THE FAIRLIGHT EXPLAINED

When the Fairlight CMI was released seven years ago, both the machine and its software represented a significant step forward in the application of computer technology to music. Today, the Fairlight is used in the making of popular music the world over, as well as performing an important role in the field of musical and technological education. Despite this, very few people are fully aware of what the CMI does and how it does it. Jim Grant, **who's been working with one for a number of years at the London College of Furniture, has decided to rectify matters by writing a series dedicated to explaining the Fairlight's workings. Part one appears below.**

Before the Fairlight's appearance, most computer music systems were the perogative of mainframes and their contribution to the world of everyday music was slight. Kim Ryrie (the Fairlight's father) and his Australian colleagues soon changed all that, however and their invention is now used in almost area of music production, to the extent that many people appreciate its sound without realising that they're listening to 'music by numbers'.

However, despite its widespread use, there are relatively few Fairlights in general circulation – less than 100 in the UK – and to see one in action at close quarters is a real treat. Herein lies the rationale for this series of articles. What does the Fairlight do? How does it do it? And what can the average musician do with it?

Hardware

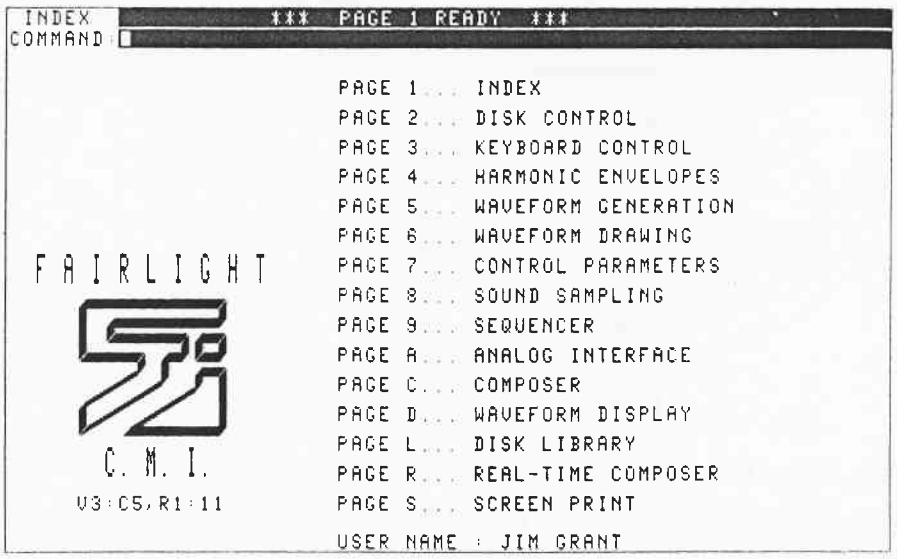
To take delivery of a Fairlight leaves your bank balance empty and your living room full. The hardware consists of a Central Processor Unit (CPU), one – or optionally two – six-octave keyboards, a typewriter-style QWERTY keyboard, and a VDU with added lightpen. In addition, there are some long connecting leads, a Systems floppy disk drive and a box of disks containing library sounds.

Software

A foolproof system of connectors – and a quick glance at the manual on the part of the user – ensures that the Fairlight can be powered up in no more than five minutes. The VDU displays the expectant message 'CMI READY', while the CPU hums quietly: there are three fans pulling air through the innards, keeping 500 watts of power dissipation down to an acceptable temperature. . . .

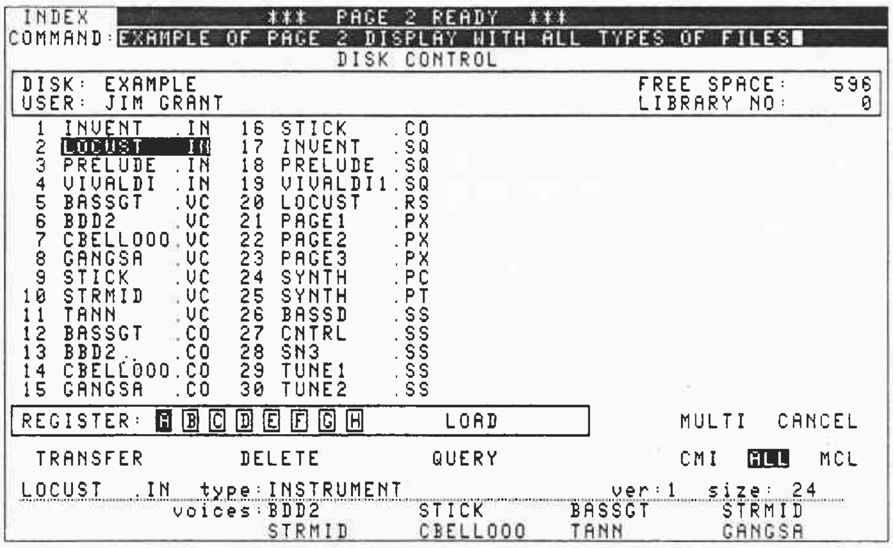
Inserting the Systems disk in the left-hand drive (Drive 0) results in a faint click as the stepper motors engage. The operating software is loaded as a series of 'fetches' – each section of program loaded pulls in the next section. When this process has been completed, the user is faced with the Index page. See Figure 1.

Here lies one of the Fairlight's most powerful features. The whole system is menu-driven and the different options correspond to different VDU displays and sets of commands which are entered from the alphanumeric keyboard. Each option is referred to as a Page. A Page has one or more files resident on the Systems disk which are loaded when the Page is selected. Page 1 is the Index Menu itself (Figure 1), while Page 2 manages the files stored on the disk in the right-hand drive (Drive 1). These files are user-created, and there are seven different types, as indicated by the suffix after the file name. These are as follows:
NAME.VC is a voice file occupying about 20kBytes. It holds waveform data (16K) and extra information regarding looping and so on.
NAME.CO holds control information such as portamento, vibrato frequency and depth.



```
INDEX     ***  PAGE 1 READY  ***
COMMAND

                         PAGE 1... INDEX
                         PAGE 2... DISK CONTROL
                         PAGE 3... KEYBOARD CONTROL
                         PAGE 4... HARMONIC ENVELOPES
                         PAGE 5... WAVEFORM GENERATION
                         PAGE 6... WAVEFORM DRAWING
 F A I R L I G H T       PAGE 7... CONTROL PARAMETERS
                         PAGE 8... SOUND SAMPLING
                         PAGE 9... SEQUENCER
                         PAGE A... ANALOG INTERFACE
                         PAGE C... COMPOSER
                         PAGE D... WAVEFORM DISPLAY
                         PAGE L... DISK LIBRARY
      C. M. I.           PAGE R... REAL-TIME COMPOSER
   V3:C5,R1:11           PAGE S... SCREEN PRINT

                         USER NAME : JIM GRANT
```

Figure 1.



```
 INDEX          ***  PAGE 2 READY  ***
COMMAND:EXAMPLE OF PAGE 2 DISPLAY WITH ALL TYPES OF FILES
                     DISK CONTROL
 DISK: EXAMPLE                                  FREE SPACE:   596
 USER: JIM GRANT                                LIBRARY NO:     0
  1 INVENT   .IN   16 STICK    .CO
  2 LOCUST   .IN   17 INVENT   .SQ
  3 PRELUDE  .IN   18 PRELUDE  .SQ
  4 VIVALDI  .IN   19 VIVALDI1.SQ
  5 BASSGT   .VC   20 LOCUST   .RS
  6 BDD2     .VC   21 PAGE1    .PX
  7 CBELLOOO.VC    22 PAGE2    .PX
  8 GANGSA   .VC   23 PAGE3    .PX
  9 STICK    .VC   24 SYNTH    .PC
 10 STRMID   .VC   25 SYNTH    .PT
 11 TANN     .VC   26 BASSD    .SS
 12 BASSGT   .CO   27 CNTRL    .SS
 13 BBD2..   .CO   28 SN3      .SS
 14 CBELLOOO.CO    29 TUNE1    .SS
 15 GANGSA   .CO   30 TUNE2    .SS
 REGISTER: A B C D E F G H        LOAD              MULTI   CANCEL

  TRANSFER        DELETE        QUERY               CMI  ALL  MCL
 LOCUST   .IN  type:INSTRUMENT                 ver:1  size: 24
             voices:BDD2         STICK     BASSGT      STRMID
                    STRMID       CBELLOOO  TANN        GANGSA
```

Figure 2.

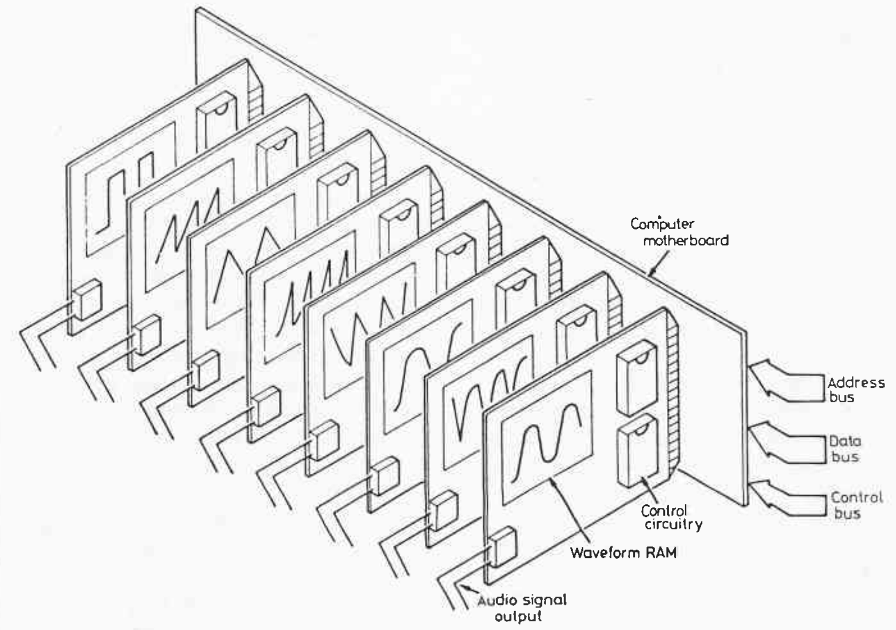
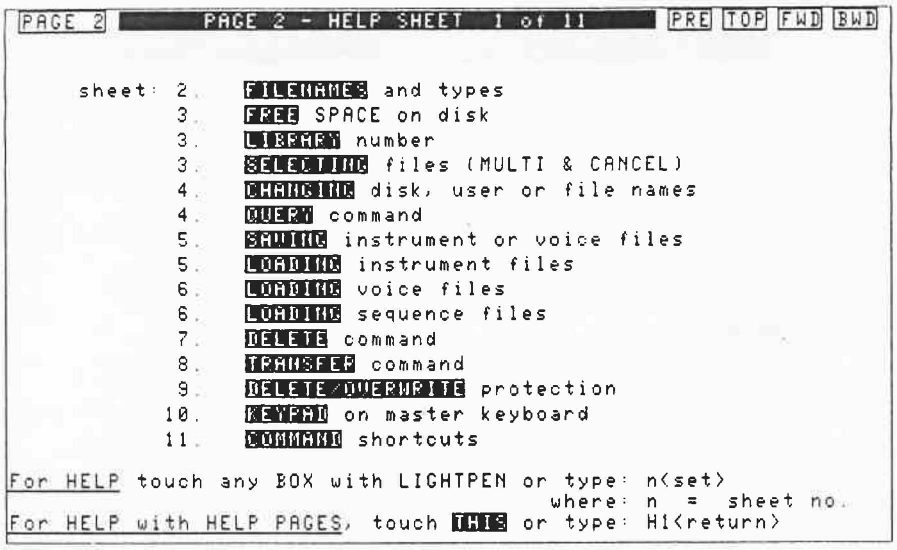
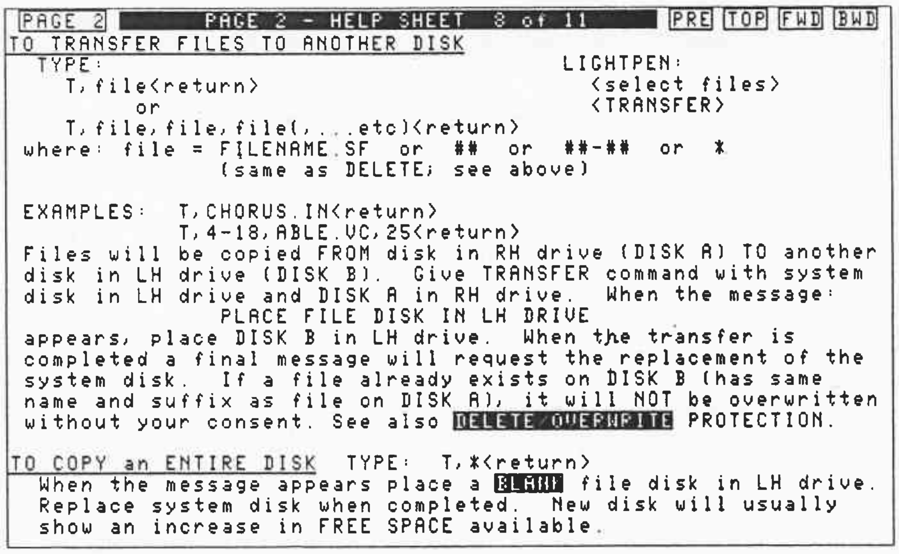


Figure 3. Fairlight channel cards with specimen waveforms illustrated.



```
PAGE 2    PAGE 2 - HELP SHEET  1 of 11    PRE TOP FWD BWD

      sheet: 2.   FILENAMES and types
             3.   FREE SPACE on disk
             3.   LIBRARY number
             3.   SELECTING files (MULTI & CANCEL)
             4.   CHANGING disk, user or file names
             4.   QUERY command
             5.   SAVING instrument or voice files
             5.   LOADING instrument files
             6.   LOADING voice files
             6.   LOADING sequence files
             7.   DELETE command
             8.   TRANSFER command
             9.   DELETE/OVERWRITE protection
            10.   KEYPAD on master keyboard
            11.   COMMAND shortcuts

For HELP touch any BOX with LIGHTPEN or type: n<set>
                                       where: n  =  sheet no.
For HELP with HELP PAGES, touch THIS or type: H1<return>
```

Figure 4.



```
PAGE 2    PAGE 2 - HELP SHEET  8 of 11    PRE TOP FWD BWD
TO TRANSFER FILES TO ANOTHER DISK
  TYPE:                                  LIGHTPEN:
    T,file<return>                         <select files>
         or                                <TRANSFER>
    T,file,file,file(,...etc)<return>
 where: file = FILENAME.SF  or  ##  or  ##-##  or  *
              (same as DELETE; see above)

 EXAMPLES:  T,CHORUS.IN<return>
            T,4-18,ABLE.VC,25<return>
 Files will be copied FROM disk in RH drive (DISK A) TO another
 disk in LH drive (DISK B).  Give TRANSFER command with system
 disk in LH drive and DISK A in RH drive.  When the message:
                 PLACE FILE DISK IN LH DRIVE
 appears, place DISK B in LH drive.  When the transfer is
 completed a final message will request the replacement of the
 system disk.  If a file already exists on DISK B (has same
 name and suffix as file on DISK A), it will NOT be overwritten
 without your consent.  See also DELETE/OVERWRITE PROTECTION.

TO COPY an ENTIRE DISK   TYPE:  T,*<return>
  When the message appears place a BLANK file disk in LH drive.
  Replace system disk when completed.  New disk will usually
  show an increase in FREE SPACE available.
```

Figure 5.

NAME.IN configures the CMI to a particular instrument state. Voices are automatically loaded and spread across the keyboard.
NAME.SQ holds polyphonic keyboard sequencer information.
NAME.RS is a real time sequencer (Page R) file.
NAME.PX corresponds to a screen dump (Page S) to disk. This can be spooled later to a dot-matrix printer for hard copy.
NAME.PC, PT, SS are Music Composition Language (MCL) files. These are generated on Page C and hold text files that describe notes with duration, dynamics and so on.

A voice file can be loaded in a number of ways. Probably the easiest is to point the lightpen at the voice name and then at the command LOAD at the bottom of the display (Figure 2). Drive 1 springs into action immediately, and after a second or two the selected voice appears on the keyboard.

So far so good. But where does the voice information go, and how does it result in a sound when the keyboard is played?

Channel Cards

Inside the Fairlight, there are usually at least 16 circuit cards, the exact number depending on various options such as an analogue interface and sync card, and eight of these are known as voice or channel cards.

The Fairlight produces sound by a process called Waveform Synthesis. Each command that deals directly with sound generation must involve at least a section of a waveform. The waveform itself is held in 16K of RAM on each channel card as a direct digital representation, so that increasing amplitudes give larger binary numbers. Therefore, when an eight-note polyphonic sound is present on the keyboard, each channel holds the same voice data.

Put simply, the channel cards can be regarded as digital oscillators whose waveform is determined by the contents of 16K of RAM (Figure 3). Different pitches – as played on the keyboard – correspond to the RAM information being read and converted by a DAC at different rates; the channel cards perform this function autonomously. The computer section of the Fairlight passes parameters such as pitch, vibrato, portamento rate and looping points along its data bus, and once these have been received, the channel card outputs the sound until the parameters are updated.

Overall pitching of the CMI is determined by a system clock resident on a special card known as the Master card. We'll be referring to this on numerous occasions over the next few months, since it holds the circuitry for a good many of the CMI's functions. A 34MHz oscillator is onboard, and this is divided and fed to the individual channel cards. It's from this clock that the RAM clocking rates – and thus keyboard pitches – are generated. The whole instrument can be tuned by scaling the master clock.

Page 2 Commands

Looking again at Figure 2, there are several commands at the bottom of the display. TRANSFER allows files to be copied from one disk to another, using Drive 0 as the destination drive. This is essential for creating backup copies of important music and/or sounds. DELETE erases unwanted files to make room on the disk. Invoking this command prompts a confirmation message to prevent accidental erasure of important files. When a file is deleted, FREE SPACE increases by the deleted file size.

At the very bottom of the display is an example of the QUERY command. This tells us that LOCUST.IN file will automatically load eight voices, whose names are shown.

Help Pages

By this time, you're probably wondering how anybody using a Fairlight ever manages to remember all the commands, especially since we've only considered Page 2 and there are another 13 still to go.

