Applied Acoustics **25** (1988) 243–268

A Microcomputer-Based Audio Digital Delay System

B. T. G. Tan & T. H. Ong

Department of Physics, National University of Singapore,
Singapore 0511

(Received 7 March 1988; accepted 21 April 1988)

ABSTRACT

An audio digital time delay system based on a bus-oriented microcomputer, the Apple II, has been designed and tested. Analog interface boards for the microcomputer bus were designed and constructed to interface the incoming and outgoing audio signals to the microcomputer. To speed up the processing of the audio samples, coprocessor boards using the 6809 and 68008 microprocessors were used. The delay system is capable of changing the required time delay dynamically under software control.

INTRODUCTION

Audio frequency delay units are widely used in audio and acoustical engineering as well as in basic research on psycho-acoustics. They are an essential component of all but the most simple sound reinforcement systems which aim to deliver intelligible speech in large halls and auditoriums. They are also being increasingly used in sound reproduction equipment to enhance domestic listening rooms by simulating the reverberation characteristics of large concert halls.

Before the advent of large-scale integrated (LSI) circuits, audio delay units were largely analog in nature. A common method then of obtaining an audio delay was to employ a closed-loop tape unit with a record and playback head closely spaced together. The main drawback of such units was the difficulty of obtaining very short delay times and the invevitable wear and tear on the tape and tape heads. Another electro-mechanical method was the use of a coiled spring to act as an electro-acoustical delay line. The quality of the delayed signal was usually only good enough for reverberation systems which did not require a high quality delayed signal.

The conversion of the analog signal to a digital form would have eliminated most of the problems associated with electro-mechanical analog delay units. However, this was not practicable with discrete electronic device technology, as the digital storage capacities required for audio delays of even a few milliseconds were far too large for discrete electronics. One family of integrated circuit devices, variously known as charge-coupled or bucket-brigade devices, represented an intermediate step in the progress to fully digital audio delay units. Such devices were able to store sampled audio signals in analog form efficiently enough to enable highly effective audio delay units to be constructed. These units were however, soon made obsolete by the arrival of fully digital audio delay units, made possible by rapid advances in semiconductor technology. Fully digital audio delay units are now in the midst of a transition from 12 bits to 16 bits which is driven by the decreasing costs both of 16-bit analog-to-digital (A/D) converters as well as random-access memory (RAM).

BASIC PRINCIPLES OF AUDIO DIGITAL DELAY UNITS

The main objective of an audio delay unit is to store an input audio signal and output the signal after it has been stored for a required length of time. The whole process is done in real time, with the input signal being continuously stored and output as it changes in accordance with the audio waveform. The amount of audio signal to be stored depends on the length of the delay time; a longer delay time means that the storage buffer for the signal needs to be larger.

Thus the basic delay unit must consist of three elements: an input unit which captures the audio signal, a storage unit which stores the signal, and an output unit which sends the delayed signal out again. The process is dynamic, with the audio signal progressing through the input unit, storage and output unit continuously as the audio waveform changes. An analog tape-based delay unit clearly shows these three elements, with the record head as the input unit, the playback head as the output unit, and the length of tape in between as the storage unit, which is continuously moved by the tape recorder transport system from input to output head.

In a digital delay unit, the stored signal is not in analog form, but is converted into digital form. This implies that two operations are performed on the audio waveform:

1. The waveform is first 'sampled' i.e. its magnitude at regularly-spaced time intervals is stored. The sampling frequency i.e. the frequency at

which these samples are taken is dependent on the upper frequency limit of the signal. By the Nyquist criterion, for an upper frequency limit of f Hz, the sampling frequency should be at least equal to $2f$.

2. The analog samples are then 'digitized' i.e. each sample value is converted to a binary digital number. The length of the binary numbers determines the accuracy with which the values are preserved. This accuracy in turn determines the signal-to-noise ratio of the audio signal when it is reconverted to its analog form. As a rule of thumb, 6 dB is obtained for each bit, e.g. 12bits will yield a 72 dB S/N ratio. This can be further enhanced by more subtle methods of digital conversion.

