INTRODUCTION

In the Beginning…

In the summer of 2000, I found myself sitting in a recording studio in *Newcastle* surrounded by the very latest in state-of-the-art technology. We were recording our second full-length album and this was the final step in a 5-year journey. However, all was not well. After 5 years, I was bored with the music, bored with the sounds and desperate to move on to any number of new projects.

The barrier, it seemed, was the compositional process itself. Frankly, I'm not a particularly gifted musician. I don't have the chops to improvise new music and what I produce is generally found through accident rather than by design. Equally, I found myself getting locked into one particular style, one particular groove and I began to yearn for a faster, more elegant process where the musician wasn't restricted to one formula and one style. I wanted to get back to just making music as a process rather than with an end product in mind, a process where you could just make music as quickly as you could imagine it.

To cut a long story short, I began to imagine a machine that could take care of the areas that I found difficult. My keyboard technique couldn't really be classed as a technique at all and so I wanted something that could play quickly and accurately but also make it simple to improvise new ideas around an existing theme. I wanted

a machine where I could test ideas quickly and without the limitations of the normal computer interface.

In short, I started to imagine *ZEIT*.

I wrote down all of the functions and properties that such a machine would have and then I went looking to see if it already existing. After all, I didn't want to reinvent the wheel. The sequencers that I found didn't really address my needs completely. They filled in part of the picture but none of them combined ease of use with functionality. This was the true starting point for the *ZEIT* project.

With the help of some good friends, *Paul Maddox* and *Paul Nagle*, we began to specify what *ZEIT* would do and how it would do it. *Paul Maddox* designed the microprocessor board that would control the machine, *Paul Nagle* added his own thoughts and ideas to my basic specification.

The first tangible result of this collaboration was a machine we christened *ATEM*. It was a four-track, menu driven machine meant as a proof of concept. It worked but the user interface was a disaster with far too many menus and key combinations to make the instrument an effective musical tool. Clearly, we needed to press forward with the next logical step, the *ZEIT Step Sequencer* itself.

In *ZEIT* we wanted to create a tool that was, first and foremost, a solid and reliable workhorse. It would perform one task and it would perform it well. And *ZEIT* also had to be completely intuitive to use. Equally at home in the studio or on stage, all of the major controls had to be available at all times with the status of each parameter always clearly visible. No weird and wonderful button combinations would be required to get from A to B. One button, one function became a kind of mantra whilst we were designing the instrument. Wherever possible, menu options had to be displayed in plain English instead of a multiplicity of three letter acronyms.

We also decided that ***ZEIT*** would evolve naturally over time. Software updates and bug fixes would arrive at regular, routine intervals. Additions to the specification would only be made if sufficient users asked for a set of features and the user base agreed that there was a real need for a new function, rather than something that was just crudely bolted on as an afterthought.

In short, we wanted to create an instrument, which like its analogue forebears would reward the musician for many, many years to come, a source of endless inspiration in a world dominated by here today, gone tomorrow instruments.

ZEIT took four years to develop. We set out to create a machine that would work with the musician as a creative tool, a machine that would reduce the distance between concept and execution as much as possible.

I believe we succeeded.

David Hughes, February 2006

Why a Hardware Sequencer?

Why choose a hardware sequencer when there are myriads of sleek, elegant, well-specified software sequencers available on the market, often for little or no cost at all?

Well, over the years, we've found that software sequencers, good though they are, rarely measure up to their hardware-based counterparts. Most of the time, you can work around their shortcomings but there are occasions when only a piece of dedicated hardware will give you what you need to get the job done.

We reckon that the main advantages of hardware based sequencers are as follows - timing integrity, ease-of-use and reliability.

1) Timing integrity

It almost goes without saying that any sequencer, be it software, hardware or a combination of the two, must have absolute timing accuracy. I've met classically trained musicians who can hear MIDI note delays of the order of a millisecond or so. They trust their ears and their ears tell them that MIDI delays are "unmusical" and "unnatural".

With a software sequencer running on a host computer, the source of your timing signals is usually a high-priority, interrupt-driven software routine and this routine periodically tells the sequencer package to wake up and do something. Consequently, the performance of your sequencer package is usually outside the control of the programmers who wrote your sequencer package. More often than not its performance is dictated by the demands placed upon host computer's operating system.

Just stop for a moment and think about everything that your average PC has to do just to stay running - poll the keyboard, check for network activity, look for e-mail, figure out what the mouse is doing, update the display card etc. If that wasn't enough, modern PC's (and Macs) boast specifications that were pure science fiction just a decade ago. The modern PC is a highly sophisticated, extremely

versatile piece of equipment, which needs a large and complex operating system to run it smoothly.

And all of this power and sophistication means that users are forever adding extra bits of software to their system. Screen savers, network printers, Web Cams, games. You name it, your PC probably has it. Sure, the machine can cope, can't it? It's not like checking e-mail takes up a huge amount of time, right?

Wrong! The more software you have installed on a machine then the more likely it is that your precious MIDI timing signals will be pushed into a queue behind some other system critical activity. And waiting in a queue for service is not the way to run a music sequencer.

Most of the time, you might not notice a problem. Most of the time, you won't even be aware that the sequencer skipped a clock or that a clock was late. But, occasionally, the timing signals won't happen at exactly the right moment and the sequencer will drop out of time with everything else. Then you'll notice. And if you were trying to create an accurately timed, precisely driven piece of music then the magic will have been lost. Completely.