The answer is simple: Help Pages.

Figures 4 and 5 show examples of Page 2 Help Pages. In fact, the entire user's manual is held on the Systems disk, and sections relevant to the current display Page can be inspected at any time by typing HELP (what else?).

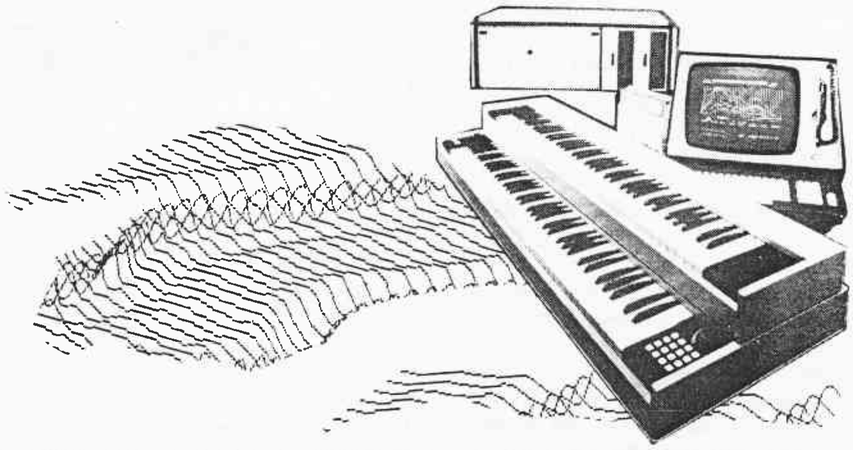
Initially, an index sheet is displayed (Figure 4). Touching any of the highlighted options with the lightpen results in the Help sheet specific to the selected option being loaded and displayed. The user can flick backwards (BWD), forwards (FWD); or recall a previous place (PRE): commands can be entered from the Help sheets while viewing their correct format. The CMI then automatically reloads the display page that called the Help sheet in the first place, and executes the command. And yes, there are even Help sheets that explain the use of the Help sheets. . . .

That about wraps up the first part of what will doubtless become a saga of some duration. Next month, we'll take a look at Page 3 – the keyboard map – and the waveform display page, Page D.

Jim Grant **E&MM**

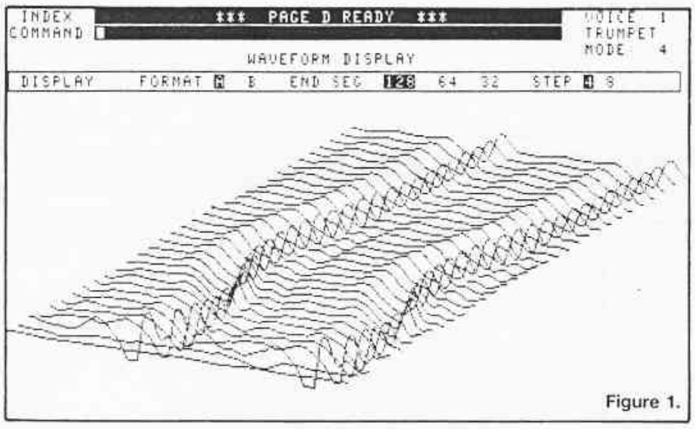
THE FAIRLIGHT EXPLAINED

The second part of our insight into one of the world's most popular computer instruments looks at display pages and what they tell CMI owners. *Jim Grant*



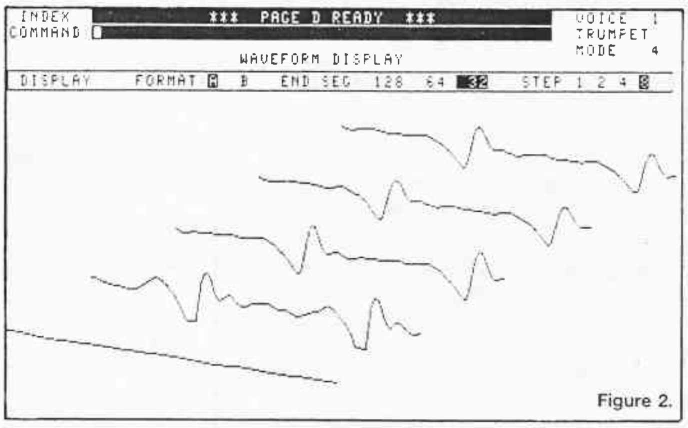
Last month we described the general software concept of the Fairlight and how much of its power lies in its ability to present its functions to the user as a group of related files called Pages. We saw that the CMI powered up with the Index Page (Page 1) and that sound and music files were managed by Page 2.

One of the problems associated with most sound-generating equipment is the way in which the sense sight is excluded from the process of sound formation. Most instruments operate on



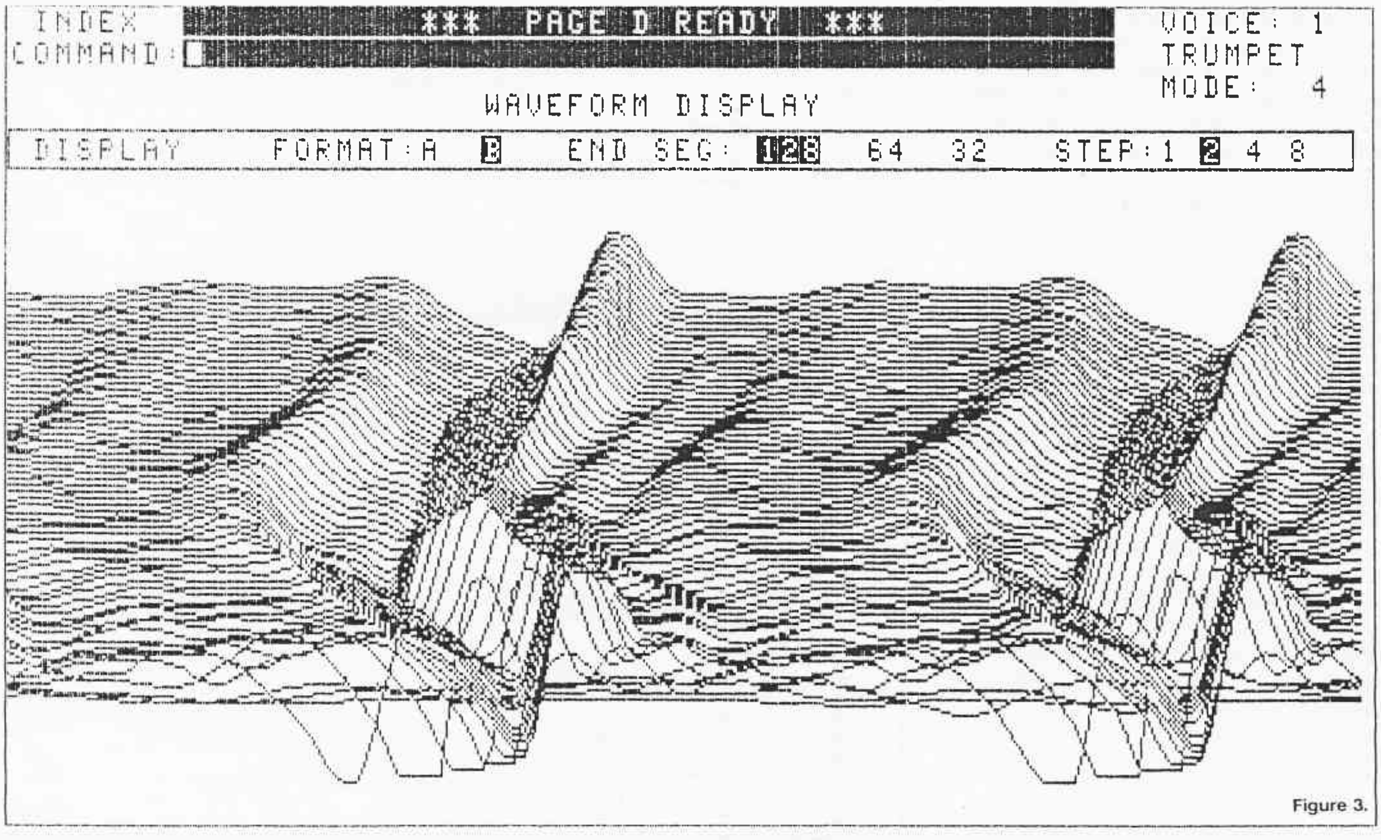
```
INDEX     *** PAGE D READY ***        VOICE   1
COMMAND                               TRUMPET
                                      MODE    4
              WAVEFORM DISPLAY
DISPLAY   FORMAT [✓ A] B   END SEG [✓ 128] 64 32   STEP [✓ 4] 8
```

Figure 1.



```
INDEX     *** PAGE D READY ***        VOICE   1
COMMAND                               TRUMPET
                                      MODE    4
              WAVEFORM DISPLAY
DISPLAY   FORMAT [✓ A] B   END SEG 128 64 [✓ 32]   STEP 1 2 4 [✓ 8]
```

Figure 2.



```
INDEX     *** PAGE D READY ***        VOICE:  1
COMMAND:                              TRUMPET
                                      MODE:   4
              WAVEFORM DISPLAY
DISPLAY   FORMAT:A [✓ B]   END SEG: [✓ 128] 64 32   STEP:1 [✓ 2] 4 8
```

Figure 3.

the basis of the user twiddling the controls and stabbing the keyboard. Fair enough: of all our senses, the ears are by far the most acute. Yet if we consider all the electronic music equipment currently available, the most user-friendly instruments have graphic displays, perhaps in the form of panel legends and LEDs or liquid crystal display. Despite the fact that sight is a very poor qualitative sense, it can be of enormous psychological help in our field. Basically it boils down to: 'If I can see it, I can understand it.'

Page D

There's no doubt that the Fairlight's visual presentation is founded on this premise. Each display Page is graphic without being ostentatious, and Page D, the voice waveform display, is a prime example of this. Typing PD followed by a RETURN on the alphanumeric keyboard will result in a display of the type shown in Figure 1. To appreciate the significance of the display, we must delve a little deeper into the workings of the CMI.

Remember that voice information is held in 16K of RAM on each channel card. To simplify matters, the CMI divides the memory (and thus the waveform) into 128 sections called segments. Each segment consists of 128 bytes, so the waveform comprises 128 segments multiplied by 128 bytes to give 16384 bytes, ie. 16K. This saves the musician handling unwieldy computer numbers when dealing with the waveform RAM.

The display shown in Figure 1 is a psuedo-3D representation of the voice called TRUMPET. Each line from left to right is a segment, and the foremost segment represents the beginning of the sound. When a keyboard note is pressed, the CMI reads out the RAM information segment by segment from the front to the rear of the display.

There are two display formats, A and B, and a number of options within each type. Figure 1 is in format A, and segments 1 to 128 are shown in steps of 4. Figure 2 is again format A, with the end segment number 32 and steps of 8: therefore only five segments are shown. Format B gives an oscilloscope-type display, but with each segment slightly above the preceding one. Again, there are a number of display options. Figure 3 shows TRUMPET segments 1 to 128 in steps of 2, and Figure 4 segments 1 to 64 in steps of 8.

Although Page D is purely for display purposes and does not support any sound creation commands, it's still an invaluable aid. At its most basic level, it answers the questions 'where has the sound gone?' and 'is the waveform zero?'.

Page 3

Page 3 is another utility-type display Page. It deals with voice tunings, Channel allocation and keyboard maps.

Figure 5 shows a typical display with eight separate voices loaded into the CMI. Registers A to H are groups of one or more of the eight Channels and 'NPHONY' is the number of notes that can be played with the sound held in a Register. A quick look at Figure 5 reveals that there are eight active Registers, each holding the voice indicated. In this case, all eight Channel cards hold a unique sound, and therefore the maximum number of notes that can be played on the keyboard with any single sound is one. This is indicated by the corresponding 'NPHONY'. If a single eight-note polyphonic voice is required, only Register A will be active and Channels 1 to 8 will be allocated to A. The active Registers are mirrored on Page 2 so that they can be loaded with sounds from disk. The Fairlight will flag an error

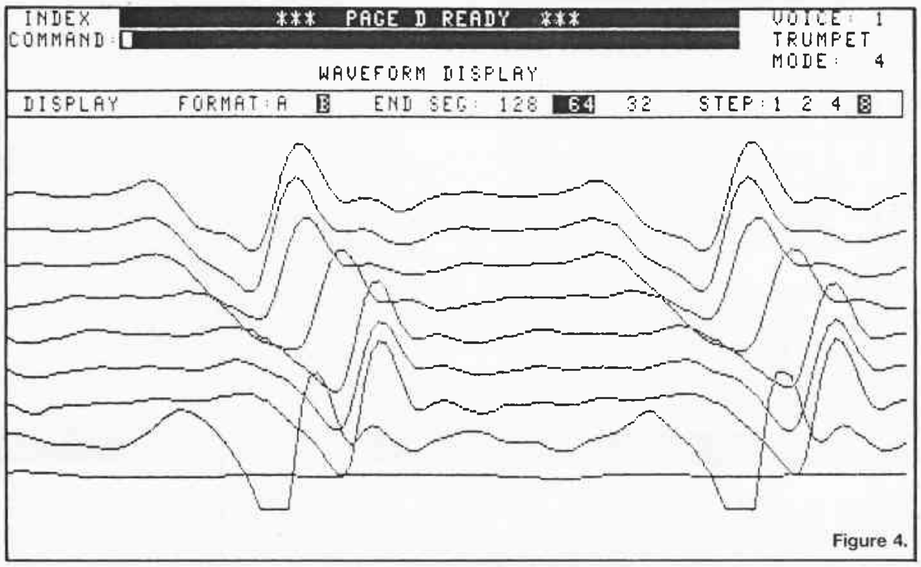
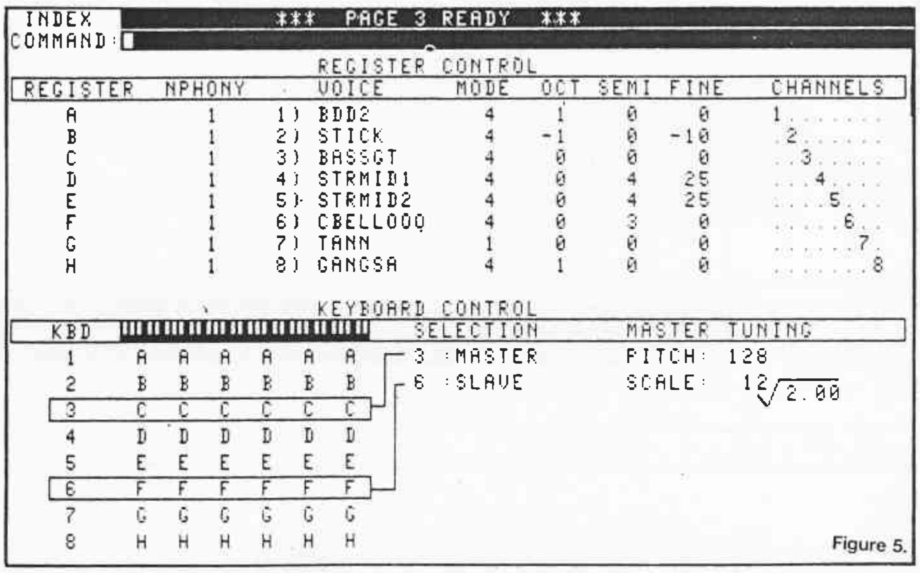
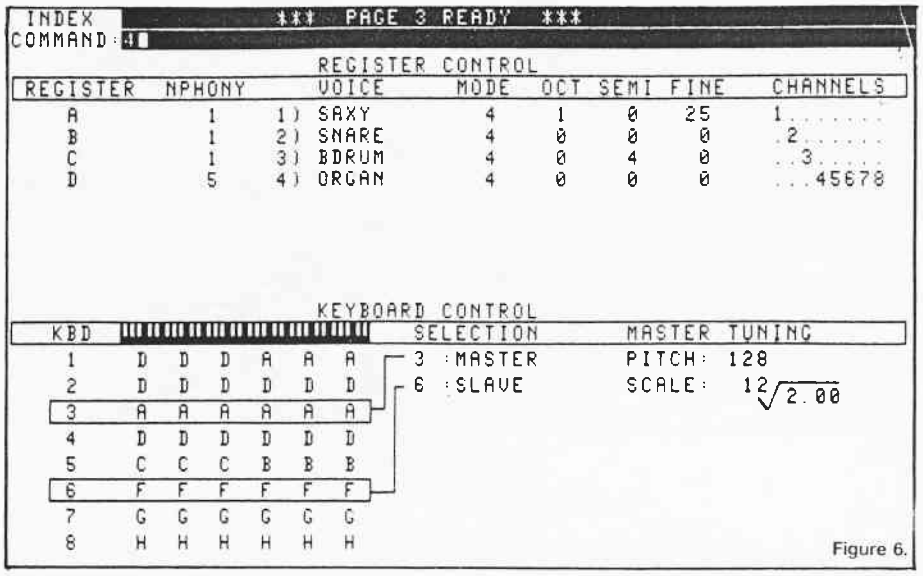


Figure 4.



```
INDEX          ***  PAGE 3 READY  ***
COMMAND:
                         REGISTER CONTROL
REGISTER   NPHONY        VOICE      MODE   OCT  SEMI  FINE    CHANNELS
    A         1      1)  BDD2        4      1     0     0     1.......
    B         1      2)  STICK       4     -1     0   -10     .2......
    C         1      3)  BASSGT      4      0     0     0     ..3.....
    D         1      4)  STRMID1     4      0     4    25     ...4....
    E         1      5)  STRMID2     4      0     4    25     ....5...
    F         1      6)  CBELLOOO    4      0     3     0     .....6..
    G         1      7)  TANN        1      0     0     0     ......7.
    H         1      8)  GANGSA      4      1     0     0     .......8

                         KEYBOARD CONTROL
  KBD                        SELECTION           MASTER TUNING
   1    A  A  A  A  A  A  ┌─ 3 :MASTER          PITCH: 128
   2    B  B  B  B  B  B  │┌ 6 :SLAVE           SCALE:  12√2.00
 [ 3    C  C  C  C  C  C ]┘│
   4    D  D  D  D  D  D   │
   5    E  E  E  E  E  E   │
 [ 6    F  F  F  F  F  F ]─┘
   7    G  G  G  G  G  G
   8    H  H  H  H  H  H
```

Figure 5.



```
INDEX          ***  PAGE 3 READY  ***
COMMAND:4
                         REGISTER CONTROL
REGISTER   NPHONY        VOICE      MODE   OCT  SEMI  FINE    CHANNELS
    A         1      1)  SAXY        4      1     0    25     1.......
    B         1      2)  SNARE       4      0     0     0     .2......
    C         1      3)  BDRUM       4      0     4     0     ..3.....
    D         5      4)  ORGAN       4      0     0     0     ...45678

                         KEYBOARD CONTROL
  KBD                        SELECTION           MASTER TUNING
   1    D  D  D  A  A  A  ┌─ 3 :MASTER          PITCH: 128
   2    D  D  D  D  D  D  │┌ 6 :SLAVE           SCALE:  12√2.00
 [ 3    A  A  A  A  A  A ]┘│
   4    D  D  D  D  D  D   │
   5    C  C  C  B  B  B   │
 [ 6    F  F  F  F  F  F ]─┘
   7    G  G  G  G  G  G
   8    H  H  H  H  H  H
```

Figure 6.

message if you try to open another Register or increase the NPHONY beyond eight. A simple rule applies: the sum of the active Registers times their NPHONY must be less than or equal to eight.

