factor (FFFFFFFF).
0000FFFF	Always transmit whether used or not.
0000LLLL	Destination Start address.
0000LLLL	Always transmit whether used or not.
0000HHHH	Address = HHHHHHHH LLLLLLLL.
000NHHHH	N=1: dest. upper, N=0: dest. lower.

3.1.10 Notes

*1 Function Code definitions:

0 = Copy Source to Destination.
1 = Scale Source according to Scale Factors.
2 = Add Source to Destination, place result in Destination.
3 = Invert Source.
4 = Reverse Source.
5 = Replicate Source page.
6 = Rotate Current Wavesample Left.
7 = Rotate Current Wavesample Right.

3.2 Transmit Data.

3.2.1 Program Parameter Message.

This message is sent whenever a parameter is changed from the front panel on the Mirage. *1

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001101	Program parameter message.
000N00PP	N=0: lower, N=1: upper, PP=Program #.
0XXXXXXX	Parameter #.
0000VVVV	Value LS nybble.
0000VVVV	Value MS nybble.

3.2.2 Wavesample Parameter Message

This message is sent whenever a wavesample parameter is changed from the front panel of the Mirage. *1

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001110	Wavesample parameter message.
	N=0: lower, N=1: upper.
000N0SSS	SSS = wavesample #.
0XXXXXXX	Parameter #.
0000VVVV	Value LS nybble.
0000VVVV	Value MS nybble.

3.2.3 Program Status Message.

Sent by Mirage when a Program number is changed from the front panel. *1

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000111	Program change status message.
000N00PP	N=0: lower, N=1: upper, PP=Program #.

3.2.4 Wavesample Status Message.

Sent by Mirage when a new wavesample is selected from the front panel. *1

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001000	Wavesample status message.
000N0SSS	N=0: lower, N=1: upper, SSS = wavesample #.

3.2.5 Wavesample Dump Received without Error (Ack).

Sent by the Mirage when the checksum of a received wavesample dump is good. Also sent when a wavesample function is completed.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001001	Wavesample Ack message.

3.2.6 Wavesample Dump Received with Error (Nak).

Sent by the Mirage when the checksum of a received wavesample dump is bad.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001010	Wavesample Nak message.

3.2.7 Program Dump Data.

Used to send the upper or lower program from the Mirage.

11110000	System Exclusive.
00001111	ENSONIQ code.
00000001	Mirage code.
000N0101	Program dump message. N=0: lower,N=1: upper.
.	
.	Program data (Table 3.4) sent as two nybbles with MS nybble clear, low nybble first, i.e., 0000LLLL then 0000HHHH.
.	
.	

3.2.8 Configuration Dump Data.

Used to send the configuration parameters from the Mirage.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000010	Configuration dump data message.
.	
.	Configuration dump data (Table 3.5) sent as two nybbles with MS nybble clear, low nybble first, i.e., 0000LLLL then 0000HHHH.
.	
.	

3.2.9 Wave Table Dump Data.

Used to send the current wavesample from the Mirage.
Note: A Wavesample should not be transmitted and received simultaneously as this will have unpredictable results.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000110	Wave table dump message.
0000CCCC	Page count LS nybble.
0000CCCC	Page count MS nybble.
.	
.	Wavesample data

.	sent as two nybbles with MS
.	nybble clear, low nybble first,
	i.e., `0000LLLL` then `0000HHHH`.
`0CCCCCCC`	Checksum of wavesample dump. Formed as a modulo 128 add of the data nybbles and the page count.

3.2.10 Notes

1 Only transmitted when enabled by External Computer Port Enable parameter* **[91].

3.3 Keypad Number Decoding Table.

Hex Code	Dec Code	Key Label	Hex Code	Dec Code	Key label
00	00	0/PROG	14	20	REC SEQ
01	01	1	15	21	PLAY SEQ
02	02	2	16	22	LOAD SEQ
03	03	3	17	23	SAVE SEQ
04	04	4	10	16	LOAD UPPER
05	05	5	11	17	LOAD LOWER
06	06	6	12	18	SAMPLE UPPER
07	07	7	13	19	SAMPLE LOWER
08	08	8	0C	12	PARAM
09	09	9	0D	13	VALUE
0A	10	ENTER/START	0E	14	(UP ARROW)
0B	11	CANCEL/STOP	0F	15	(DOWN ARROW)

3.4 Program Dump Table Format.

The Program dump contains eigth Wavesample Control Blocks, eight Segment Lists and four Program Parameter Blocks. The data is transferred in nybbles and reassembled to form a block of bytes.