The great advantage of digital processing is that the accuracy of the analog waveform, once fixed by the choice of sampling rate and bit length, is perfectly preserved. In addition, the time delay lengths can also be accurately set. Audio digital delay units as used in the audio industry are usually stand-alone or modular units. The bit length is usually 12- or 16-bits depending on the price of the unit. In order for such units to be of general use in typical applications such as sound reinforcement systems, the time delays available range from a few to several hundred milliseconds. The variation in delay time is usually controlled by a front-panel control. Some sophisticated units provide external control of the time delay, usually by the application of an external analog voltage. Such methods are adequate for real-time control in musical performance, which does not require extremely precise control of the delay length. Time delay units meant for sound reinforcement systems are usually preset to some fixed value and hence seldom provide a means of rapid real-time control of the delay time.

In the course of experiments on psycho-acoustics, a need was felt for a digital time delay system whose delay time could be instantaneously varied under computer control. The availability of inexpensive microcomputer systems made it worthwhile to investigate the possibility of designing a time delay system based on such a microcomputer system. The time delay system which is described in this paper was designed specifically for use in a series of psycho-acoustic experiments to determine the effect of time delays in stereophonic sound reproduction. It may have more general applicability in any work in which a rapidly variable software-controllable delay length is required.

DESIGN OF THE DELAY SYSTEM

It was decided at the outset that the use of a microcomputer with a bus-oriented structure would provide the greatest flexibility and economy. The most-well-known of these bus-oriented personal computer systems are the Apple II family and the IBM PC family (and other computers compatible with them). Microcomputer systems are generally organized around a microprocessor chip which performs the function of a central processing unit (CPU) as in a conventional mini or mainframe computer. The microprocessor communicates with its memory and its input/output (I/O) channels through a set of circuit lines known as a bus. In bus-oriented computers, this bus is directly accessible to the user, thus simplifying the connection of external devices. Ths bus is usually available in the form of slots on the main circuit-board (or motherboard) into which smaller boards carrying the interfaces to the external devices can be inserted. The additional hardware for the audio delay system could thus be designed in the form of one or more interface boards for insertion into the bus slots.

The microcomputer chosen for the system was the Apple II. However, the design principles apply to any other bus-oriented microcomputer such as the IBM PC. Basically, any digital delay system requires an A/D converter to convert the input audio signal to digital form, random-access memory (RAM) to store the digital samples, and a D/A converter to convert the delayed digital signals back to analog form. The whole process of inputting, storing and outputting the digital samples is controlled by a microprocessor. In our system, the microprocessor and RAM memory are supplied by the Apple II. The A/D and D/A converters are added to the system in the form of an interface board which fits into a slot of the Apple II. In this way, the additional costs to convert the Apple II into an audio delay system are much lower than that of a stand-alone delay system. In addition, the Apple II's microprocessor can exercise greater control over the process than is possible for a stand-alone unit.

THE AUDIO INPUT/OUTPUT BOARDS

An interface board to input and output the audio analog signals was designed and constructed for the Apple II bus. It was decided to use at least 12-bit digitization as 8 bits would have yielded a signal-to-noise ratio of only about 48 dB, which is too low for high quality audio applications. Sixteen bits would have been ideal, since the resultant 96 dB S/N ratio would have been equivalent to digital compact disc standards. However, at the time of the experiment, 16-bit A/D converters were prohibitively expensive, while 12-bit A/D and D/A converters were easily available and reasonably priced. The S/N ratio obtained with 12 bits, 72 dB, was also acceptable for all but the most demanding audio applications.

The main chips on this audio input/output board were the AD574 12-bit

A/D converter and the AD565 12-bit D/A converter, both from Analog Devices.[1] These two chips are easily obtainable and are virtually industry standards. Each of these chips was interfaced to the Apple II bus via a 6821 PIA (Peripheral Interface Adapter) chip. Each 6821 PIA occupies four memory addresses in the Apple II's memory map, and greatly facilitates the control of the AD574 and AD565. All communication with the AD574 and AD565 is done through the four addresses of the PIA.