Figure 6 shows another example of the Register allocations. Here, another set of voices has been loaded, so the NPHONY and Registers are configured differently. Although the Register and NPHONY settings may seem a little confusing and limited, the exact configuration is determined entirely by the musician, and changes can be effected very quickly for evaluation.

Keyboards and Tuning

Any voice can be tuned in increments of plus or minus one hundredth of a semitone up to ±6 octaves with crystal accuracy. Scale allows the Western tempered tuning of 12th root of 2.00 to be changed to any other macro/micro tuning, eg. for quarter tones, you simply change Scale to 24th root of 2.00. Pitch is a master tuning control which can vary tuning of all the loaded voices by a quarter of a tone in 256 discrete steps, to bring the CMI in tune with other instruments if necessary.

The Keyboard Maps each consist of a keyboard number (1 to 6) followed by six letters indicating the Register assigned to each octave. As the CMI is a musician's instrument, it supports two six-octave keyboards called the Master and the Slave. Using the maps, it's possible to create eight different keyboard configurations by choosing which sounds will play on each octave within a keyboard. The Master and Slave can be linked (as shown in Figures 5 and 6) to any map by changing the Selection numbers.

The information presented in Page 3 is known as an Instrument file. The file can be saved on the user disk and is given the suffix NAME.IN. When this file is loaded it will pull the specified voices into the CMI, allocate the Registers automatically, adjust the tuning and spread the sounds across the keyboards. Instrument files are a usefully quick way of bringing the CMI up to a playable state with 'preset' voices and tuning.

Hardware

At the time the Fairlight was designed, the microprocessor was considered to be a medium- to slow-speed device. To increase the power of any computing system, designers have two basic choices.

One of these (the Synclavier approach) is to base the instrument around a discrete logic minicomputer, thus utilising the raw speed of logic chips. This is quite an elegant solution since 'music by numbers' requires lots of number crunching, but the other choice, and one which is becoming increasingly popular, is computing concurrency. In a basic CMI system, there are four microprocessors of the 6800 family. Two of these are in peripherals, ie. one each in the music keyboard and the alphanumeric keyboard, and this means there can be several independent processes being executed concurrently.

The microprocessor in the alphanumeric scans the keys and passes the data to the music keyboard when requested. At the same time, the music keyboards are scanned for pressed notes, key velocities are calculated, and the control sliders and switches read.

At the right-hand end of the Master keyboard is a calculator-style keypad and alphanumeric display used for rapid loading of voices in a live situation. Music keyboard information has the highest priority of all data in the CMI, which responds instantly to the packets of data sent flying down the cables at 9600 Baud.

Well that about finishes off our description of the utility-type display Pages. In case you're wondering what the Mode setting on Page 3 is for, don't worry: all will be explained.

Next month, the controls on Page 7 and sampling on Page 8. ■

THE FAIRLIGHT EXPLAINED

Part three, and a discussion of how the CMI samples a sound and why its own particular sampling techniques are employed. *Jim Grant*

At last we've covered enough of the CMI basics to concentrate on the more interesting sound creation Pages. Now, the single feature that characterises the Fairlight in many people's minds is its ability to sample natural sounds: this aspect is dealt with by Page 8, and is surprisingly simple to use.

At the rear of the CMI lies a selection of line and mic inputs to suit most applications. That about takes care of the hardware, because everything else is dealt with by software. Typing 'S' or touching 'Sample' with the lightpen initiates the sampling process. After a second or so, the Display box shows the sound envelope for quick monitoring of input levels (see Figure 1). If all is well with the Keyboard Maps on Page 3, the sampled sound will be playable on the music keyboard.

So far so good. But what are the other functions for? Well, some of them are self-explanatory. Sample Level is a software-based volume control and can be used to attenuate signals that exceed the input range of the Fairlight. Unwanted frequencies can be rejected by using digitally-controlled high-pass and low-pass filters: their cutoff points are set by Filter High and Filter Low. The actual circuitry lives on the ubiquitous Master Card, and takes the form of switched resistor networks using the much-loved CMOS 4051 chip.

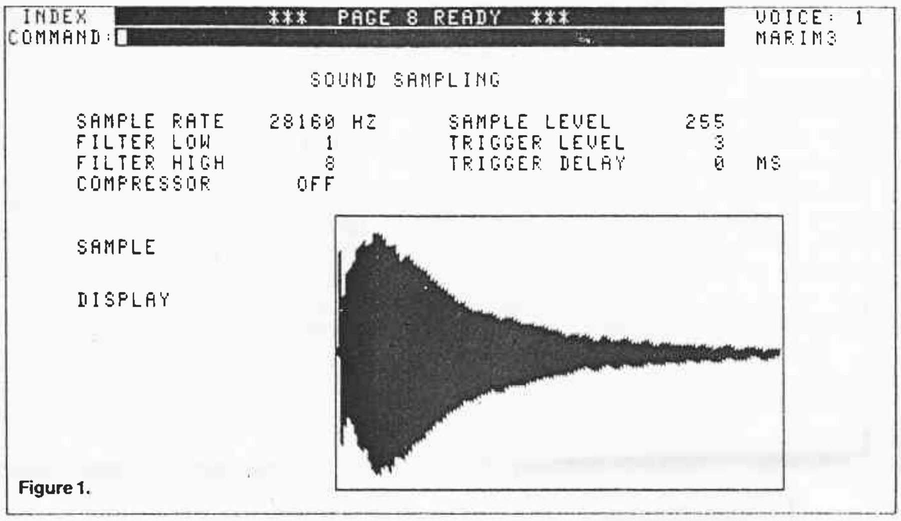
Whenever a signal is converted to a stream of digital numbers, it's necessary to bandlimit it to one half, or less, of the Sample Rate. Stated simply, this means that we must have at least two sample values of the input signal's amplitude for the highest frequency present. If this condition is not met, the information that the sampling process has captured is not sufficient to reconstruct the original signal without frequency distortion. This type of distortion is known as 'aliasing' and is both extremely noticeable and rather unpleasant. The CMI guards against 'aliasing' by incorporating tracking filters controlled by the Sample Rate.

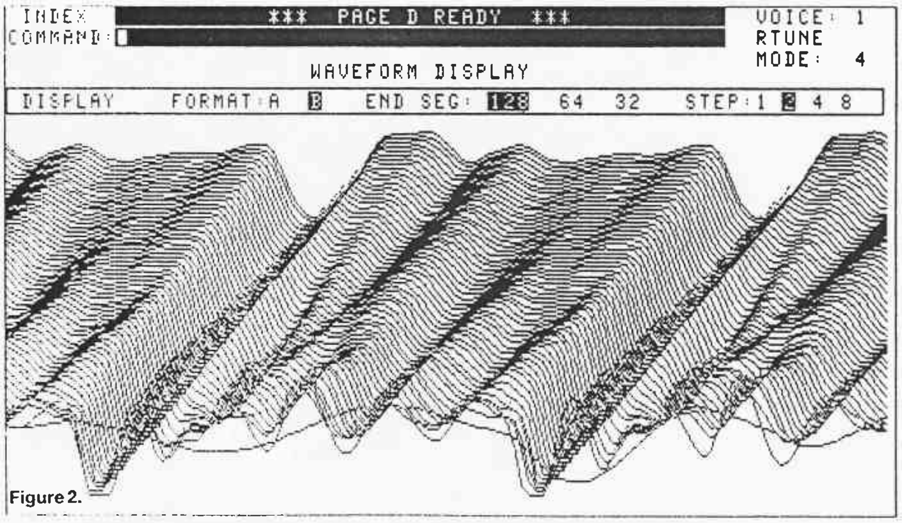
Sample Rate

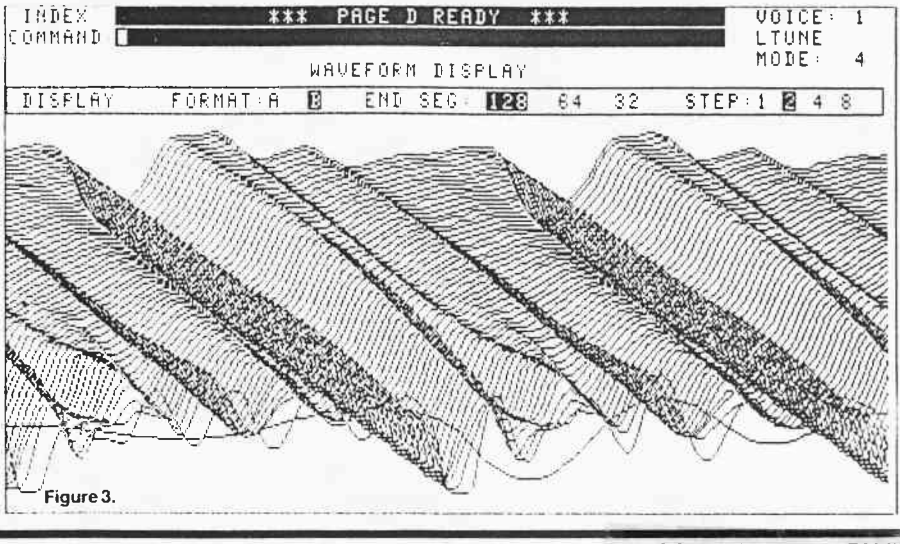
Although sampling itself is very simple, and impressive results can be obtained very quickly, it's well worth the trouble spending some time adjusting the Sample Rate to a value that suits the pitch of the input signal. The sound as played on the keyboard will only be in tune if one cycle of the resulting sampled waveform fits exactly into one segment of waveform RAM. (Remember, one segment is 128 bytes.) This is achieved when the Sample Rate equals the frequency of the input signal multiplied by 128, ie. 128 samples per cycle. Since we can adjust the tuning of the voices on Page 3, an out-of-tune sample is not in itself a problem. However, there is one more important aspect of the CMI that obliges us to pay attention to the correct sample rate.

The number of original samples taken is fixed and equals the length of the waveform RAM, ie. 16384. The faster these samples are taken, the shorter the duration of the sound becomes (although the fidelity increases). This results in a short sound when the sample is played on the keyboard, and this becomes shorter as we ascend the octaves. To overcome this, the Fairlight allows sections of the waveform RAM to be read out repeatedly (or looped) as the key is held down, thus sustaining the sound. The smallest section of RAM than can be looped is known as a Segment.

Here lies the crux of choosing a suitable Sample Rate. If we attempt to loop a Segment or group of segments that doesn't contain a whole number of cycles, the 'ends' of the loop won't join up without causing a sudden jump in amplitude. Choosing an inappropriate sample


```
INDEX              ***  PAGE 8 READY  ***              VOICE: 1
COMMAND:                                               MARIM3

                          SOUND SAMPLING

   SAMPLE RATE     28160 HZ      SAMPLE LEVEL     255
   FILTER LOW          1         TRIGGER LEVEL      3
   FILTER HIGH         8         TRIGGER DELAY      0   MS
   COMPRESSOR        OFF

   SAMPLE

   DISPLAY
```
Figure 1.


```
INDEX              ***  PAGE D READY  ***              VOICE: 1
COMMAND:                                               RTUNE
                       WAVEFORM DISPLAY                MODE:  4
DISPLAY    FORMAT:A  B    END SEG: 128  64  32    STEP:1 2 4 8
```
Figure 2.


```
INDEX              ***  PAGE D READY  ***              VOICE: 1
COMMAND                                                LTUNE
                       WAVEFORM DISPLAY                MODE:  4
DISPLAY    FORMAT:A  B    END SEG: 128  64  32    STEP:1 2 4 8
```
Figure 3.

rate results in a dreadful glitch which increases at a rate proportional to the pitch played on the keyboard. If the sample rate is almost right, a one-segment loop produces a sudden slight pitch shift, and waveform crests and troughs 'drift' laterally through a Page D display. This is shown in Figures 2 and 3, where a drift to the right (sharp) is caused by the sample being set too high and a drift to the left (flat) by it being too low.

Figure 4 shows a sound which is in tune with the system and therefore loops perfectly. At the other end of the scale, if the sample rate is totally wrong the display becomes a hopeless jumble (Figure 5). The relationship between a whole number of cycles and each segment of waveform RAM is also the relationship required for a visually coherent display. Thus samples that look good will inevitably sound good, too.

Now, if all this sounds rather complicated and you're beginning to wonder how anyone gets anywhere near choosing the correct sample rate, then take heart. It's all in the help pages for Page 8 – see Figure 6. A useful sample rate table is included, and with a little practice it becomes quite easy to arrive at the correct setting within the space of a few trial samples.

ADC

The actual analogue-to-digital conversion is accomplished by a 10-bit converter, even though the CMI is an eight-bit machine. Only the top eight bits of the sample values are stored, while the two LSBs (Least Significant Bits) are ignored. This improves the linearity of the conversion, which means that the signal step size required to cause a conversion value to change by one LSB is fairly constant over the range of the ADC.

The relationship between the amplitude of the input signal and the sample values generated is linear. When the signal level changes by a given amount irrespective of the absolute value the conversion code always changes by the same amount. This is where the Fairlight differs from most other sampling machines such as the Emulator. That uses a non-linear conversion method (called 'companding') which allows more codes to be generated for small signal values than for large ones. Whenever a signal is represented by a finite range of numbers – in this case 0–255 (eight bits) – two things suffer: noise and dynamic range. The noise is only heard when a sampled sound is actually being played through a DAC, ie. the ADC and DAC are not in themselves inherently noisy. Making sure that the peak of the input signal causes the maximum ADC code to be generated ensures that most of the noise is masked by the volume of the signal on playback.

The Display box in Figure 1 is an invaluable aid in this respect.

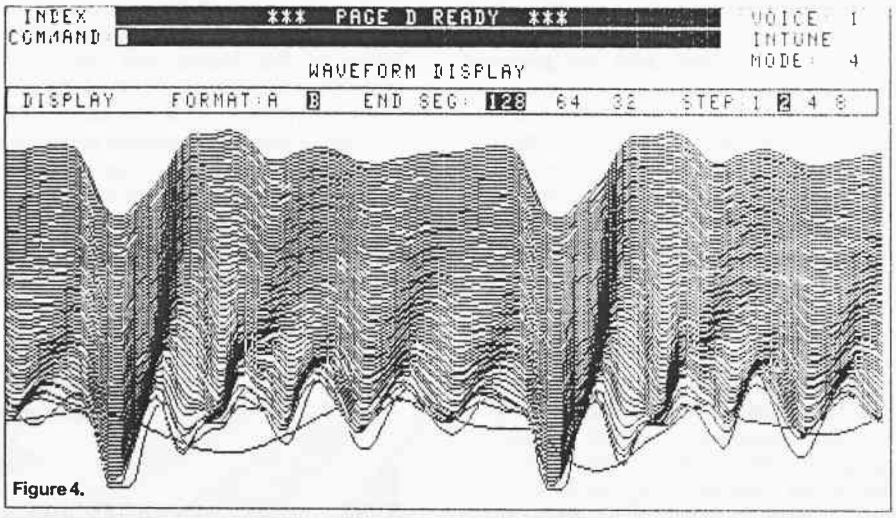
Dynamic range, on the other hand, is a measure of the range of different amplitude values that the ADC can handle. In a linear system, this is directly related to the number of bits used in the process and, roughly speaking, the dynamic range of the sampled signal is 6dB times the number of conversion bits. Since the Fairlight uses eight bits, this gives 8 × 6dB = 48dB dynamic range. Companding techniques result in a larger dynamic range (about 70dB) for the same number of bits used, but at the expense of greater noise at low signal amplitudes. The reasons for Fairlight's choice of a linear converter will become more apparent when we look at the functions on Page 6.

The actual sample rate is very cleverly generated on Channel card 1. Normally, the onboard circuitry is used to generate the correct clocking rates required for digital-to-analogue conversion when a keyboard note is pressed. However, for the duration of the sampling period the CPU grabs Channel 1, and forces it to produce a stream of pulses at a frequency of 128 times the sample rate shown on Page 8. This is supplied to the ADC, which resides on the Master card, and once the sampling process is finished the CPU restores Channel 1 to its original task.

Trigger Level is the amplitude threshold at which the sampling process is triggered to begin. When the Sample command is given, the system waits until this level is reached before proceeding. Once the threshold has been exceeded, it's possible to delay the conversion by using the Trigger Delay, which has a range of 0–65533 milliseconds. This can be especially useful when sampling from tape, for example, as a tone burst can be recorded shortly before the signal to be sampled and used instead of the signal itself to trigger the sampling process. Trigger Delay can then be used to define the precise point at which sampling will actually begin. This is extremely useful for sounds with a gentle attack such as slow strings.