Byte Offset	Length	Description
0	1	Sound Rev Level
1	24	Wavesample Control Block 1
25	24	Wavesample Control Block 2
49	24	Wavesample Control Block 3
73	24	Wavesample Control Block 4

97	24	Wavesample Control Block 5
121	24	Wavesample Control Block 6
145	24	Wavesample Control Block 7
169	24	Wavesample Control Block 8
193	32	Segment List, Wavesample 1
225	32	Segment List, Wavesample 2
257	32	Segment List, Wavesample 3
289	32	Segment List, Wavesample 4
321	32	Segment List, Wavesample 5
353	32	Segment List, Wavesample 6
385	32	Segment List, Wavesample 7
417	32	Segment List, Wavesample 8
449	32	Segment List, spare
481	36	Parameter Block, Program 1
517	36	Parameter Block, Program 2
553	36	Parameter Block, Program 3
589	36	Parameter Block, Program 4

Total Length: 625 bytes

Wavesample Control Block Length: 24 bytes

Each Wavesample Control Block contains pointers which are used internally by the Mirage operating system.

WARNING: no wavesample control parameters should be changed externally. The Mirage perfroms internal housekeeping tasks when any of these values are changed. Use front panel command equivalents to change them. Consider these control blocks to be Read Only.

Byte Offset	Parameter #	Parameter Name	Range
0		Sample Start Pointer	
2		Sample End Pointer	
4		Loop Start Pointer	
6		Loop End Pointer	
8	**[65]**	Loop Switch	0..1
9	**[67]**	Coarse Tune	0..7
10	**[68]**	Fine Tune	0..255
11	**[69]**	Relative Amplitude	0..63
12	**[70]**	Relative Filter Freq.	0..198
13	**[71]**	Maximum Filter Freq.	0..198
14	**[72]**	Top Key	0..60
15	**[60]**	Wavesample Start	0..FE
16	**[61]**	Wavesample End	1..FF

17	**[62]**	Loop Start	0..FE
18	**[63]**	Loop End	1..FF
19	**[64]**	Loop End Fine	0..FF
20		Free Run Flag	0..1
21		spare 3 bytes	0

Segment List Length: 32 bytes

The Segment List is a set of operating system variables which control the digital oscillators.

As with the Wavesample Control Blocks, the Segment Lists should not be changed externally and sent back to the Mirage.

Program Parameter Block Length: 36 bytes

Byte Offset	Parameter #	Parameter Name	Range
0	**[28]**	Mono Mode Switch	0..1
1	**[31]**	LFO Frequency	0..99
2	**[32]**	LFO Depth	0..99
3	**[33]**	Osc Detune	0..99
4	**[34]**	Osc Mix	0..252
5	**[35]**	Mix Vel Sens.	0..124
6	**[36]**	Filter Cutoff Freq.	0..198
7	**[37]**	Resonance	0..160
8	**[38]**	Filter Kybd Tracking	0..4
9		spare	0
10	**[27]**	Initial Wavesample	0..7
11	**[28]**	Mix Mode Switch	0..1
12	**[40]**	A Filter Envelope	0..31
13	**[41]**	P	0..31
14	**[42]**	D	0..31
15	**[43]**	S	0..31
16	**[44]**	R	0..31
17	**[45]**	Av	0..124
18	**[46]**	Pv	0..124
19	**[47]**	Dk	0..124
20	**[48]**	Sv	0..124
21	**[49]**	Rv	0..124
22	**[50]**	A Amplitude Envelope	0..31
23	**[51]**	P	0..31
24	**[52]**	D	0..31
25	**[53]**	S	0..31
26	**[54]**	R	0..31

27	[55]	Av	0..124
28	[56]	Pv	0..124
29	[57]	Dk	0..124
30	[58]	Sv	0..124
31	[59]	Rv	0..124
32		spare 4 bytes	0