Suppose now that you have a collection of drum machines and sequencers all synced to your master PC. Usually, the main source of timing synchronisation is MIDI clock signals, which are fundamentally a poor way of syncing two devices but we're stuck with them.

MIDI clock signals - you gotta love them. Sometimes, they arrive in bursts, all bundled up together. Sometimes they spread out. And your systems have to respond to these changes in a manner that is deemed 'musical'. It's asking a lot. Most of the time, this isn't down to the design of the software sequencer itself. It's nearly always due to the operating system within the computer.

This is where accuracy counts most and this is where hardware sequencers really score over software devices. Every time.

With ZEIT, timing signals always have the highest priority. When the system timer expires it is serviced in less than 1 microsecond. (That's one millionth of a second!) And timing signals are never pushed into a queue - they're always processed immediately. The serial nature of MIDI means that notes are never going to arrive precisely on time. However, ZEIT always tries to stream the note information as smoothly as possible by using fast, efficient output buffers that are serviced very quickly indeed - as soon as the previous character has cleared the output transmitter. There’s no sitting around in a queue, waiting for a slice of the processor’s valuable time.

2) Ease of use

Every software package I've ever used has suffered from a feature christened "bloatware", that is, the tendency of software writers to add feature after feature to a program without there being much tangible benefit to the end user. Remember that 99% of users only utilise around 50% of a package's functionality - they know enough to get by and that's it.

Think of it this way. In my word processor package, there's an on-screen icon marked **BOLD**. How often do I use the on-screen icon marked **BOLD**? I don't. I use a keystroke command, <*CTRL-B*>, to make the text bold. It's quicker and faster than hunting for the icon marked **BOLD**. Equally, there's also a drop-down menu that lets me set a line of text to **BOLD** too. How often do I use that feature? Sometimes, during the final stages of editing. I'll wager you're much the same. I'd really just prefer a button marked **BOLD**. And another button marked *Italique*, another marked "Underline" etc.

That's the thing with software sequencers. They do too much. You have to think too much if you want to use them. More features mean larger programs, which means more code and, consequently, more bugs.

A sequencer is a tool. You use it to make music. It is reasonable to expect a PC or a Macintosh to do just the same. Most professional musicians I've met have insisted that the PC is kept as simple as possible. They recognize that simple = reliable. And yet most of their dedicated studio machines have games installed, e-mail running, screen savers, MSN messaging etc. And most are surprised, even angry when their machines crash for no apparent reason.

In designing ZEIT, our mantra was "if you need to look in the manual then the design is wrong!". The system is as simple and as complex as it needs to be. Once you're familiar with it then you should never need to look in the manual again. You just switch it on and play.

With ZEIT, the user interface is fixed and won't change. Software upgrades may be released that will change small details but our personal assurance to you is that the front panel won't suddenly change out of all recognition. We make the functionality completely obvious and, as far as possible, we stick to our golden rule of 'one function per key, one key per function'. A simple, elegant design with no weird and wacky functions that only power users ever touch. No multiple keystrokes or "Press this then this then that" key sequences. Switch it on and press "play". That's all you need.

3) Reliability

A few years ago, I went to a demonstration of a software sequencer at a local high street store. I won't mention the name of the company or the package for obvious reasons. It was not their finest hour.

Two of this company's leading demonstrators hopped up on stage and proudly declared, in front of a very large and highly attentive audience, that the latest revision of their wonderful sequencer package was their best yet. It not only surpassed all previous versions but every other sequencer on the market. It did everything. It was the ultimate sequencer. You would need no other sequencer tool.

They pressed "Start" and ... it crashed.

Attempts to restart the computer were met with curious blue screens. When the computer did restart, it wouldn't load all of the software. Critically, the sequencer package we'd all come to see wouldn't run.

And so everyone in the room sat around whilst the puzzled technical bods at the front of the room called up equally puzzled technical bods at the service centre and discussed possible fixes. Eventually, they did get the computer and the software working and it certainly looked fantastic - very bright, very colourful, feature rich and just absolutely wonderful in every way. Except that the demonstrators couldn't remember how to use it and made all sorts of embarrassed excuses about version creep and software updates and so forth. They tried this option and that option and when users started asking questions from the floor, it became clear that they just weren't familiar with the software package at all. This was because the all-new, all-singing, all-dancing version had just been released and it had very little in common with the previous release.

Now, just when you though life couldn't get any worse for these poor guys, the computer crashed again, mid-demo.

Quite frankly, I'd had enough. I went home to my ancient and well worn Macintosh 266MHz G3, which was last updated in 1998, switched it on and got on with finishing off a couple of tunes I'd just written. I still have version 3.01 of their software sequencer. It's reliable, it works and it does everything I need ... except step sequence, which is why I designed ZEIT.

Reliability is central to the philosophy behind ZEIT. Our main selling point is that the sequencer is there when you need it. Switch it on and it works. It should never crash, for any reason. Software fixes are never rushed out. They're always tested properly amongst a range of qualified and experienced beta-testers. We won't ever dump untested, untried software on our users.

Turn a knob or push a switch and the machine will respond instantly. ZEIT doesn't play games and it doesn't use MSN Instant Messenger. But it does play sequences and we think it plays sequences rather well.