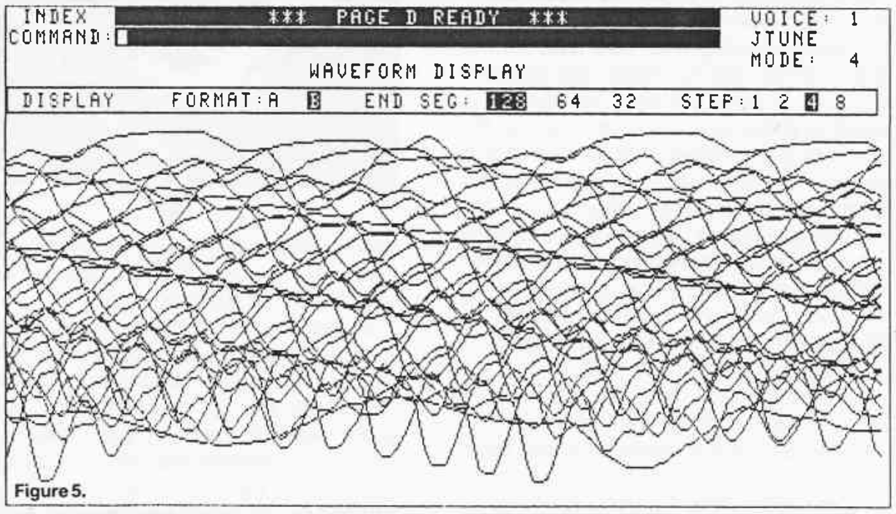
Lastly, the Compressor is a software switch which controls a hardware option. Basically, this turns the conversion process into a non-linear system, thus enhancing the dynamic range. The electronics use the same type of circuitry as that in many analogue companding systems. However, very few Fairlights are fitted with this option as it can have a strange effect on the commands on Page 6.

Well, that about wraps it up for Page 8. There isn't room this month for a discussion on Page 7, so we'll have to leave that for next month. ■



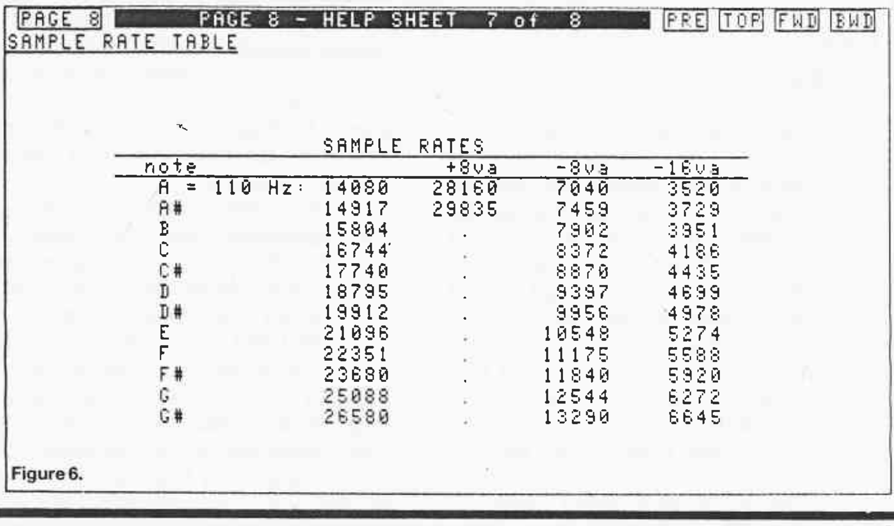
```
INDEX      ***  PAGE D READY  ***          VOICE: 1
COMMAND:                                   INTUNE
                WAVEFORM DISPLAY           MODE:  4
DISPLAY    FORMAT:A B    END SEG: 128 64 32    STEP: 1 2 4 8
```

Figure 4.



```
INDEX      ***  PAGE D READY  ***          VOICE: 1
COMMAND:                                   JTUNE
                WAVEFORM DISPLAY           MODE:  4
DISPLAY    FORMAT:A B    END SEG: 128 64 32    STEP: 1 2 4 8
```

Figure 5.



```
PAGE 8   PAGE 8 - HELP SHEET  7 of  8     PRE  TOP  FWD  BWD
SAMPLE RATE TABLE

                    SAMPLE RATES
  note                        +8va      -8va     -16va
   A = 110 Hz:  14080        28160      7040      3520
   A#           14917        29835      7459      3729
   B            15804          .        7902      3951
   C            16744          .        8372      4186
   C#           17740          .        8870      4435
   D            18795          .        9397      4699
   D#           19912          .        9956      4978
   E            21096          .       10548      5274
   F            22351          .       11175      5588
   F#           23680          .       11840      5920
   G            25088          .       12544      6272
   G#           26580          .       13290      6645
```

Figure 6.

THE FAIRLIGHT EXPLAINED

How the CMI provides for special effects and looping – in a language just about everyone can understand. *Jim Grant*

Last month we dealt with one of the Fairlight's main sound creation methods: sampling. The simple act of pointing a microphone at a sound source and typing 'S' on the alphanumeric keyboard transforms the Fairlight from an expensive computer into a powerful musical instrument. And the keyword here is 'musical'. The ability to create new and interesting sounds (or indeed sample them) is not in itself enough. What is required is *control* over that sound and, to coin a popular phrase, the control must be real-time. Musicians, of course, call this control 'expression', and it's a particularly difficult feature to build into a computer-based musical instrument.

Consider a typical case in which a Fairlight user might be playing the music keyboard while listening to a sequence pre-recorded on Page 9. Everything is running smoothly: the CMI is reading sequence information from the disk, sorting it out, and sending the data to the voice channels to be played. At the same time, the music keyboard is being scanned for pressed notes and more data sent to the channel cards: notes are stolen if necessary. Next the user may decide to swell a particular voice by moving the appropriate footpedal. Here the CMI is forced to deal with an asynchronous event in the normal proceedings, so the pedal value has to be updated constantly and the values obtained used to scale the amplitude of the voice throughout its duration. And if this were not enough, the Fairlight has 17 parameters capable of being controlled in real-time.

It's the unusual multi-processor architecture of the CMI that enables it to handle so many asynchronous tasks simultaneously, but let's move on to the presentation of the controls and their use.

Page 7

All the controls are handled by Page 7, and a typical display is shown in Figure 1. The page features all the usual controls associated with processing sound: each of the eight voices loaded can have its own unique control setting and can be saved to disk with a chosen filename and the suffix CO. When a voice is loaded into the CMI, it will pull in a specified control file if it was previously Linked to the voice, using the command LNK.

At the bottom of the display is a box which indicates currently-loaded voices. The control file may have a different name from its intended voice so the two are differentiated visually. The active control file is the name highlighted while the active voice is shown in the top right-hand corner. Other control files can be inspected by pointing the lightpen at the names in the display box or by typing 'V,n', where 'n' is the voice number.

There are six real-time faders and five switches patchable to most parameters. Three of the faders and two of the switches are on the left-hand side of the music keyboard, while the other faders (or footpedals) are accessible *via* Cannon-type connectors on the rear of the keyboard. In addition, the music keyboard passes key velocity information to the CMI, and this can be patched to Level and Attack as KEYVEL.

Below the voice list in Figure 1 is a complete list of the controls and switches that are available. This is used in conjunction with the lightpen and provides a quick way of patching the controls to various parameters: the lightpen is pointed first at the parameter and then at the control list. A patch can also be established by tabbing a cursor around the display using the QWERTY keyboard and typing in the appropriate name or numeric value. Figure 2 shows one of the 'Help' sheets for Page 7 which provide a quick reference for the range and possible patches available.

Some of the control parameters are self-explanatory, such as 'Level', 'Vib Speed' and 'Vib Depth'. Again, we come across the enigmatic 'Mode' switch, which is best left until Pages 4 and 5 are discussed (*the suspense is killing me – Ed*). 'EXP' is the other half of the companding process that was an option on Page 8 discussed last month. As you may remember, it's a hardware option and is very rarely fitted to the CMI due to the non-linear sampling data that results from its use. The Filter is a low-pass tracking filter resident on each Channel card, used to attenuate any unwanted high-frequency content present in the voice: the cutoff frequency is raised by simply increasing the value. It's all really a case of swings and roundabouts – a high filter setting gives a bright realistic sound but often with digital birdies warbling in the background, while low filter values suppress any funnies but reduce the sound to a dull noise.

When Portamento is on, each Channel allocated to the voice produces a continuous glide between each new pitch it is to play and the last pitch played, the rate of note glide being set by the Speed control. 'Glissando' differs from Portamento in that the glide is not continuous but chromatic, and all the notes on the keyboard between the start and end notes are played. If both Portamento and Glissando are selected, Portamento takes precedence. 'Constant Time' is a switch which selects between two types of glide: when it's turned on, the same time is taken to travel any musical interval and the rate of change alters according to that interval, hence the name 'Constant Time'. This results in polyphonic portamento or glissando, in which the notes arrive at their destinations at the same time producing a coherent chord. With the switch off, the rate of change remains fixed (determined by Speed) and the time taken to glide varies with the size of the interval.

Attack and Damping

The Attack parameter has a range of zero to 16,384 milliseconds, and may be patched to 'KEYVEL' for touch-sensitive control of the attack time. It's active only for Mode 4 sounds, and is extremely useful for imposing a degree of artificial enveloping upon sampled sounds. 'Damping' has a range of zero to 65,536 milliseconds, reduced to 16,384 milliseconds in Mode 4. The value determines the final decay time of the voice, ie. from key release to silence. If a loop is active and one or more segments are repeated continuously, the voice plays the loop until the damping time expires (when the key is released), otherwise the voice continues through the remaining segments. Should the end segment be reached before the damping time expires, the voice stops abruptly.

The 'Slur' switch is useful for glissando and portamento effects, as it causes Channel cards allocated to a voice to sustain indefinitely in a loop that may be active until a new note is played. New notes are started at the beginning of the loop without playing any of the preceding segments. 'Sustain' determines the behaviour of the voice once the key is released. Normally, a voice fades out either playing its loop or until it hits the end of the segments, but when Sustain is on, Damping is ignored and the voice loops for the duration of key depression: upon key release, the voice continues to play its remaining segments with no decay of amplitude.

Looping

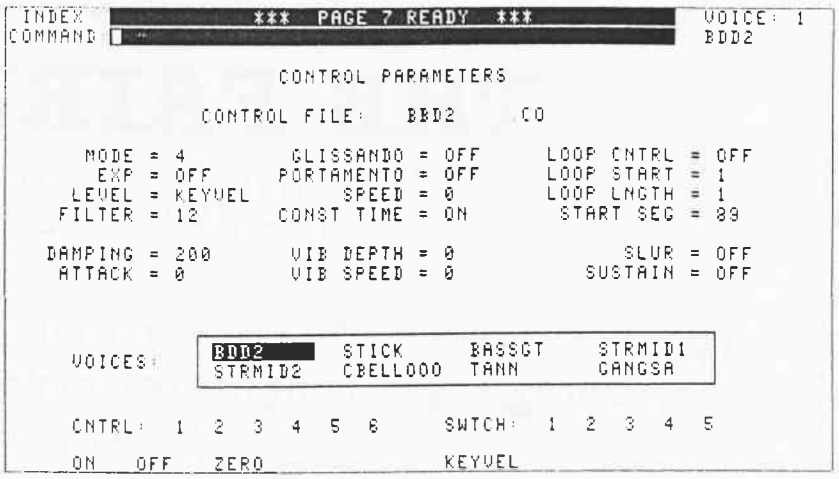
Choosing the correct looping point of a voice waveform can make or break a good sound on the Fairlight. Nasty glitches can occur if an inappropriate sample rate is chosen or if the section to be looped spans a natural change of amplitude.

Imagine trying to loop a percussive sound such as a drum. The three loop controls on Page 7 provide a quick way of finding the best loop, and a typical setting might be as shown in Figure 3. Here Control 1 is used to define start point of the loop while Control 2 sets the length. Switch 1 freezes the effect of Control 1 and Control 2 when off, preventing accidental movement of the looping points once these have been decided. A useful feature is that loop parameters are saved with the voice information as well as any Linked control file, so that the sound is playable even though no performance controls are required. The actual loop points are displayed graphically on Page 4, as shown in Figure 4, where the horizontal axis represents the segment number and therefore time. The loop is indicated by the row of highlighted boxes under the Harmonic Profiles graph, and since the voice shown was sampled, there are none of these present. This Page offers a convenient method of selecting looping points using the lightpen.

'Start Seg' is a powerful expression control. It allows the starting segment of the voice to be chosen according to a control value as a new key is played. To explain: suppose we had sampled the classic synthesiser filter sweep and playing the keyboard resulted in a 'fruity' decay. Using a control fader to set the start segment would then enable us to play the synth sound from different parts of the filter sweep. As the control was moved from segment 1 to 128, the sound would begin with plenty of filter sizzle at low control values but become shorter and more mellow as we started the sound further down the sweep (by increasing the Start Segment number). This technique can also be used to control the amount of 'breath' on sampled wind sounds or the amount of bowing on stringed instruments.

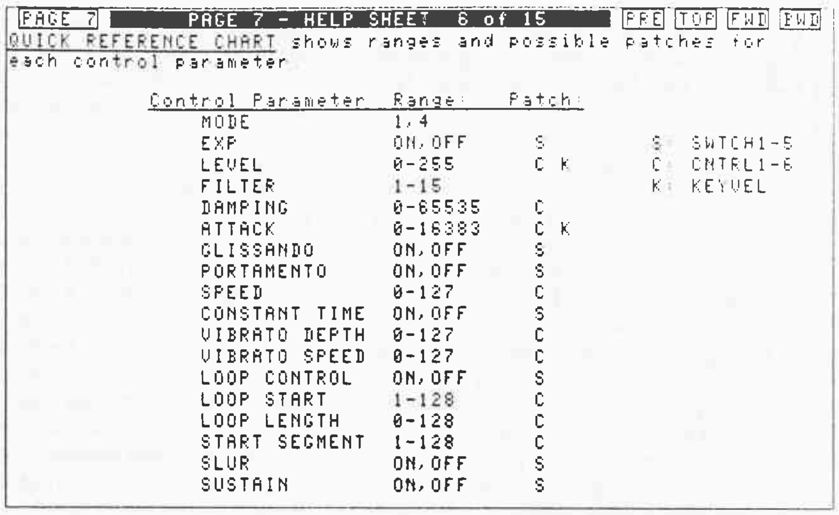
Next month, page 5 and some revelations concerning the mysterious 'Mode'. ■

Figure 1.



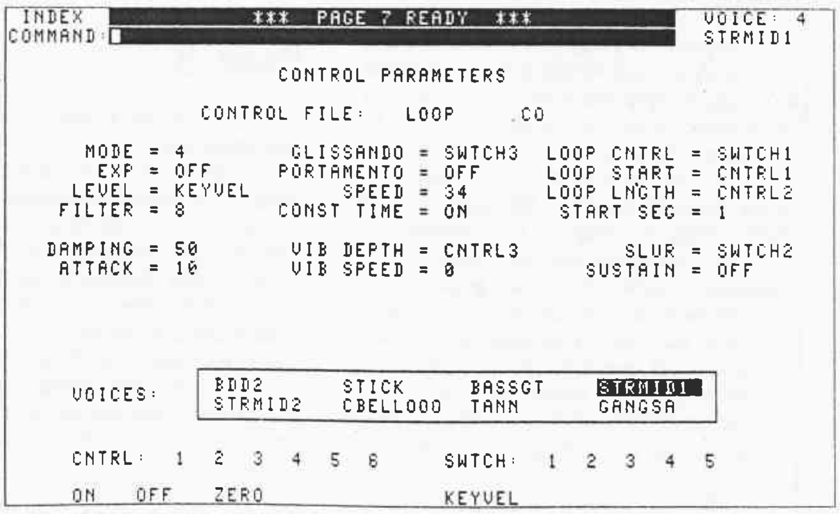
```
INDEX          ***  PAGE 7 READY  ***                    VOICE: 1
COMMAND:                                                 BDD2

                        CONTROL PARAMETERS

                   CONTROL FILE:   BBD2     .CO

     MODE = 4           GLISSANDO = OFF       LOOP CNTRL = OFF
      EXP = OFF        PORTAMENTO = OFF       LOOP START = 1
    LEVEL = KEYVEL          SPEED = 0         LOOP LNGTH = 1
   FILTER = 12         CONST TIME = ON         START SEG = 89

  DAMPING = 200         VIB DEPTH = 0               SLUR = OFF
   ATTACK = 0           VIB SPEED = 0            SUSTAIN = OFF


    VOICES:    BDD2      STICK      BASSGT     STRMID1
               STRMID2   CBELLOOO   TANN       GANGSA

    CNTRL:  1  2  3  4  5  6     SWTCH:  1  2  3  4  5

    ON   OFF   ZERO              KEYVEL
```

Figure 2.



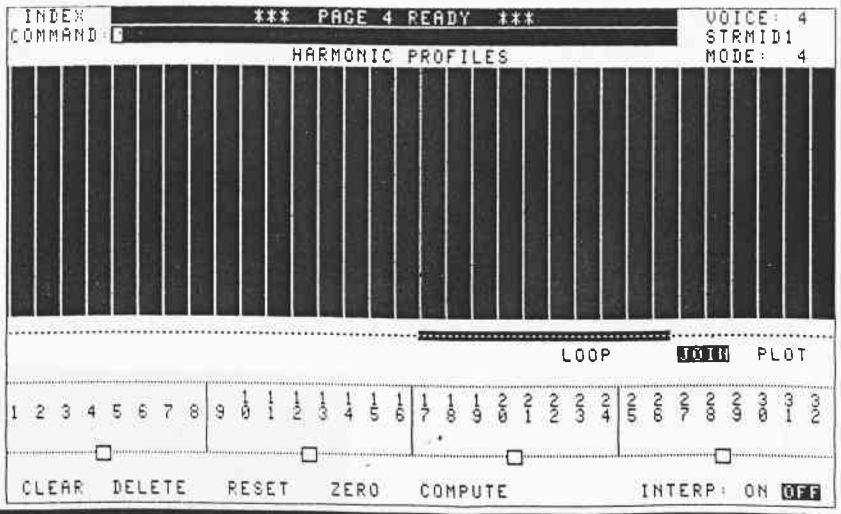
```
PAGE 7       PAGE 7 - HELP SHEET   6 of 15        PRE  TOP  FWD  FWD
QUICK REFERENCE CHART shows ranges and possible patches for
each control parameter

           Control Parameter  Range:     Patch:
               MODE           1,4
               EXP            ON,OFF      S          S: SWTCH1-5
               LEVEL          0-255       C K        C: CNTRL1-6
               FILTER         1-15                   K: KEYVEL
               DAMPING        0-65535     C
               ATTACK         0-16383     C K
               GLISSANDO      ON,OFF      S
               PORTAMENTO     ON,OFF      S
               SPEED          0-127       C
               CONSTANT TIME  ON,OFF      S
               VIBRATO DEPTH  0-127       C
               VIBRATO SPEED  0-127       C
               LOOP CONTROL   ON,OFF      S
               LOOP START     1-128       C
               LOOP LENGTH    0-128       C
               START SEGMENT  1-128       C
               SLUR           ON,OFF      S
               SUSTAIN        ON,OFF      S
```

Figure 3.



```
INDEX          ***  PAGE 7 READY  ***                    VOICE: 4
COMMAND:                                                 STRMID1

                        CONTROL PARAMETERS

                   CONTROL FILE:   LOOP     .CO

     MODE = 4           GLISSANDO = SWTCH3    LOOP CNTRL = SWTCH1
      EXP = OFF        PORTAMENTO = OFF       LOOP START = CNTRL1
    LEVEL = KEYVEL          SPEED = 34        LOOP LNGTH = CNTRL2
   FILTER = 8          CONST TIME = ON         START SEG = 1

  DAMPING = 50          VIB DEPTH = CNTRL3          SLUR = SWTCH2
   ATTACK = 10          VIB SPEED = 0            SUSTAIN = OFF


    VOICES:    BDD2      STICK      BASSGT     STRMID1
               STRMID2   CBELLOOO   TANN       GANGSA

    CNTRL:  1  2  3  4  5  6     SWTCH:  1  2  3  4  5

    ON   OFF   ZERO              KEYVEL
```

Figure 4.



```
INDEX          ***  PAGE 4 READY  ***                    VOICE: 4
COMMAND:                                                 STRMID1
                      HARMONIC PROFILES                  MODE:  4
```

THE FAIRLIGHT EXPLAINED

Sampling may be the CMI's most talked-about feature, but as this article shows, defining sounds using harmonic information can be just as dramatic.