3.5 Configuration Dump Table.

Byte Offset	Parameter #	Parameter Name	Range
0		Dummy byte must be zero	
1	[21]	Master Tune	0..99
2	[22]	Pitch Bender Range	0..34
3	[23]	Velocity Sensitivity	0..63
4	[24]	Upper/lower Balance	0..126 *1
5	[25]	Program Link Switch	0..1
6	[73]	Sample Time Adjust	20..99
7	[74]	Input Filter Freq.	0..198 *1
8	[75]	Mic/Line Switch	0..1
9	[76]	Sampling Threshold	0..126 *1
10	[77]	Multisample Switch	0..1
11	[81]	Midi Omni-mode Flag	0..1
12	[82]	Midi Channel	0..15
13	[83]	Midi Thru-mode switch	0..1
14	[84]	Midi Mod Enable Switch	0..1
15	[85]	Source Start MSB	0..255
16	[86]	Source Start LSB	0..255
17	[87]	Source End MSB	0..255
18	[88]	Source End LSB	0..255
19	[89]	Destination MSB	0..255
20	[90]	Destination LSB	0..255
21	[94]	Destination Bank	0..1
22	[95]	Scale Start Factor	0..255
23	[96]	Scale End Factor	0..255
24	[91]	External Comp. Switch	0..1
25	[92]	Baud Rate Switch	0..1
26	[93]	Cartridge Filter Freq.	0..25
27	[97]	Software Version	0..255
28		Spare	0..255

3.5.1 Notes

**1 All parameters with this mark are displayed divided by 2. They must always have their least significant bit reset.*

APPENDIX D - MIRAGE MIDI IMPLEMENTATION 1.1

This is the normal Mirage MIDI implementation which is available when the Mirage is booted from a Sound Disk or Blank Formatted Diskette.

1.0 TRANSMIT DATA.

1.1 Channel Information.

status	second	third	Description
1000NNNN	0KKKKKKK	0VVVVVVV	Note off: K[36..96], V[1..127].
1001NNNN	0KKKKKKK	0VVVVVVV	Note on: K[36..96], V[1..127].
1011NNNN	01000000	01111111	Sustain pedal on (depressed). *1,*2
1011NNNN	01000000	00000000	Sustain pedal off (released). *1 ,*2
1011NNNN	00000001	0VVVVVVV	Mod wheel amount: V[0..127]. *2
1110NNNN	00000000	0VVVVVVV	Pitch bender: V[0..127]; V=64:center, V=0:max flat, V=127:max sharp. *2
1011NNNN	01111011	00000000	All Notes Off, transmitted at end of sequence.

1.2 Real Time Data

11111000	Timing clock sent when internal seq. clock is selected or when MIDI clock is selected and a MIDI clock is received. Also sent when external clock is selected and an external clock occurs.

1.3 Notes

*1 *Only sent if pedal is in Sustain Pedal Mode, parameter* **[89]**.
*2 *Only sent if enabled by Midi Controller Enable, parameter* **[84]**.

On power up Mirage defaults to Omni on (mode 1)
and will transmit on Channel number 1 (NNNN=0).

Mirage is always in poly mode.
Mirage can be set to Omni off Poly mode (mode 3).

2.0 RECOGNIZED RECEIVE DATA.

2.1 Channel Information.

status	second	third	Description
1000NNNN	0KKKKKKK	0VVVVVVV	Note off: K[36..96], V[1..127]; V=0 will be treated as V=64.
1001NNNN	0KKKKKKK	0VVVVVVV	Note on: (V≠0) K[36..96], V[1..127]; V=0 treated as note off with V=64.
1011NNNN	01000000	01111111	Sustain pedal on (depressed).
1011NNNN	01000000	00000000	Sustain pedal off (released).
1011NNNN	00000001	0VVVVVVV	Mod wheel amount: V[0..127]. *1
1110NNNN	0XXXXXXX	0VVVVVVV	Pitch bender: v[0..127]; V=64:center, V=0:max flat, V=127:max sharp. *1 X is ignored.
1011NNNN	01111011	00000000	All notes off command. *2
1011NNNN	01111100	00000000	Omni mode off. All notes off. *2
1011NNNN	01111101	00000000	Omni mode on. All notes off. *2
1011NNNN	01111110	00000000	Mono mode on. All notes off. *2 Mono mode off is ignored but all notes are turned off.
1011NNNN	01111111	00000000	Poly mode on. All notes off. *2

2.2 Real Time Data.

11111000	Timing clock recognized when MIDI clock is selected.
11111010	Start sequencer.
11111011	Continue sequencer.
11111100	Stop sequencer.

2.3 Notes

1 Only recognized when enabled by Midi Controller Enable, parameter* **[84].
**2 These function as a key release for all notes. The notes will not stop sounding instantly unless the envelope release is zero.*

The Mirage powers up in Omni on, Poly mode. In this case, channel data is recognized on all channels.
When the Mirage is in Omni off mode, channel data is only recognized for the channel selected by the MIDI Channel Select, parameter **[82]**.