The audio input/output board provides one audio input channel via the AD574 and one audio output channel via the AD565, which is adequate for many audio applications. However, in applications such as multi-channel sound reinforcement systems, several outputs with different time delays have

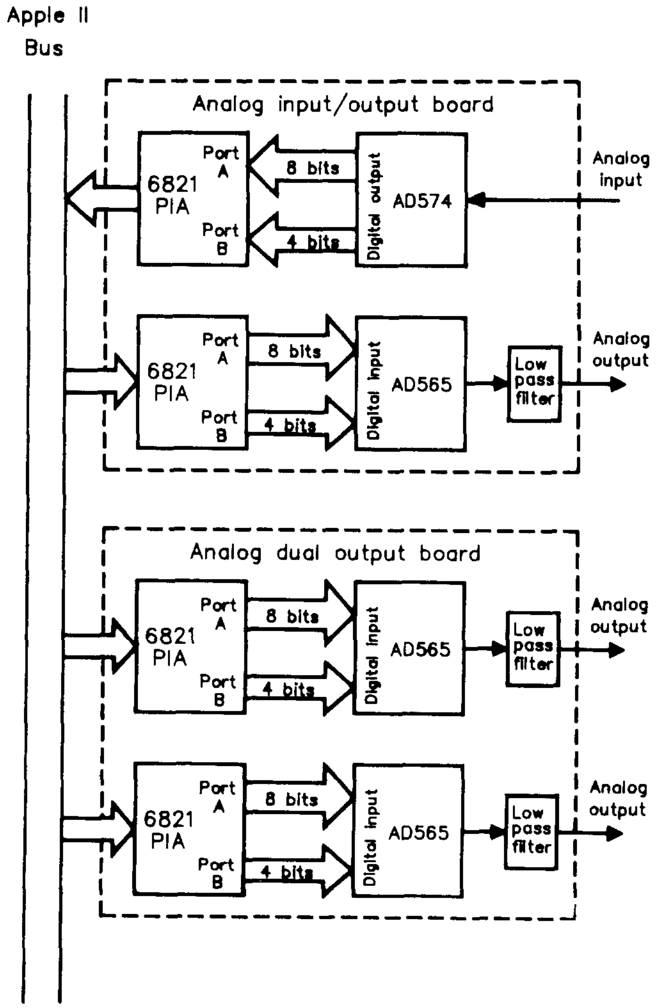

Fig. 1. Block diagrams of the audio input/output board and audio dual output board.

to be provided for. Thus a second audio interface board was designed and constructed to provide a further two audio output channels utilising two AD565 12-bit D/A converters. For this audio dual output board, each of the two AD565 chips was interfaced to the bus via a 6821 PIA chip in the same way as the AD565 on the other board. The block diagrams of these two boards are given in Fig. 1.

The decision to employ 12-bit A/D conversion automatically imposed a severe strain on the 8-bit 6502 microprocessor used in the Apple II. The memory of the Apple II is organized into 65 536 8-bit locations.[2] The 6502 can transfer 8-bit data to and from each memory location. Each memory location in the Apple II has a unique address, which itself is 16 bits long. Thus there are a total of 65 536 or 64 × 1024 addresses ranging from $0000 (where $ denotes a hexadecimal number; 4 bits are equal to one hex digit) to $FFFF or decimal 65 535.

The 12-bit samples produced by the AD574 will thus have to be stored in two adjacent 8-bit locations in the Apple II RAM. The 6502 processes 8 bits at a time, so that the storage of one 12-bit sample requires two operations. Similarly, the outputting of one 12-bit delayed sample through the AD565 will require two operations by the 6502. This will slow down the throughput of the 12-bit samples through the memory, and thus greatly affect the sampling rate which can be imposed on the audio signal. Hence another method of processing the 12-bit samples was investigated.