Jim Grant

So far we have discussed only one method of actually creating sounds with the Fairlight sampling. This aspect of sound formation is probably the single most important feature of the CMI, and was certainly the focus of public attention when the machine was announced. However, the ability to specify sound by means of harmonic information can not only result in some very interesting sounds, it's also rather useful in an educational environment.

Two display Pages, 4 and 5, allow the construction of waveforms by harmonic data. They deal with exactly the same information but present it to the musician in different ways.

First of all, though, let's clear up a little mystery that's been evading us for some months – the Mode switch. Actually, this is very simple and is therefore something of an anti-climax. When a voice operates in Mode 1, only the first 32 segments of waveform RAM (4k bytes) are used to represent the sound. An unlooped Mode 1 sound will stop at the 32nd segment, even though another 96 segments of RAM exist. In order to compensate for shorter note event time as played on the keyboard, each of the 32 segments is looped several times before moving on to the next segment: this maintains a fairly constant net event length for any pitch. Mode 4 uses the entire waveform RAM (all 128 segments of it) and is always used for sampling since long, high bandwidth sounds need lots of numbers to represent them.

So what's the use of Mode 1? Well, calculating a time waveform from harmonic data can be quite time-consuming, especially if the supplied data is detailed and enables subtle nuances of sound to be generated. However, more often that not, only a simple waveform is required, and to calculate the RAM waveform for all 128 segments when a short loop is all that's needed is rather wasteful, to say the least. There's no hard and fast rule about which Mode a sound should be in: the choice is entirely the musician's. However, using a voice as the destination for sampling data *always* results in all 128 segments being overwritten, even if the voice selected is Mode 1.

Page 5

Figure 1 shows a typical Page 5 display. This page displays the harmonic overtone series as a set of 32 'faders' similar to those on a graphic equaliser. Each fader is logarithmic in nature and has a range of zero to 255, allowing a good degree of control over harmonic amplitudes and thus enabling the application of a Fourier type harmonic series.

As an example, Figure 2 shows a square wave generated by the CMI, computed from the values of Fourier components shown on the faders. The resultant waveform is visually very similar to the real thing, and perhaps more importantly, sounds indistinguishable.

However, the power of the CMI lies in its ability not only to compute a complex waveform from a set of Fourier components so that it can be played on the keyboard, but also to compute a different waveform for each segment. Every segment has a unique Page 5 display, so while a Mode 1 sound has 32 sets of faders, a Mode 4 one will have 128! The current segment number is indicated on the display, and this allows a synthesised sound to change drastically throughout its duration, simply by the user filling each segment with a different waveform calculated from its own Page 5 fader settings.

In fact, the technique of using different waveform segments as the sound progresses is very much the domain of PPG synthesis. Generally speaking, these progressions are known as wavetables, and in the PPG, a sound consists of a set of 64 waveforms that reside initially in EPROM (they are transferred to RAM on power-up) which are read out sequentially when a key is pressed. The idea behind this system was to circumvent the need for filters by constructing wavetables that held a set of representative waveforms of, say, the classic filter sweep. Unfortunately, this results in a very hard, metallic sound, as the sound changes abruptly from one waveform to another, slightly different one. It's still a good sound, but in the interests of flexibility, PPG have chosen to incorporate the usual VCFs and ADSRs as well as extensive wavetable modulation.

The CMI is also capable of this form of synthesis to a limited degree, using the loop controls on Page 7. For example,

Figure 2.

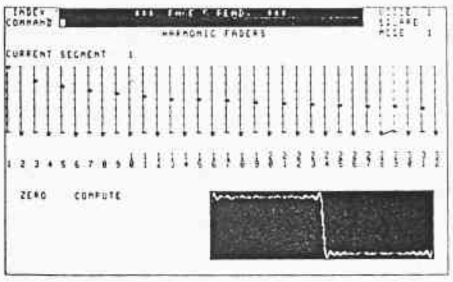
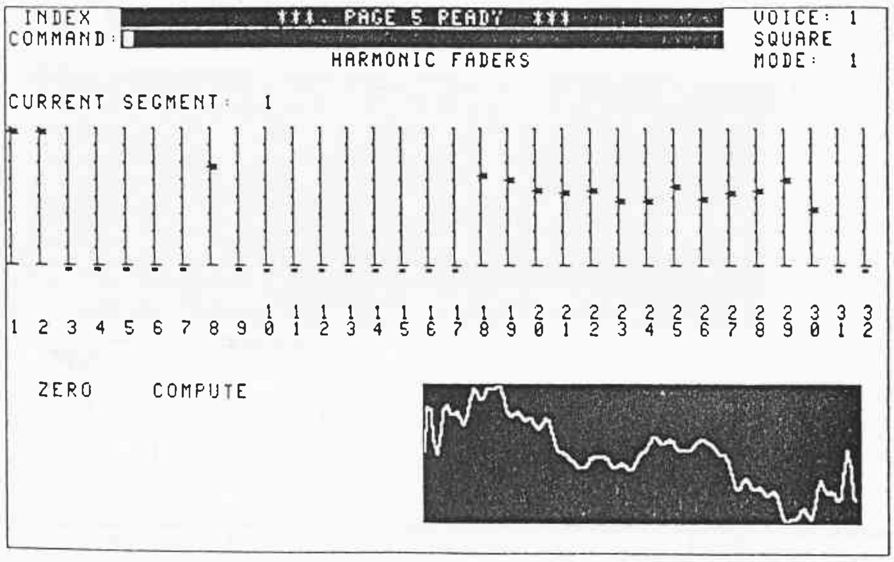

```
INDEX          *** PAGE 5 READY ***          VOICE: 1
COMMAND:                                      SQUARE
                  HARMONIC FADERS             MODE: 1
CURRENT SEGMENT: 1

ZERO    COMPUTE
```

Figure 1.


```
INDEX          *** PAGE 5 READY ***          VOICE: 1
COMMAND:                                      SQUARE
                  HARMONIC FADERS             MODE: 1
CURRENT SEGMENT: 1

                  1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2

ZERO    COMPUTE
```

suppose we had filled all 128 segments with waveforms that change very slightly as we progress through the waveform (see Figure 3). Now, if the loop controls were set up as shown in Figure 4 (this is a Page 7 display), moving CNTRL1 on the music keyboard would result in a different timbre when the note was played. Using this technique allows for some expressive playing, since the principle is rather akin to varying the filter frequency control on a synthesiser, the only difference being that the actual timbres can be radically different from one segment to the next.

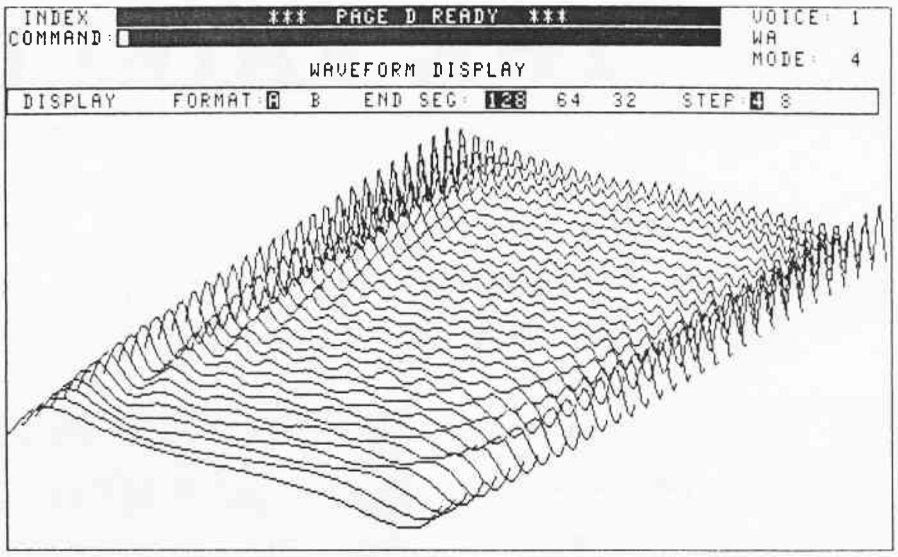
To increase the timbral movement within a sound, CNTRL2 can be patched to 'LOOP LENGTH' on Page 7, resulting in sections of different waveforms being read out repeatedly. Since the waveform data is computed by the CMI, it's always constructed so that the waveform fits exactly into one segment, thereby overcoming looping problems.

Fourier Series

Well, with all this talk of 'Fourier components' and the like, some of you may reasonably be thinking 'what's it got to do with music?' The answer, of course, is not much. Only scientists and engineers delight in quantifying the world which our senses seem to handle perfectly adequately. However, in order to express ourselves explicitly and unambiguously about a wide variety of concepts (some of which may be abstract) we need to use the language of mathematics. Fourier analysis and synthesis are mathematical statements about something which is not intuitively obvious: the fact that any truly periodic waveform can be decomposed into an infinite sum of sinewaves, usually called harmonics. Similarly, any periodic waveform can be constructed from the sum of an infinite number of sinewaves. The sinewaves have frequencies that are related to the fundamental of the waveform in such a way that the second harmonic lies at twice the fundamental frequency, the third harmonic lies at three times the fundamental, and so on. The fundamental itself is often referred to as the first harmonic.

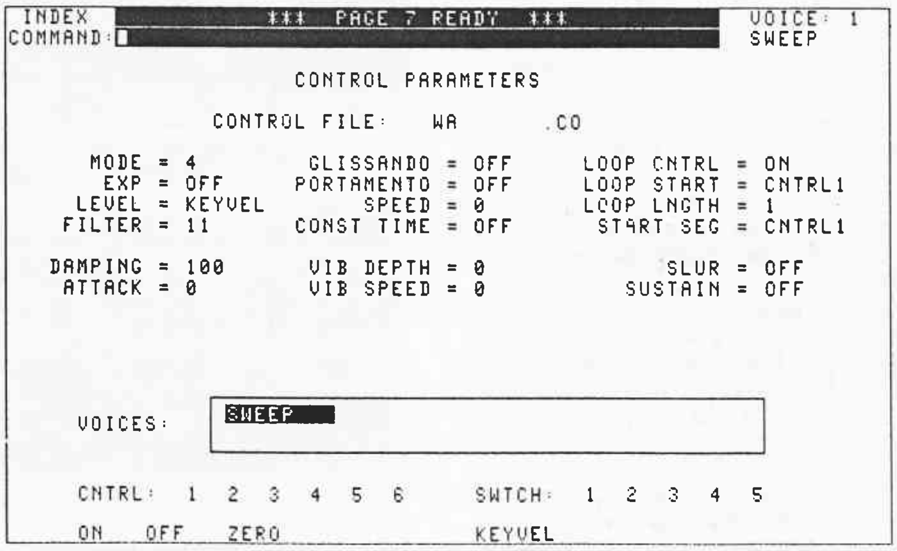
Of course, obtaining a reasonable representation of a desired waveform does not require an infinite number of sinewaves: more than 16 is enough to give a good approximation. The Fairlight uses a maximum of 32 harmonics, which enables most waveforms to be synthesised with a fair degree of accuracy. Figures 5 to 8 show the development of a square wave by successively adding further harmonics and using Page 5 to compute the resultant waveform. The square wave doesn't contain any even harmonics (2, 4, 6 and so on) and we can see that the lower harmonic numbers set the basic square shape while the higher ones fill in the bumps and sharpen the edges. Even with all the odd harmonics the Fairlight can compute, the square wave is still not visually perfect, but it sounds OK. ■

Figure 3.



```
INDEX     *** PAGE D READY ***                       VOICE: 1
COMMAND:                                             WA
                                                     MODE:  4
                     WAVEFORM DISPLAY
DISPLAY    FORMAT: A  B    END SEG: 128  64  32    STEP: 4  8
```

Figure 4.



```
INDEX     *** PAGE 7 READY ***                       VOICE: 1
COMMAND:                                             SWEEP

                    CONTROL PARAMETERS

              CONTROL FILE:   WA       .CO

   MODE = 4          GLISSANDO = OFF     LOOP CNTRL = ON
    EXP = OFF       PORTAMENTO = OFF     LOOP START = CNTRL1
  LEVEL = KEYVEL         SPEED = 0       LOOP LNGTH = 1
 FILTER = 11        CONST TIME = OFF      START SEG = CNTRL1

DAMPING = 100        VIB DEPTH = 0             SLUR = OFF
 ATTACK = 0          VIB SPEED = 0          SUSTAIN = OFF



  VOICES:      SWEEP


  CNTRL:  1   2   3   4   5   6      SWTCH:  1   2   3   4   5
  ON     OFF     ZERO                KEYVEL
```

Figure 5.

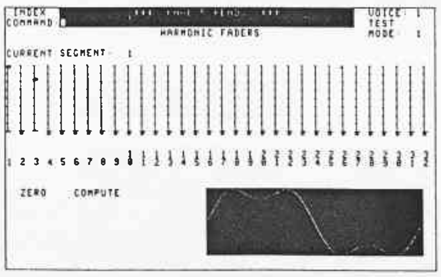


Figure 6.

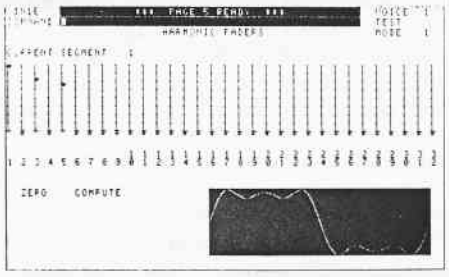


Figure 7.

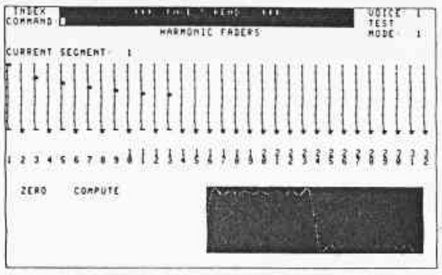
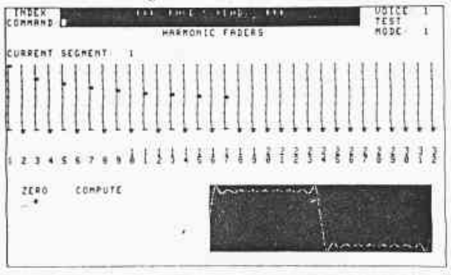


Figure 8.



THE FAIRLIGHT EXPLAINED

Waveforms, lightpens and interpolation all come under examination in this instalment of our Fairlight CMI Grand Tour. *Jim Grant*

Just when you thought it was safe to open up a copy of E&MM without reading anything about the world's most influential computer musical instrument, your intrepid reporter returns from a New Year hangover with another action-packed episode. This month we look at the information presented by Page 5 in a slightly different light. You'll remember that Page 5 held the values for 32 harmonic faders and computed the resultant waveform for the current segment. You should recall also that the only way to create a complete sound of 32 segments was to define the fader levels for each segment and compute over the whole waveform; or define a few segments and Fill the harmonic data to the rest of the segments before computing. It's not hard to see that this method of creating sounds may be very precise but can also be extremely tedious. In a lot of cases, all we need is a way of tailoring the harmonics as the sound progresses: harmonic envelopes, in other words. Enter Page 4.

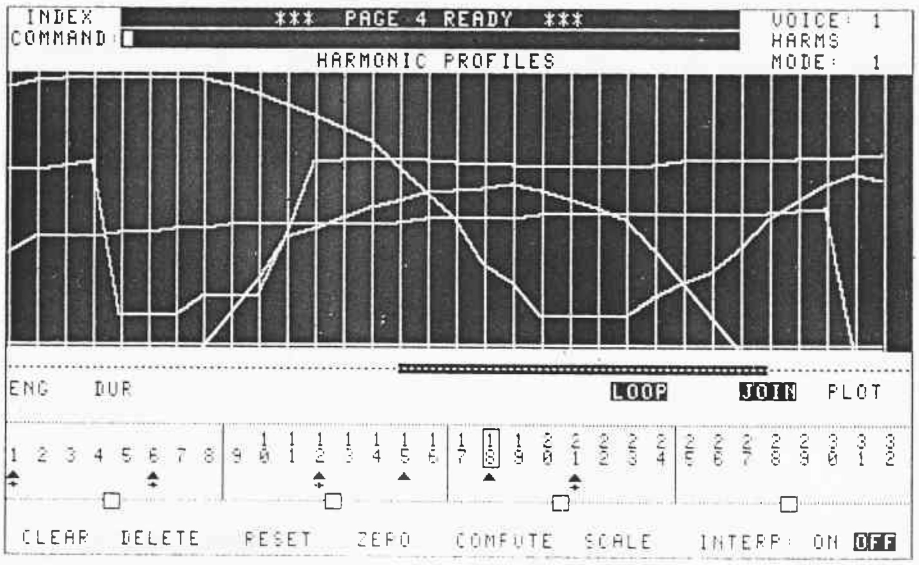
Figure 1 is a typical Page 4 display, and shows that it's one of the two Fairlight display Pages to be almost exclusively lightpen-driven. The large dark area is in fact a reverse video image, and pointing the lightpen in this region results in an arrow cursor appearing on the screen at the current lightpen position. For those unfamiliar with the term, the lightpen is now a fairly common computer add-on, mainly because of its simplicity of operation. Contained within every lightpen is a fast photoelectric diode or transistor which produces a voltage pulse as the TV line passes beneath it. Usually, the pulse is squared up and passed to the video controller chip, which stores the TV line number and position along the line in a couple of registers. This information can then be used by the programmer to initiate predefined events such as plotting a point or executing a command.