Controllers only affect notes played by MIDI key commands. A MIDI Pitch Bend or Mod Wheel command will only affect notes which sounded as a result of MIDI key commands. Notes played on the local keyboard or from the sequencer will not be affected.

3.0 SYSTEM EXCLUSIVE FORMAT.

3.1 Receive Data.

3.1.1 Mirage Command Code (front panel commands).

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000001	Mirage command code.
0NNNNNNN	Up to 5 bytes of front panel message (Table 3.3). Must be a complete message, i.e. terminated with ENTER if the command requires it.
.	
.	
.	
01111111	End of Command marker.

3.1.2 Program Dump Request.

This command instructs the Mirage to dump its current Programs for the upper or lower keyboard.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
000N0011	Program dump request; N=0: lower,N=1:upper.

3.1.3 Wavesample Dump Request.

Used to ask the Mirage to dump the current wavesample as selected by Wavesample Select, parameter **[26]**.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00000100	Wavesample dump request.

3.1.4 Program Dump Data.

Used to send upper or lower programs to the Mirage.

11110000	System Exclusive.
00001111	ENSONIQ code.
00000001	Mirage code.
000N0101	Program dump data message; N= 0: lower,N=1: upper.
.	
.	Program data (Table 3.4).
.	
.	

3.1.5 Wavesample Dump Data.

Used to transfer data into the currently selected wavesample on the Mirage.
Note: A Wavesample should not be transmitted and received simultaneously, as this will have unpredictable results.

11110000	System Exclusive.
00001111	ENSONIQ Code.

00000001	Mirage code.
00000110	Wavesample dump data message.
0000CCCC	Page count LS nybble.
0000CCCC	Page count MS nybble.
.	
.	Each data byte sent as 2 nybbles with
.	the MS nybble clear. Low nybble is sent
.	first, i.e. 0000LLLL then 0000HHHH.
0CCCCCCC	Checksum: formed as a modulo 128
	add of each nybble and the page count.

3.2 **Transmit Data.**

3.2.1 Wavesample Dump Received without Error (Ack).

Sent by the Mirage when the checksum of a received wavesample dump is good. Also sent when a wavesample function is completed.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001001	Wavesample Ack message.

3.2.2 Wavesample Dump Received with Error (Nak).

Sent by the Mirage when the checksum of a received wavesample dump is bad.

11110000	System Exclusive.
00001111	ENSONIQ Code.
00000001	Mirage code.
00001010	Wavesample Nak message.

3.3 Keypad Number Decoding Table.

Hex Code	Dec Code	Key Label	Hex Code	Dec Code	Key label
00	00	0/PROG	14	20	REC SEQ
01	01	1	15	21	PLAY SEQ
02	02	2	16	22	LOAD SEQ
03	03	3	17	23	SAVE SEQ
04	04	4	10	16	LOAD UPPER
05	05	5	11	17	LOAD LOWER
06	06	6	12	18	SAMPLE UPPER
07	07	7	13	19	SAMPLE LOWER
08	08	8	0C	12	PARAM
09	09	9	0D	13	VALUE
0A	10	ENTER/START	0E	14	(UP ARROW)
0B	11	CANCEL/STOP	0F	15	(DOWN ARROW)

3.4 Program Dump Table Format.

The Program dump contains eigth Wavesample Control Blocks, eight Segment Lists and four Program Parameter Blocks. The data is transferred in nybbles and reassembled to form a block of bytes.

Byte Ofiset	Length	Description
0	1	Sound Rev Level
1	24	Wavesample Control Block 1
25	24	Wavesample Control Block 2
49	24	Wavesample Control Block 3
73	24	Wavesample Control Block 4
97	24	Wavesample Control Block 5
121	24	Wavesample Control Block 6
145	24	Wavesample Control Block 7
169	24	Wavesample Control Block 8
193	32	Segment List, Wavesample 1
225	32	Segment List, Wavesample 2
257	32	Segment List, Wavesample 3
289	32	Segment List, Wavesample 4
321	32	Segment List, Wavesample 5
353	32	Segment List, Wavesample 6
385	32	Segment List, Wavesample 7
417	32	Segment List, Wavesample 8
449	32	Segment List, spare
481	36	Parameter Block, Program 1
517	36	Parameter Block, Program 2

553		36	Parameter Block, Program 3
589		36	Parameter Block, Program 4

Total Length: 625 bytes

Wavesample Control Block Length: 24 bytes

Each Wavesample Control Block contains pointers which are used internally by the Mirage operating system.

WARNING: no wavesample control parameters should be changed externally. The Mirage perfroms internal housekeeping tasks when any of these values are changed. Use front panel command equivalents to change them. Consider these control blocks to be Read Only.

Byte Offset	Parameter #	Parameter Name	Range
0		Sample Start Pointer	
2		Sample End Pointer	
4		Loop Start Pointer	
6		Loop End Pointer	
8	**[65]**	Loop Switch	0..1
9	**[67]**	Coarse Tune	0..7
10	**[68]**	Fine Tune	0..255
11	**[69]**	Relative Amplitude	0..63
12	**[70]**	Relative Filter Freq.	0..198
13	**[71]**	Maximum Filter Freq.	0..198
14	**[72]**	Top Key	0..60
15	**[60]**	Wavesample Start	0..FE
16	**[61]**	Wavesample End	1..FF
17	**[62]**	Loop Start	0..FE
18	**[63]**	Loop End	1..FF
19	**[64]**	Loop End Fine	0..FF
20		Free Run Flag	0..1
21		spare 3 bytes	0

Segment List Length: 32 bytes

The Segment List is a set of operating system variables which control the digital oscillators.

As with the Wavesample Control Blocks, the Segment Lists should not be changed externally and sent back to the Mirage.

Program Parameter Block Length: 36 bytes

Byte Offset	Parameter #	Parameter Name	Range
0	**[28]**	Mono Mode Switch	0..1
1	**[31]**	LFO Frequency	0..99
2	**[32]**	LFO Depth	0..99
3	**[33]**	Osc Detune	0..99
4	**[34]**	Osc Mix	0..252
5	**[35]**	Mix Vel Sens.	0..124
6	**[36]**	Filter Cutoff Freq.	0..198
7	**[37]**	Resonance	0..160
8	**[38]**	Filter Kybd Tracking	0..4
9		spare	0
10	**[27]**	Initial Wavesample	0..7
11	**[28]**	Mix Mode Switch	0..1
12	**[40]**	A Filter Envelope	0..31
13	**[41]**	P	0..31
14	**[42]**	D	0..31
15	**[43]**	S	0..31
16	**[44]**	R	0..31
17	**[45]**	Av	0..124
18	**[46]**	Pv	0..124
19	**[47]**	Dk	0..124
20	**[48]**	Sv	0..124
21	**[49]**	Rv	0..124
22	**[50]**	A Amplitude Envelope	0..31
23	**[51]**	P	0..31
24	**[52]**	D	0..31
25	**[53]**	S	0..31
26	**[54]**	R	0..31
27	**[55]**	Av	0..124
28	**[56]**	Pv	0..124
29	**[57]**	Dk	0..124
30	**[58]**	Sv	0..124
31	**[59]**	Rv	0..124
32		spare 4 bytes	0

APPENDIX E - MASOS PARAMETER SETTINGS

The following three charts give a complete list of all relevant parameter settings for the three sound templates on the MASOS disk. The first sound provides two equal-sized Wavesamples per keyboard half. The second sound provides four Wavesamples per half and the third provides eight wavesamples per half.

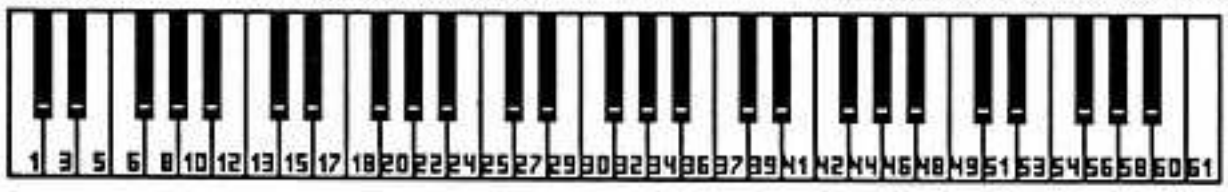
These charts can be used as a reference for creating your own sounds using the fourth blank template chart provided. You can make copies of this fourth chart and fill in your own numbers and notes for any sounds you sample. It is always a good idea to note what sample rate and input filter setting you used in sampling a sound, for future reference.