The Fairlight system is no different, except that the video controller is constructed from discrete logic chips and resides on a single eight-inch board within the CPU. In addition to latching the TV co-ordinates when the lightpen is used, it generates an interrupt to the processors to execute the selected task.

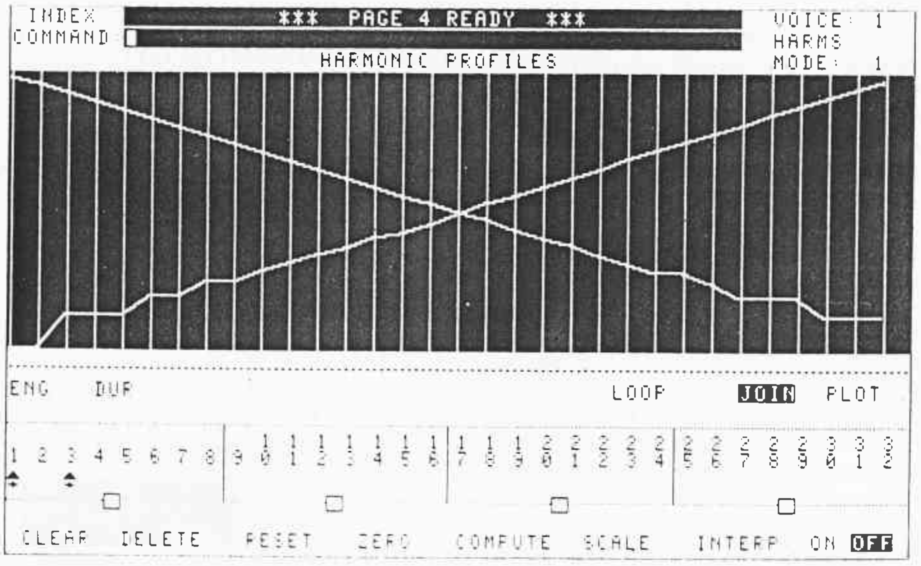
At first glance, the graph area in Figure 1 looks a bit confusing but it's really quite straightforward: the vertical axis represents amplitude while the horizontal shows time and hence the segment number.

Along the bottom of the display are the harmonic numbers 1 to 32. A small triangle under the number indicates that the time profile of that harmonic is being displayed on the graph, while a cross shows that the profile has a non-zero value.

So what does all this mean? Have a look at Figure 2. Two profiles are shown, one of which is the First harmonic (left to right downwards) and the other the Third (left to right upwards). On receipt of a Compute command, the CMI will fill the waveform segments with sound which initially at least, has a strong fundamental



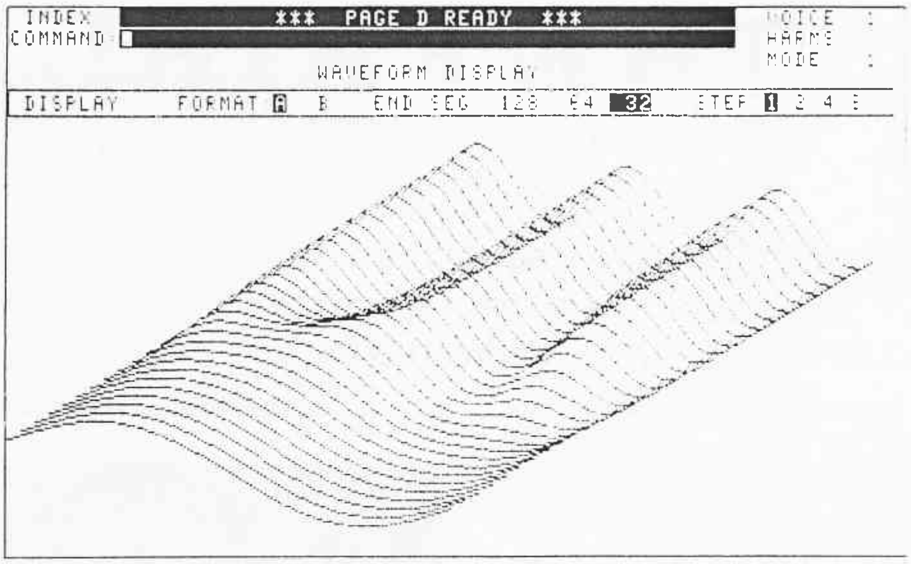
```
INDEX          *** PAGE 4 READY ***          VOICE: 1
COMMAND                                      HARMS
                HARMONIC PROFILES            MODE:  1

ENG   DUR                         LOOP    JOIN    PLOT

1 2 3 4 5 6 7 8 | 9 10 11 12 13 14 15 16 | 17 18 19 20 21 22 23 24 | 25 26 27 28 29 30 31 32

CLEAR  DELETE  RESET  ZERO  COMPUTE  SCALE  INTERP: ON OFF
```

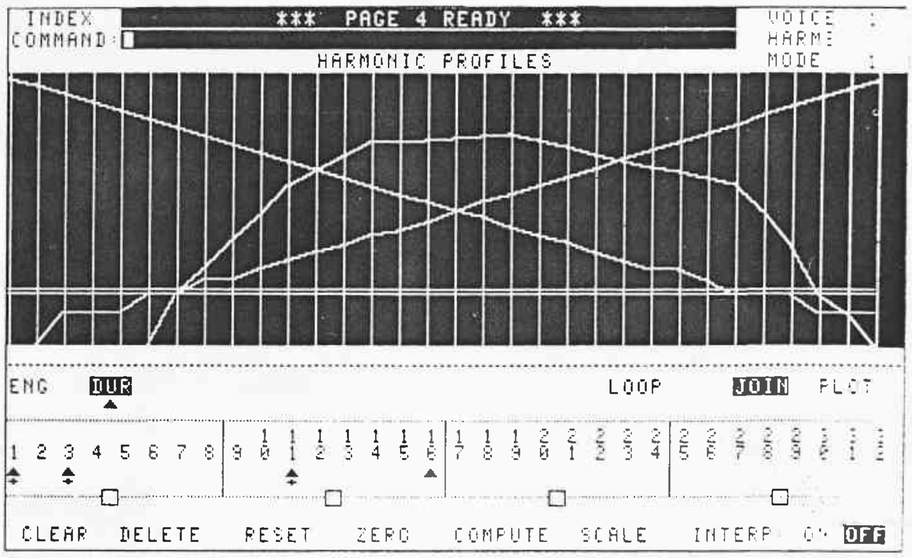


```
INDEX          *** PAGE 4 READY ***          VOICE: 1
COMMAND                                      HARMS
                HARMONIC PROFILES            MODE:  1

ENG   DUR                         LOOP    JOIN    PLOT

1 2 3 4 5 6 7 8 | 9 10 11 12 13 14 15 16 | 17 18 19 20 21 22 23 24 | 25 26 27 28 29 30 31 32

CLEAR  DELETE  RESET  ZERO  COMPUTE  SCALE  INTERP  ON OFF
```

ut degenerates into dominant third armonic.

You don't believe it? Look at Figure 3. This is great, because with 32 harmonics at our disposal, we can create sounds with interesting harmonic structures quickly and easily by selecting harmonics and waving the lightpen in the general direction of the profile area. Another bonus is that the profile data is mirrored segment by segment on the Page 5 faders, allowing detailed harmonic micro-surgery of the sound. Remember, though, that not every sound is created from harmonic data so that if, for instance, you're working on a sampled sound, calling up Page 4 will result in a completely lank profile graph.





More About Mode

One of the previously mentioned features of a Mode 1 voice is the way in which the first 32 segments are looped several times to maintain the net event time of the sound across the keyboard. In fact, we have some control over how long a segment lasts before everything moves on to the next one, and this is accomplished *via* the profile. Figure 4 shows a harmonic profile graph with the duration profile indicated by a double line: the default value is approximately 50mS per segment and increases as the profile is drawn higher up the graph. This is particularly useful for creating sounds with a short click at the beginning of each note, such as that of a Hammond organ. A very short duration value can be drawn for the first one or two segments, and then a longer profile for the remainder of the sound. If the duration profile is made zero, the sound degenerates to a Mode 4 condition, except that it only lasts for 32 segments.

Another interesting aspect of Mode 1 sounds is their ENG profile. This is an artificial envelope that's superimposed on the waveform in much the same way as the more usual ADSR principle. But this one's a lot more flexible. When the Compute command is given, the CMI calculates the waveform segment by segment and scales the amplitude so that it fits exactly into the dynamic range of eight bits. The ENG profile is also generated (and its shape implied) by the harmonic data, but can be altered by the lightpen to control the amplitude of the sound on playback.

Auxiliary Functions

Along the very bottom of the display Page are a number of useful commands. Clear deletes all the displayed profiles from the graph, but they remain active and can be brought back to life simply by the programmer selecting the harmonic numbers with the lightpen. Delete, on the other hand, merely removes the currently-selected profile. The same sort of structure applies to Reset and Zero. Invoking Reset causes a confirmation message to be printed and also results in Page 4 being restored to a complete default condition. Zero isn't quite so drastic, and results only in the current profile being set to zero. Every time you give a Compute command, a new energy profile is generated. Scale is the opposite: it redraws the harmonic profiles from a modified energy profile. This is not without its dangers, however, as it can result in some harmonic profiles being scaled beyond their maximum amplitude, which leads to clipping. The Fairlight *will* inform you of the situation when it occurs (by displaying 'Overflow') but is powerless to prevent it happening if the Scale command is issued. The only way to recover the sound then is to reload the voice.

Time now to introduce another concept with which some of you may not be overly familiar. Interpolation is the skill of guessing an unknown value that lies between two known ones, and is commonly used to predict values of points on graphs that aren't the actual ones originally plotted. When the Interp switch is On, each waveform segment is computed from a mix between the harmonic profiles of that segment and those of the next one. The difference between the two is subtle, and is only really noticeable when the profiles contain rapid changes throughout the duration of the sound.

Incidentally, becoming proficient at using the lightpen for drawing can take a lot of practice, so the CMI helps out by providing a Join Plot selector: when Join is active, any two points struck on the graph are immediately connected by a straight line, while fine detail can be drawn by selecting Plot. I imagine that most of you will be familiar with the Fairlight's Loop function by now, so I won't go through it all again. Suffice to say then that Page 4 offers a quick way of drawing the loop start and length. Mode 1 sounds are always calculated so that the waveform fits perfectly into a segment: the first harmonic does one cycle, the second harmonic two cycles, and so on. Gone are the Bad Old Days of trying to sample a sound to make it fit segments evenly. All you have to do now is use any old loop to span the sections of a waveform that are of interest, and Bob Moog's your uncle.

Next month (yes, there's still more to come), we'll take a look at Page 6 which, among many other weird and wonderful things, allows you to splice a sound down to no more than 16,384th of its length. And you thought a razor blade was powerful . . . ■

THE FAIRLIGHT EXPLAINED

When a CMI page is as helpful a source of music graphics as Page 6, you can bet your life it needs a lot of explaining – so this month's episode is dedicated entirely to it. *Jim Grant*

If you're trying to create a new sound on the Fairlight, whether it's from existing sampled data using Page 8 or harmonic profiles generated on Pages 4 and 5, you can be sure that Page 6 will be referred to over and over again.

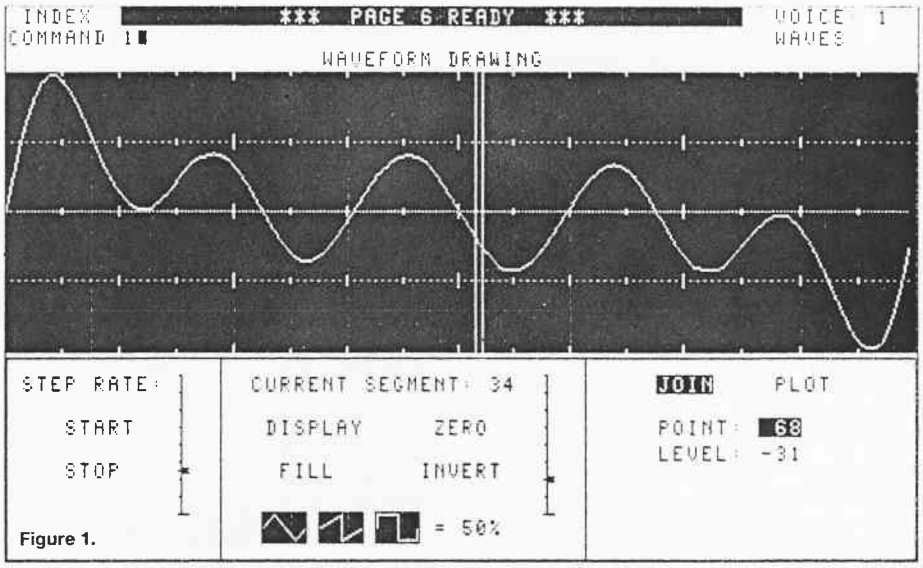
At its simplest level, Page 6 is a static 'oscilloscope' containing one segment of the waveform RAM. Like Page 4, the large black area is a reverse video image and represents the region that can be hit by the now-infamous lightpen. But unlike an oscilloscope, which acts only like a window on the information, Page 6, in conjunction with its commands and lightpen, behaves more like a door through which we can directly access the waveform data.

The Manual Approach

Regular readers of this series will probably recall that each segment consists of 128 bytes, and that each byte can hold a range of values from -128 to 127, giving a grand total of 256 possible values. Now, we can highlight any particular byte by positioning a narrow vertical window over the display area. The byte in question, shown in Figure 1, has its position indicated by POINT and its value by LEVEL: these can be changed by typing in different values from the alphanumeric keyboard. So, theoretically at least, you could type in 128 levels for each byte shown in Page 6, and repeat the process for all 128 Page 6 displays to cover the entire sound, giving 16,384 levels in all. In practice though, such a feat would take an unbelievable amount of manual labour (exactly what much of the Fairlight's software was designed to eliminate) to achieve, and this has led me to wonder whether this feature will be dropped when the Series III Fairlight, complete with waveform RAM of megabyte dimensions, makes its appearance later this year.

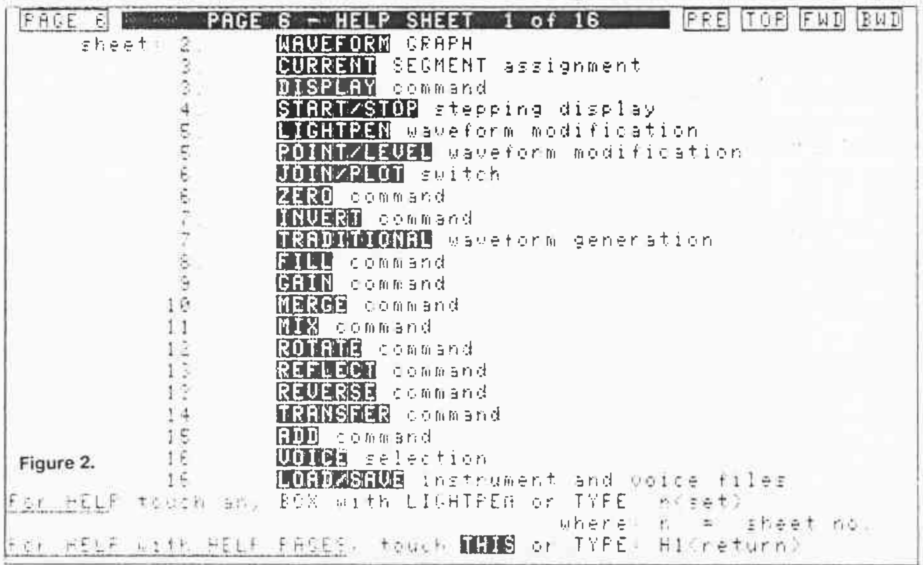
A more practical approach is, of course, to make use of the lightpen. This is as simple as drawing on the back of a bus ticket, but just in case even that is beyond the user's artistic capabilities, the JOIN and PLOT functions also present on Page 4 are available here, too, to help out with the drawing of geometric shapes.

In fact, trying out a few sample sketches soon reveals that waveforms which look drastically different may *sound* surprisingly similar. Scientifically speaking, this is probably due to the fact that as a



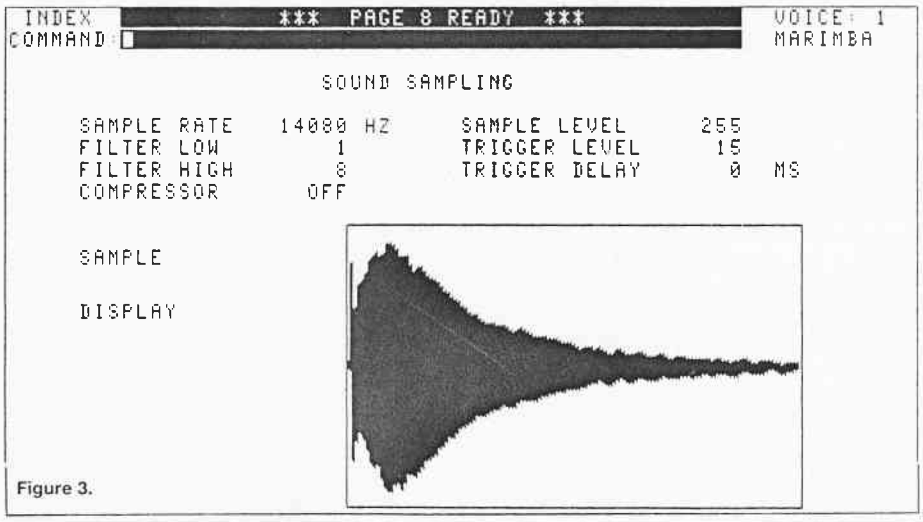
```
INDEX          ***  PAGE 6 READY  ***          VOICE  1
COMMAND 1                                      WAVES
                    WAVEFORM DRAWING

STEP RATE:     CURRENT SEGMENT: 34       JOIN     PLOT
  START        DISPLAY      ZERO         POINT: 68
  STOP         FILL         INVERT       LEVEL: -31
                             = 50%
```

Figure 1.



```
PAGE 6    PAGE 6 - HELP SHEET  1 of 16    PRE TOP FWD BWD
    sheet: 2      WAVEFORM GRAPH
           3      CURRENT SEGMENT assignment
           3      DISPLAY command
           4      START/STOP stepping display
           5      LIGHTPEN waveform modification
           5      POINT/LEVEL waveform modification
           6      JOIN/PLOT switch
           6      ZERO command
           7      INVERT command
           7      TRADITIONAL waveform generation
           8      FILL command
           9      GAIN command
          10      MERGE command
          11      MIX command
          12      ROTATE command
          13      REFLECT command
          13      REVERSE command
          14      TRANSFER command
          15      ADD command
          16      VOICE selection
          16      LOAD/SAVE instrument and voice files
For HELP touch any BOX with LIGHTPEN or TYPE  n(set)
                                   where: n  =  sheet no.
For HELP with HELP PAGES, touch THIS or TYPE: H1(return)
```

Figure 2.



```
INDEX          ***  PAGE 8 READY  ***          VOICE: 1
COMMAND:                                       MARIMBA

                    SOUND SAMPLING

SAMPLE RATE   14080 HZ     SAMPLE LEVEL     255
FILTER LOW        1        TRIGGER LEVEL     15
FILTER HIGH       8        TRIGGER DELAY      0   MS
COMPRESSOR      OFF

SAMPLE

DISPLAY
```

Figure 3.

species, we have little prior experience gained from our senses that relates visual time domain waveforms to what we perceive in terms of harmonics or partials.

But Page 6 waveform-drawing is of enormous help, first because it gives the user extensive and direct control of waveform RAM with one sweep of the hand, and second because the ability to bring about such drastic sonic change so simply and visually (almost artistically, some might say) is a practicality light years away from the sophistication and complexity of the underlying machine and its software. Or to put it another way, it makes the CMI seem less of a beast and more of a pet.

The Practical Approach

Alteration of the display image is not reflected in the waveform RAM until the FILL command is used. This allows the reverse video field to be used as a scratchpad area in which waveshapes can be developed before they are finally committed to RAM.

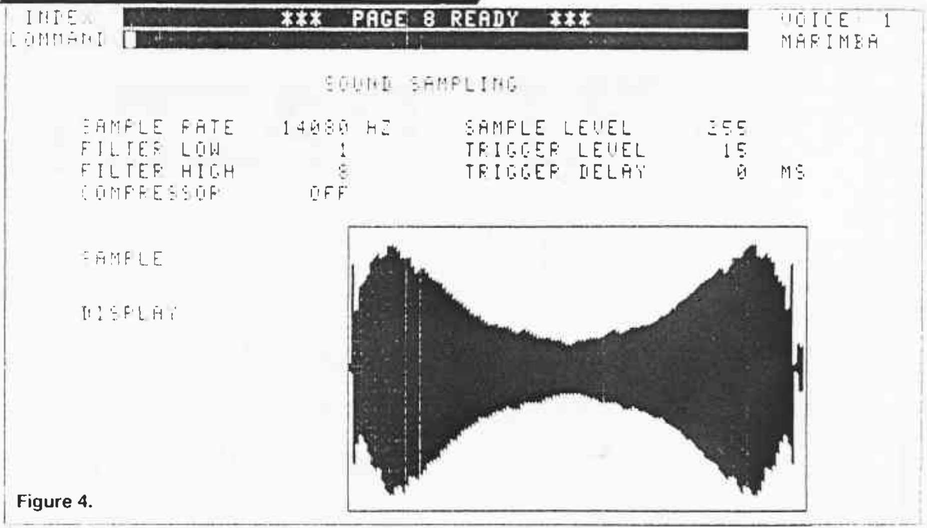
Looking again at Figure 1, notice there are two 'slider' controls similar to those on Page 5. These control the segment number to be displayed and the stepping rate through the entire memory instigated when the command Step is issued. Also shown are three classic waveforms – triangle, sawtooth, and square – which deposit a perfectly fitting single cycle wave into the segment. The pulse width of the square can be varied from 1% to 99% by changing the value held next to the square wave symbol.

Auxiliary Commands

In addition to the graphic capabilities of Page 6, there are a number of commands that can only be entered from the Fairlight's alphanumeric keyboard. Figure 2 shows the range of functions available, as presented by the HELP sheet menu.

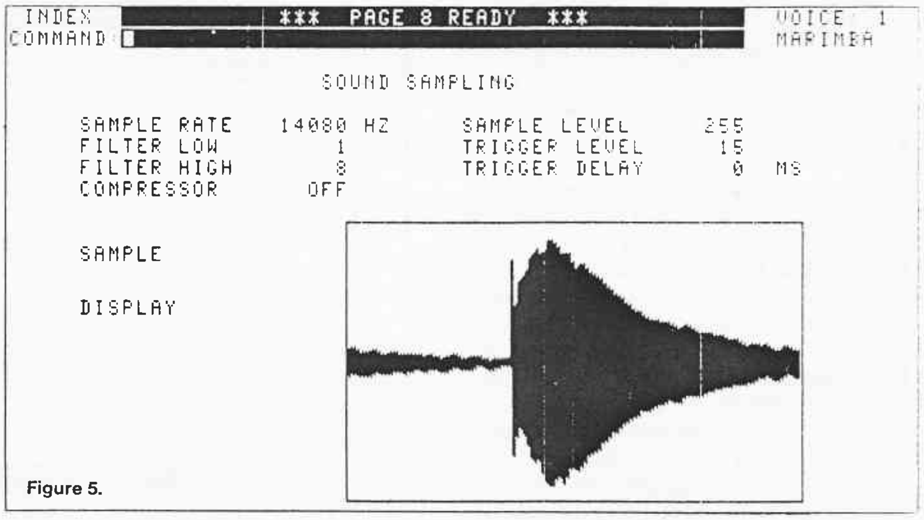
First off are a few utility-type commands. GAIN scales the displayed data by a specified percentage: if the rescaled waveform is about to exceed the amplitude range of the system, the CMI will ask you whether or not you want to proceed and thereby induce clipping. If you reply in the affirmative, the command will be duly executed. Meanwhile, the INVERT command inverts the phase of the waveform: this is useful as a prelude to some of the other functions such as MIX, MERGE and ADD.

A particularly neat little command is ZERO, which allows us to create a null voice in preparation for the ADD command. The entire waveform RAM can be



```
INDEX   *** PAGE 8 READY ***                       VOICE 1
COMMAND                                            MARIMBA

                      SOUND SAMPLING

  SAMPLE RATE    14080 HZ      SAMPLE LEVEL     255
  FILTER LOW         1         TRIGGER LEVEL     15
  FILTER HIGH        8         TRIGGER DELAY      0  MS
  COMPRESSOR       OFF

  SAMPLE

  DISPLAY
```

Figure 4.



```
INDEX   *** PAGE 8 READY ***                       VOICE 1
COMMAND                                            MARIMBA

                      SOUND SAMPLING

  SAMPLE RATE    14080 HZ      SAMPLE LEVEL     255
  FILTER LOW         1         TRIGGER LEVEL     15
  FILTER HIGH        8         TRIGGER DELAY      0  MS
  COMPRESSOR       OFF

  SAMPLE

  DISPLAY
```

Figure 5.

turned end on end by using REVERSE, and this results in the backwards sounds that have become familiar to pop music followers the world over.

REFLECT is a less commonly-used but if anything more interesting variation on this theme. It allows us to place an imaginary mirror in the sound and reflect every part of the waveform in front of this mirror to waveform RAM behind it. Take a look at Figures 3 and 4 for before and after views: the 'mirror' is at segment 64.

Now on to another of the seemingly insurmountable problems brought on by the onset of new musical technology. Very often, when a sound has been sampled, the beginning of the captured data does not occur at exactly the start of the RAM, perhaps due to some extraneous noise pretriggering the ADC. However, if the effect isn't too severe, a convenient method of correction is to ROTATE the sound within the waveform RAM to bring the start in line with byte 1. This has the (often desirable) side-effect of shifting the first part of RAM to the end. See Figures 5 and 3 for another revealing before and after picture.

I imagine NOISE will be fairly self-explanatory to even the least clued-up of this column's readers: it fills sections of the RAM with the output of a random number generator hidden inside the software. Funnily enough, the GAIN command is also used to tailor the amplitude of the noise to that of the rest of the sound...

Now, BLEND is a strangely out of place command. Personally, I think it should be on Page 4, since its role is to help smooth out glitches caused by imperfect looping points. It may be that to find a good loop it makes extensive use of extrapolation techniques, and that seeing as this is a central feature of the MIX and MERGE commands, Fairlight decided to bung it on Page 6 alongside them.

However, rather than discuss the remaining Page 6 commands in any detail right now, I think it's probably best for all concerned if I demonstrate their power in the context of a typical edit session in which a sound is created from scratch, ie. without any sampled data. So, now you know what I'll be talking about next month. ■

AH! GETTING A BIT OVERDRAWN AT THE BANK, TIME TO COMPOSE ANOTHER MEGA-SMASH HIT SINGLE, I THINK!
OH NO!!
BANK
FOSTERLITE CMI
HMMMM....THINK I'LL START WITH A 4/4 RHYTHM....
SURPRISE, SURPRISE!
TAP TAP
FOSTERLITE CMI
...SOME STRING AND BRASS PRESETS, A TOUCH OF "ORCH 5" AND SOME SCRATCHING EFFECTS....
..."ONLY LIMITED BY YOUR IMAGIN-ATION"....PAH!!
TAP TAP
FOSTERLITE CMI
5 MINUTES LATER...
...ANOTHER MASTERPIECE FINISHED! THINK I'LL JUST DO FOUR REMIXES OF THIS ONE!!
...GIVES ME A HEAD-ACHE JUST TRYING TO THINK DOWN TO HIS LEVEL!!
FOSTERLITE CMI

THE FAIRLIGHT EXPLAINED

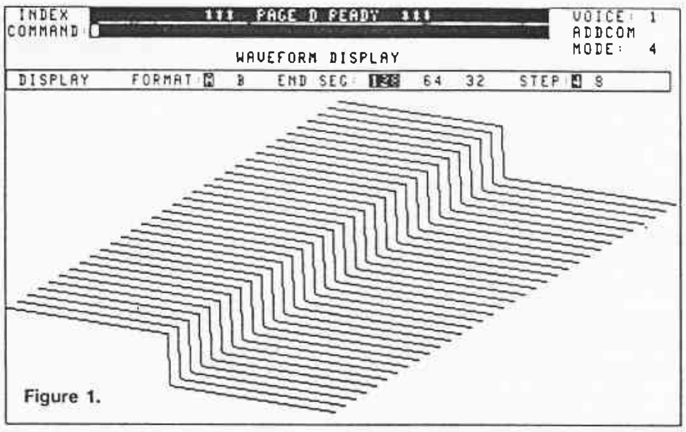
In which we take a look at the difference between linear and logarithmic conversion, and incidentally end up creating a sound using the Fairlight's synthesis facilities. *Jim Grant*

Most people involved in the modern music industry are already aware of the Fairlight's incredible potential as a music production tool. These days, you switch on the television, radio or record player in the almost certain knowledge that a CMI will make its presence felt somewhere along the line, and when you consider just how useful its specification is to studio engineers and producers, that's hardly surprising. Even in 1985, there aren't many machines capable of spreading six octaves of sampled sound across the keyboard, and manipulating that sound within user sequences to the nth degree of precision.

But if you're fortunate enough to sit in front of a CMI for any length of time *without* any production deadlines to meet, you'll soon discover that its creative power lies as much with sound synthesis as it does with music production *per se*. Pages 4 and 5 are good examples of this in that they offer the fairly standard synthesis tools of harmonic sliders and profiles, but Page 6, which we introduced last month, is something of a software oddity, since it allows control over the whole waveform – from a single byte to macro type commands such as `GAIN`, `MIX` and `MERGE`.

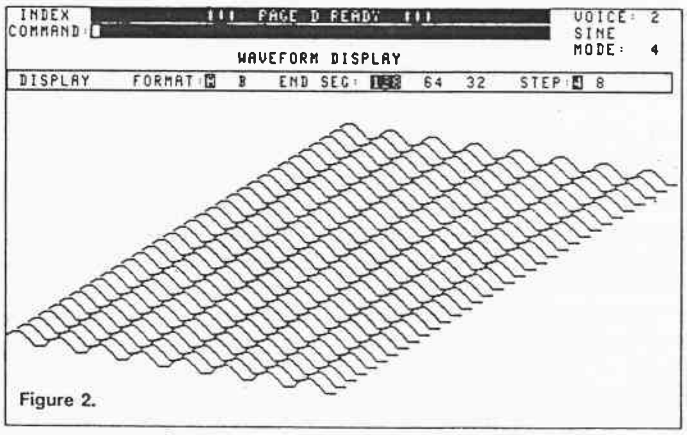
Now, for any command or process to be really useful in the field of sound synthesis, it must be responsible for some radical change in the sound structure that's both intuitive and easily understandable. For example, the VCF of an analogue synth changes the sound a great deal, and can be simply explained and understood in terms of the attenuation of harmonics. FM synthesis, on the other hand, also results in vast timbral differences, but comprehension of the processes involved (and their possible results) is a lot more difficult. That's why actually arriving at a pre-specified sound on something like a Yamaha DX7 requires so much in the way of practice and patience – and why so many musicians prefer programming analogue synths, even if the ultimate sonic potential isn't as great.

Fortunately, the Fairlight's internal configuration side-steps most of these operational problems, and a good example of how this is done is the `ADD` command. This takes a choice of segments from one loaded voice and adds them directly into the same segments of the currently selected voice, scaling the amplitude to avoid clipping if necessary. If you were to `ADD` all the segments into another, playing the keyboard would result in both sounds being heard together but only using one voice. Figure 1 shows a square wave and Figure 2 a sinewave both resident in different channels of the CMI: the result of `ADD`ing them together is shown in Figure 3. The addition's proportion can be varied by using the `GAIN` command prior to the action, or by repeatedly `ADD`ing one voice to another to increase its amplitude relative to the composite sound.



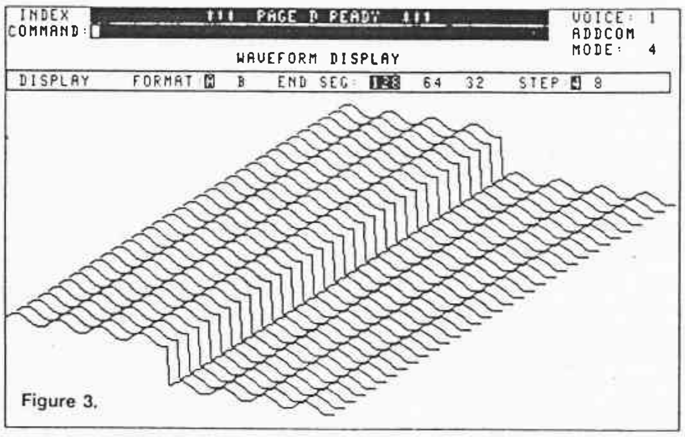
INDEX ‡‡‡ PAGE D READY ‡‡‡ VOICE: 1
COMMAND: ADDCOM
MODE: 4
WAVEFORM DISPLAY
DISPLAY FORMAT B END SEG 128 64 32 STEP 8

Figure 1.



INDEX ‡‡‡ PAGE D READY ‡‡‡ VOICE: 2
COMMAND: SINE
MODE: 4
WAVEFORM DISPLAY
DISPLAY FORMAT B END SEG 128 64 32 STEP 8

Figure 2.



INDEX ‡‡‡ PAGE D READY ‡‡‡ VOICE: 1
COMMAND: ADDCOM
MODE: 4
WAVEFORM DISPLAY
DISPLAY FORMAT B END SEG 128 64 32 STEP 8

Figure 3.

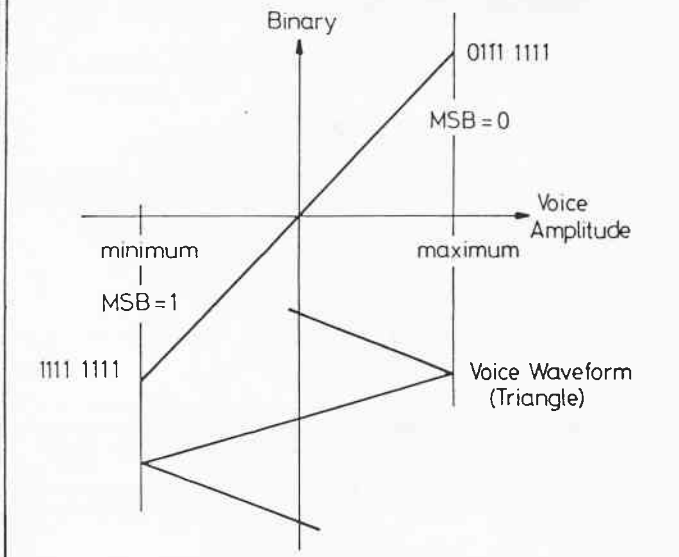


Figure 4. Linear conversion.