The Programs provided on these templates are all the same, giving "plain vanilla" envelopes and filtering for minimum effect on the sound. A small amount of velocity sensitivity is provided on the envelopes to get you started and all other special effects (chorusing, etc.) are disabled.

Note that the blank template form has no provisions for filling in those parameters that are part of the system configuration. It is assumed that settings for pitch-bend, MIDI, Sequencer, etc. have all been set by you and saved using the Save Configuration command.

ensoniq

Mirage
Programmer
Reference



Creator:

Sound Name:
MASOS Lower 1
Date: **4/3/85**

PROGRAM

1	2	3	4	
01	01	01	01	**[27]** Init Wavesample
off	off	off	off	**[28]** Mix Mode
off	off	off	off	**[29]** Mono Mode
14	14	14	14	**[31]** LFO Freq
00	00	00	00	**[32]** LFO Depth
00	00	00	00	**[33]** Osc 2 Detune
00	00	00	00	**[34]** Osc Mix
01	01	01	01	**[35]** Osc Mix Vel Sens
45	45	45	45	**[36]** Filter Cutoff Freq
00	00	00	00	**[37]** Filter Resonance
04	04	04	04	**[38]** Filter Kybd Track

Filter Envelope

00	00	00	00	**[40]** Attack
00	00	00	00	**[41]** Peak
15	15	15	15	**[42]** Decay
00	00	00	00	**[43]** Sustain
31	31	31	31	**[44]** Release
00	00	00	00	**[45]** Attack Vs
00	00	00	00	**[46]** Peak Vs
00	00	00	00	**[47]** Decay Kyb
00	00	00	00	**[48]** Sustain Vs
00	00	00	00	**[49]** Release Vs

Amplitude Envelope

00	00	00	00	**[50]** Attack
10	10	10	10	**[51]** Peak
15	15	15	15	**[52]** Decay
10	10	10	10	**[53]** Sustain
10	10	10	10	**[54]** Release
00	00	00	00	**[55]** Attack Vs
21	21	21	21	**[56]** Peak Vs
00	00	00	00	**[57]** Decay Kyb
21	21	21	21	**[58]** Sustain Vs
00	00	00	00	**[59]** Release Vs

WAVESAMPLE

1	2	3	4	5	6	7	8	
00	80							**[60]** Wavesample Start
7F	FF							**[61]** Wavesample End
7E	FE							**[62]** Loop Start
7E	FE							**[63]** Loop End
FF	FF							**[64]** Loop End Fine Adj.
off	off							**[65]** Loop Switch
04	04							**[67]** Rel. Tuning Coarse
80	80							**[68]** Rel. Tuning Fine
63	63							**[69]** Rel. Amplitude
30	30							**[70]** Rel. Filter Freq.
99	99							**[71]** Max. Filter Freq.
15	31							**[72]** Top Key
34	34							**[73]** Sample Time Adj.
80	80							**[74]** Input Filter Freq
off	off							**[75]** Line Input
10	10							**[76]** Sampling Threshold
								Note (Pitch) Sampled

Notes:

This Template provides two Wavesamples per keyboard half, with minimal filtering and velocity response.

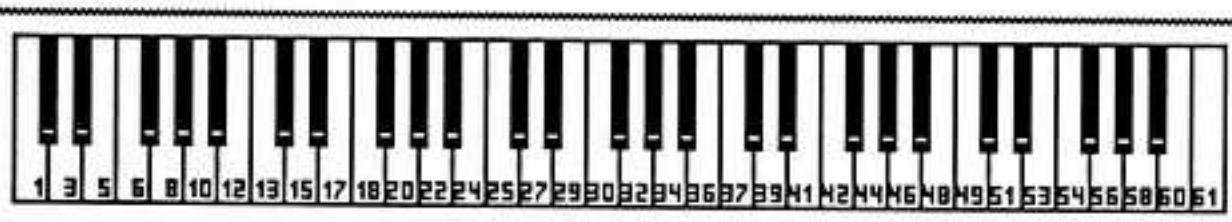
All Programs are identical.

Settings for unused Wavesamples are ignored.

The Upper settings are identical to these with these exceptions: Top Key of Wavesample 1 =46
Top Key of Wavesample 2 =61
Relative Filter Freq. =10

ensoniq

Mirage
Programmer
Reference



Creator:

Sound Name:
MASOS Lower 2
Date: **4/3/85**

PROGRAM

1	2	3	4	
01	01	01	01	**[27]** Init Wavesample
off	off	off	off	**[28]** Mix Mode
off	off	off	off	**[29]** Mono Mode
14	14	14	14	**[31]** LFO Freq
00	00	00	00	**[32]** LFO Depth
00	00	00	00	**[33]** Osc 2 Detune
00	00	00	00	**[34]** Osc Mix
01	01	01	01	**[35]** Osc Mix Vel Sens
45	45	45	45	**[36]** Filter Cutoff Freq
00	00	00	00	**[37]** Filter Resonance
04	04	04	04	**[38]** Filter Kybd Track

Filter Envelope

00	00	00	00	**[40]** Attack
00	00	00	00	**[41]** Peak
15	15	15	15	**[42]** Decay
00	00	00	00	**[43]** Sustain
31	31	31	31	**[44]** Release
00	00	00	00	**[45]** Attack Vs
00	00	00	00	**[46]** Peak Vs
00	00	00	00	**[47]** Decay Kyb
00	00	00	00	**[48]** Sustain Vs
00	00	00	00	**[49]** Release Vs

Amplitude Envelope

00	00	00	00	**[50]** Attack
10	10	10	10	**[51]** Peak
15	15	15	15	**[52]** Decay
10	10	10	10	**[53]** Sustain
10	10	10	10	**[54]** Release
00	00	00	00	**[55]** Attack Vs
21	21	21	21	**[56]** Peak Vs
00	00	00	00	**[57]** Decay Kyb
21	21	21	21	**[58]** Sustain Vs
00	00	00	00	**[59]** Release Vs

WAVESAMPLE

1	2	3	4	5	6	7	8	
00	40	80	C0					**[60]** Wavesample Start
3F	7F	BF	FF					**[61]** Wavesample End
3E	7E	BE	FE					**[62]** Loop Start
3E	7E	BE	FE					**[63]** Loop End
FF	FF	FF	FF					**[64]** Loop End Fine Adj.
off	off	off	off					**[65]** Loop Switch
04	04	04	04					**[67]** Rel. Tuning Coarse
80	80	80	80					**[68]** Rel. Tuning Fine
63	63	63	63					**[69]** Rel. Amplitude
30	30	30	30					**[70]** Rel. Filter Freq.
99	99	99	99					**[71]** Max. Filter Freq.
08	15	23	31					**[72]** Top Key
34	34	34	34					**[73]** Sample Time Adj.
80	80	80	80					**[74]** Input Filter Freq
off	off	off	off					**[75]** Line Input
10	10	10	10					**[76]** Sampling Threshold
								Note (Pitch) Sampled

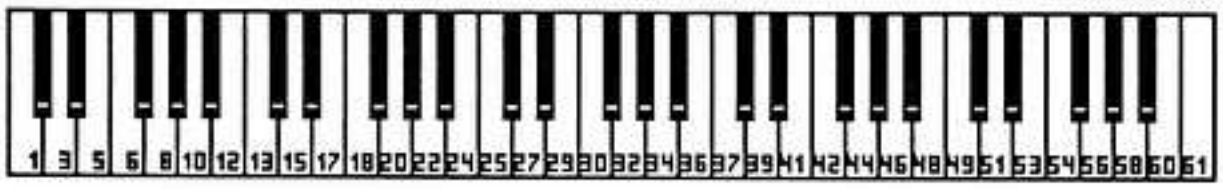
Notes:

This Template provides four Wavesamples per keyboard half. The notes for MASOS Lower 1 apply.

The Upper settings are identical to the Lower with these exceptions: Top Key of Wavesample 1 = 39
Top Key of Wavesample 2 = 46
Top Key of Wavesample 3 = 54
Top Key of Wavesample 4 = 61
Relative Filter Freq. = 10

ensoniq

Mirage
Programmer
Reference



Creator:

Sound Name:
MASOS Lower3
Date: **4/3/85**

PROGRAM

1	2	3	4	
01	01	01	01	**[27]** Init Wavesample
off	off	off	off	**[28]** Mix Mode
off	off	off	off	**[29]** Mono Mode
14	14	14	14	**[31]** LFO Freq
00	00	00	00	**[32]** LFO Depth
00	00	00	00	**[33]** Osc 2 Detune
00	00	00	00	**[34]** Osc Mix
01	01	01	01	**[35]** Osc Mix Vel Sens
45	45	45	45	**[36]** Filter Cutoff Freq
00	00	00	00	**[37]** Filter Resonance
04	04	04	04	**[38]** Filter Kybd Track