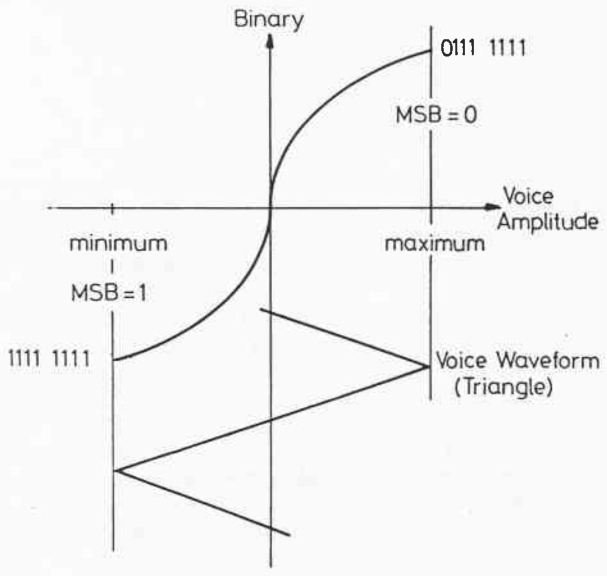
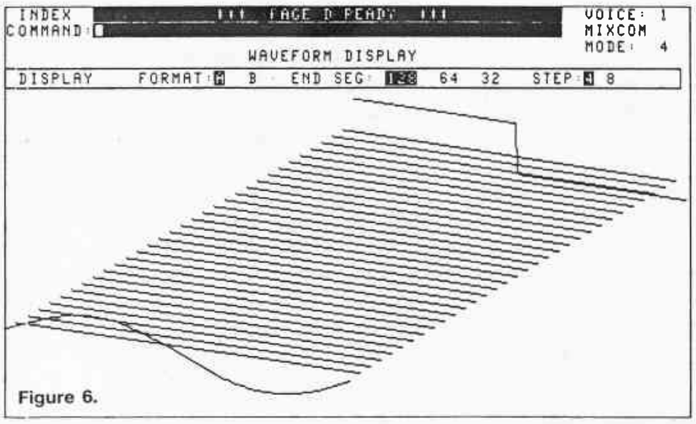
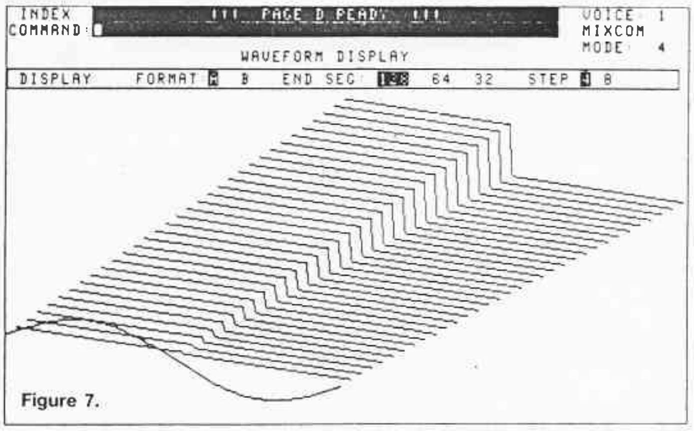


Figure 5. Non-linear (logarithmic) conversion.



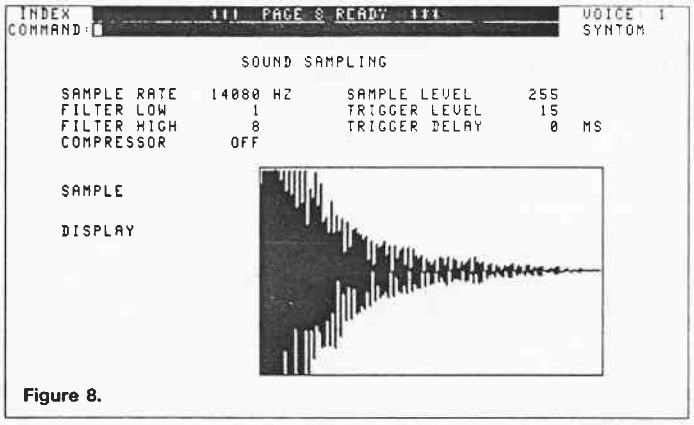
```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     MIXCOM
                 WAVEFORM DISPLAY            MODE:  4
DISPLAY    FORMAT: A  B   END SEC: 128  64  32   STEP: 4  8
```

Figure 6.



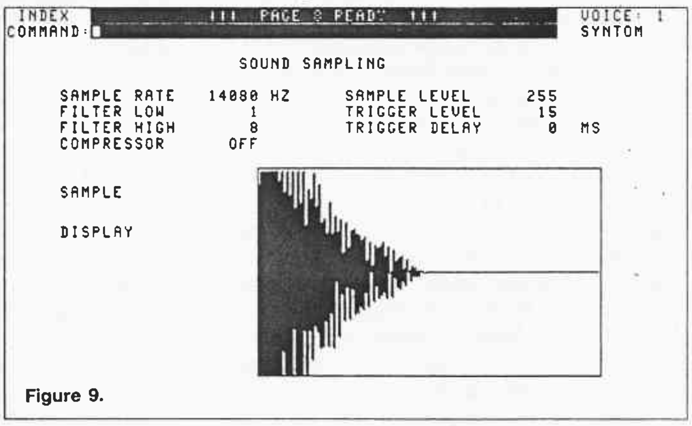
```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     MIXCOM
                 WAVEFORM DISPLAY            MODE:  4
DISPLAY    FORMAT: A  B   END SEC: 128  64  32   STEP: 4  8
```

Figure 7.



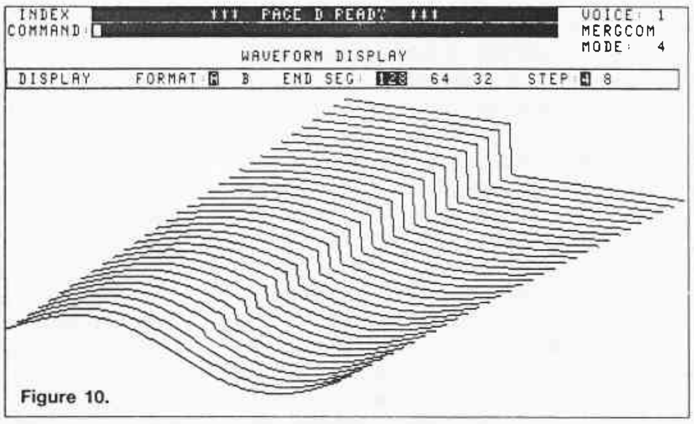
```
INDEX          *** PAGE 8 READY ***          VOICE: 1
COMMAND:                                     SYNTOM

                  SOUND SAMPLING

SAMPLE RATE   14080 HZ     SAMPLE LEVEL     255
FILTER LOW        1        TRIGGER LEVEL     15
FILTER HIGH       8        TRIGGER DELAY      0  MS
COMPRESSOR      OFF

SAMPLE

DISPLAY
```

Figure 8.



```
INDEX          *** PAGE 8 READY ***          VOICE: 1
COMMAND:                                     SYNTOM

                  SOUND SAMPLING

SAMPLE RATE   14080 HZ     SAMPLE LEVEL     255
FILTER LOW        1        TRIGGER LEVEL     15
FILTER HIGH       8        TRIGGER DELAY      0  MS
COMPRESSOR      OFF

SAMPLE

DISPLAY
```

Figure 9.

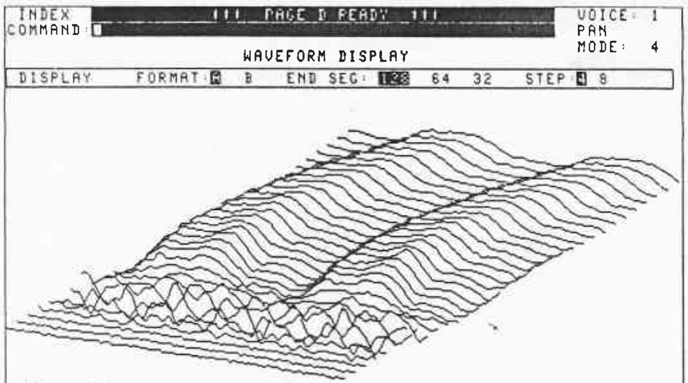


```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     MERGCOM
                 WAVEFORM DISPLAY            MODE:  4
DISPLAY    FORMAT: A  B   END SEC: 128  64  32   STEP: 4  8
```

Figure 10.

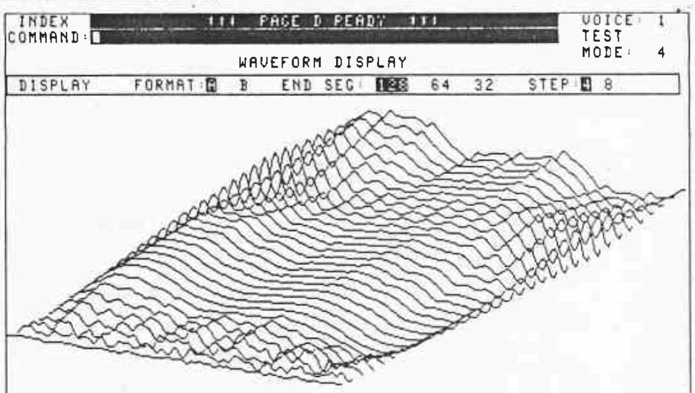
Linearity

Now seems as good a time as any to explain something we've mentioned many times in the past but haven't really discussed in any detail, namely the difference between linear and non-linear voice data. As you may remember, a waveform is stored in 16K of RAM, in which each byte has a binary value that corresponds directly to the amplitude of the waveform at that point. A byte consists of eight bits, and considering all the possible combinations of these results in the amplitude of the sound at any point being limited to one of 256 levels. The term 'linear' refers to the relationship between the actual amplitude and the value of the binary number used to represent it. If all this sounds a bit on the technical side (and it ought to), have a quick glance at Figure 4. This shows that zero amplitude corresponds to binary 0, while maximum negative excursion is ▷



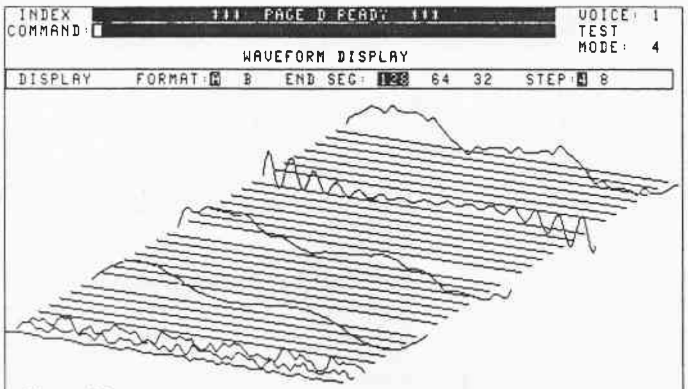
```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     PAN
                                             MODE:  4
               WAVEFORM DISPLAY
DISPLAY   FORMAT: A  B   END SEG: 128  64  32   STEP: 4  8
```

Figure 11.



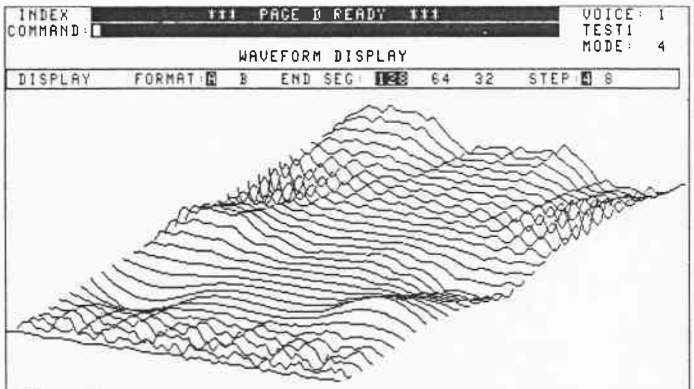
```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     TEST
                                             MODE:  4
               WAVEFORM DISPLAY
DISPLAY   FORMAT: A  B   END SEG: 128  64  32   STEP: 4  8
```

Figure 13.



```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     TEST
                                             MODE:  4
               WAVEFORM DISPLAY
DISPLAY   FORMAT: A  B   END SEG: 128  64  32   STEP: 4  8
```

Figure 12.



```
INDEX          *** PAGE D READY ***          VOICE: 1
COMMAND:                                     TEST1
                                             MODE:  4
               WAVEFORM DISPLAY
DISPLAY   FORMAT: A  B   END SEG: 128  64  32   STEP: 4  8
```

Figure 14.

represented by 1111 1111, or 255, and the maximum positive value is held as 0111 1111, or 127. Still confused? Well, the value of the Most Significant Bit holds the key. If that value is 0, the waveform is positive, while a 1 gives negative excursions. Anything else in between is in simple proportion. This form of representation results in a ratio between the smallest and largest signal that can be handled (or in other words, dynamic range) of about 48dB. You might consider that to be not a particularly impressive figure, since it means that at low signal amplitudes, the sound is more or less surrounded by hiss.

The fact is, storage (in one form or another) of low-amplitude sounds is a perennial engineering problem, to which the most common solution is some sort of noise reduction system such as Dolby. In the digital world, and as a direct result of research into digital telephony, a different solution is to use more binary bits of the byte to represent low-signal levels than you use for the high ones. This is shown in Figure 5, in which the lower values of the triangle wave use up more of the binary bits than a corresponding increase at large triangle amplitudes. The binary data is now no longer linear – it's logarithmic. For the waveform to be recovered, the data must be passed through a DAC that has a curve bent the opposite way to 'straighten out' the sound. I know all this sounds more than a little involved, but it does bring the magic dynamic range ratio up to about 72dB, which is at least respectable. Only catch is, the process only achieves this with a corresponding increase of quantisation noise at larger signal levels, though this is masked by the volume of the signal itself.

You might be familiar with this conversion process under its commonly used name of companding, and it's a system used by many hardware manufacturers including Linn and E-mu.

So if it's so good, why doesn't the CMI use it? Simple. Remember your school days when you added log numbers to multiply? Well, this is what would happen if the ADD command was used with sounds held in the form generated by Figure 5: instead of adding the sounds together to produce a mix, we'd get the product, and end up with VCA-type effects at low frequencies and strange sidebands at higher ones. Which isn't, all things considered, a particularly desirable state of affairs.

Mixing

Anyway, enough of the lecture and back to the Fairlight. The MIX command can also drastically alter the waveform RAM. Essentially, it generates a crossfade between two specified segments which must not be adjacent, ie. there must be at least one segment in between. The waveform memory of each segment between the start and end points contains a proportion of the existing waveform in that segment and that of the end segment: this is best illustrated by examining Figure 6 and Figure 7 for before and after views. Remember that the new contents of each segment is a mix between where you are in the waveform and the destination segment. Thus from Segment 2 onwards, the waveform simply fades up to a square wave. MIX is most commonly used to add a clean fadeout to a sound that decays to noise or doesn't decay properly in 128 segments.

Have a look at the percussion sound sampled using Page 8 and shown in Figure 8. It's pretty clear that the sample ends in a dither of noise. Now, suppose we needed nothing more than a short percussive strike, and that only the beginning of the sound was of any interest to us. A quick solution would be to turn to Page 6 and ZERO, say, Segments 64 to 128, halve the sound, and then MIX from Segment 45 to 64. Looking at Figure 9 shows the result – a sound that dies away evenly to a noise-free end, much to the relief of all concerned.

MERGE is fairly similar to MIX – with one fundamental difference. Again, a form of crossfade is generated between start and end segments, but this time, the previous contents of intermediate segments don't figure in the result. Quite simply, the segments in between contain a decreasing proportion of the start segment and an increasing proportion of the end segment. Figures 6 and 10 (*oh yes, very logical* – ***Ed***) reveal all. The MIX and MERGE commands are tremendously powerful for splicing together sounds of differing origins and producing an even fade from, say, a violin bow attack to a sung 'ahh'.

Creating a Sound

So, now that we've discussed most of the commands available, let's try to create a sound using everything *except* the Fairlight's Page 8 sampling facility. The question is: am I allowed to use Page 2 and pull a sound off-disk to work with? Well, I've decided I'll have to cheat a bit because I already have a thoroughly marvellous sound called PAN.VC, which attacks with the characteristic breath chiff of pan-pipes.

First off, we configure Page 3 to generate two voices, one with an NPHONY of 7 in Register A to be played on the keyboard, the other monophonic in Register B as a scratchpad voice. Using Page 2, we load Register B with PAN.VC, which can be seen in Figure 11. The breath chiff is clearly visible at the beginning of the sound, but unfortunately, the sampling started a fraction too soon, and the waveform has a few initial segments of low-level rubbish – nothing to do with *Electronic Soundmaker*, you understand. The cure is to rotate the voice left to bring the start of the sound proper coincident with the start of the RAM.

The next step is to flick to Voice 1 in Register A and ZERO it. Using a new command, TRANSFER, the first few segments from PAN.VC can be copied to the blank voice currently selected. Stabbing the keyboard at this juncture reveals that all is well, so the next thing to do is to work on the body of the sound itself.

Before any sound can really make the grade as far as aesthetics are concerned, it must have plenty of timbral and amplitude movement within it. A good way of producing harmonically-rich waveforms is to use Page 5 and create a few segments spread across those unused by the chiff. Figure 12 gives the general idea – note that the created waveforms are all different. So what about the segments in between? Well, this is where MERGE comes in handy, filling in the ZEROed segments, and using the segments created on Page 5 as the start and end points.

OK, so far it sounds quite interesting timbrally (and looks it too, as Figure 13 shows) but it's still in need of some amplitude variation. An easy way to achieve this is to invert a couple of segments (numbers 32 and 96, say) and MIX from segments 1 to 32, 64 to 32, 64 to 96 and 128 to 96, using Page 6. If you look closely at the differences between Figures 13 and 14, you shouldn't have much difficulty identifying the variation in amplitude, especially in the sound's first quarter.

All that remains is to insert loop points on Page 7 or Page 4, and adjust the attack and damping on Page 7. ■

I'VE JUST BOUGHT THIS NEW 'MEGA-AUTO-COMPOSER' SOFTWARE - ONLY COST ME £500!
I THINK THEY SAW YOU COMING, MATE!
MEGA COMP
IT CONTAINS ALL OF THE TECHNIQUES AND STYLES OF THE WORLDS GREATEST COMPOSERS! 'CAN TURN THE MOST BORING AND MEDIOCRE SONG INTO AN UNFORGETTABLE CLASSIC!' IT SAYS HERE!
RLITE CMI
CLICK!
NOW I CAN COMPOSE LIKE MOZART AND BEETHOVEN I'LL BE MEGA-FAMOUS IN NO TIME! LET'S SEE WHAT THE PROGRAM DOES WITH THAT PIECE OF AMBIENT MUSIC I DID LAST NIGHT!
NELLY THE ELEPHANT PACKED HER TRUNK A
SAID GOODBYE THE CIRCUS OFF SHE
TRUMPETY
GARY McLEOD