Filter Envelope

00	00	00	00	**[40]** Attack
00	00	00	00	**[41]** Peak
15	15	15	15	**[42]** Decay
00	00	00	00	**[43]** Sustain
31	31	31	31	**[44]** Release
00	00	00	00	**[45]** Attack Vs
00	00	00	00	**[46]** Peak Vs
00	00	00	00	**[47]** Decay Kyb
00	00	00	00	**[48]** Sustain Vs
00	00	00	00	**[49]** Release Vs

Amplitude Envelope

00	00	00	00	**[50]** Attack
10	10	10	10	**[51]** Peak
15	15	15	15	**[52]** Decay
10	10	10	10	**[53]** Sustain
10	10	10	10	**[54]** Release
00	00	00	00	**[55]** Attack Vs
21	21	21	21	**[56]** Peak Vs
00	00	00	00	**[57]** Decay Kyb
21	21	21	21	**[58]** Sustain Vs
00	00	00	00	**[59]** Release Vs

WAVESAMPLE

1	2	3	4	5	6	7	8	
00	20	40	60	80	A0	C0	E0	**[60]** Wavesample Start
1F	3F	5F	7F	9F	BF	DF	FF	**[61]** Wavesample End
1E	3E	5E	7E	9E	BE	DE	FE	**[62]** Loop Start
1E	3E	5E	7E	9E	BE	DE	FE	**[63]** Loop End
FF	FF	FF	FF	FF	FF	FF	FF	**[64]** Loop End Fine Adj.
off	off	off	off	off	off	off	off	**[65]** Loop Switch
04	04	04	04	04	04	04	04	**[67]** Rel. Tuning Coarse
80	80	80	80	80	80	80	80	**[68]** Rel. Tuning Fine
63	63	63	63	63	63	63	63	**[69]** Rel. Amplitude
30	30	30	30	30	30	30	30	**[70]** Rel. Filter Freq.
99	99	99	99	99	99	99	99	**[71]** Max. Filter Freq.
04	08	12	15	19	23	27	31	**[72]** Top Key
34	34	34	34	34	34	34	34	**[73]** Sample Time Adj.
80	80	80	80	80	80	80	80	**[74]** Input Filter Freq
off	off	off	off	off	off	off	off	**[75]** Line Input
10	10	10	10	10	10	10	10	**[76]** Sampling Threshold
								Note (Pitch) Sampled

Notes:

This Template provides eight Wavesamples per keyboard half. The notes for MASOS Lower 1 apply.

The Upper settings are identical to the Lower with these exceptions: Top Key of Wavesample 1 = 35
Top Key of Wavesample 2 = 39
Top Key of Wavesample 3 = 43
Top Key of Wavesample 4 = 46
Top Key of Wavesample 1 = 50
Top Key of Wavesample 2 = 54
Top Key of Wavesample 3 = 58
Top Key of Wavesample 4 = 61
Relative Filter Freq. = 10

ensoniq

Mirage
Programmer
Reference

Creator:

Sound Name:

Date:

PROGRAM
1 2 3 4

[27] Init Wavesample
[28] Mix Mode
[29] Mono Mode

[31] LFO Freq
[32] LFO Depth
[33] Osc 2 Detune
[34] Osc Mix
[35] Osc Mix Vel Sens
[36] Filter Cutoff Freq
[37] Filter Resonance
[38] Filter Kybd Track

Filter Envelope

[40] Attack
[41] Peak
[42] Decay
[43] Sustain
[44] Release
[45] Attack Vs
[46] Peak Vs
[47] Decay Kyb
[48] Sustain Vs
[49] Release Vs

Amplitude Envelope

[50] Attack
[51] Peak
[52] Decay
[53] Sustain
[54] Release
[55] Attack Vs
[56] Peak Vs
[57] Decay Kyb
[58] Sustain Vs
[59] Release Vs

WAVESAMPLE
1 2 3 4 5 6 7 8

[60] Wavesample Start
[61] Wavesample End

[62] Loop Start
[63] Loop End
[64] Loop End Fine Adj.
[65] Loop Switch

[67] Rel. Tuning Coarse
[68] Rel. Tuning Fine
[69] Rel. Amplitude
[70] Rel. Filter Freq.
[71] Max. Filter Freq.
[72] Top Key

[73] Sample Time Adj.
[74] Input Filter Freq
[75] Line/Mic Input
[76] Sampling Threshold

Note (Pitch) Sampled

Notes:

INDEX