16-BIT COPROCESSORS

Though the Apple II uses the 6502 as its main microprocessor, it is possible to add other microprocessors in order to increase the processing speed of the Apple II. In such a situation, the additional microprocessor co-exists with the 6502 as a coprocessor. The 6502 is usually made to handle the background I/O tasks, such as reading the keyboard and maintaining the screen display, while the coprocessor gets on with the actual processing of the data. There are several coprocessor boards available which interface to the Apple II through the bus slots. In order to increase the processing speed to obtain the required sampling frequency, two 16-bit coprocessor boards were used separately, one employing the 6809 microprocessor, a hybrid 8/16-bit microprocessor, and the other employing the 68008, a powerful 16/32-bit microprocessor.

Actually, the criteria as to whether a microprocessor (MPU) is 8 or 16-bit are not very clearly defined. It would appear that the ability to handle 16-bit data would be the main criterion; however, even the 6502 can handle 16-bit data in a limited way. The 6809 can internally handle 16-bit data and has special 16-bit instructions, though like the 6502, it only has an 8-bit data bus for data transfer to and from the outside world. The 68008 is a member of the powerful 68000 microprocessor family, being able to handle 16- and 32-bit data internally, but also having an 8-bit data bus. While both the 6809 and 68008 coprocessor boards are somewhat handicapped by having to interface with the slow Apple II 8-bit data bus, the superior processing speeds of the 6809 and the 68008 as compared with the 6502 made their utilisation for the digital time delay system worthwhile.

PROCESSING OF THE DIGITAL DELAY SAMPLES

As described above, the audio time delay is obtained by sending the audio input into an A/D converter where it is sampled and converted into 12-bit data. This stream of 12-bit data is channelled through the RAM and exits through the D/A converter where it becomes the delayed audio signal. It may therefore seem that the best way to perform this would be for the input 12-bit data to enter at a fixed input address in the memory, be shifted along one pair of addresses each time a new sample came in, and eventually exit further along at a fixed output address through the D/A converter. This is exactly analogous to the closed-loop tape delay unit, where the audio signal enters via the fixed record head and is carried along by the moving tape to exit at the fixed playback head.

However, in the microcomputer system, an analogous approach would require the shifting of all the 12-bit data between the A/D and D/A converters each time a new sample was input. This is actually how charge-coupled or bucket-brigade delay units perform; the analog voltages stored on each device are actually shifted along as each new analog sample is input. In the fully digital case, an analogous approach would require a shift register. In a shift register, the registers are arranged sequentially and all the data can easily be simultaneously shifted in one direction. (In actual fact, a bank of shift registers is required, one for each bit of the digital data.)

This approach is cumbersome in the case of the microcomputer system. The RAM in a microcomputer is *not* functionally like a shift register, and so the microprocessor would have to be used to shift all the data between the input and output addresses for each new input sample. The number of data samples to be shifted might in practice be very large, since in one second several thousand samples would have to be stored for a high quality audio signal of several kHz. This would thus take up far too much processing time and hence reduce the bandwidth of the delay system to unacceptably low frequencies. Thus the approach adopted in the present experiment was to keep the data in the RAM in fixed addresses and 'shift' the A/D and D/A converters along the RAM addresses.

The best way to understand this approach is to use the tape delay again as an analogy. In the normal method, the two tape heads are fixed while the tape moves from the record to the playback head. The analogous situation to this second approach would for the tape to be stationary and the two tape heads to be rotating around the closed tape loop. The relative motion between the tape heads and the tape is still maintained so that the record head impresses a continuous stream of audio data on the tape, while the playback head picks up the delayed signal previously recorded by the record head, thus satisfying the condition of continuous input and output.

Though this method is less practical in the case of the tape loop, it is much more feasible in the case of the microcomputer system. The audio samples stay in the same addresses during the time that they are in the RAM, hence avoiding the need to shift them. The A/D converter and D/A converter are instead made to store and retrieve the audio samples from a continuously changing sequence of addresses. Since this entails shifting only two addresses (those of the two converters) instead of the large number of data samples in between converters, the processing time is considerably reduced.

THE MEMORY LOOP

In the tape system, the audio signals are stored in a tape loop. The equivalent of the tape loop in the microcomputer is a fixed portion of its memory. The length of memory required will depend on the maximum delay length required as well as the sampling rate. If an audio signal has a bandwidth of 8 kHz, the sampling rate should be at least 16 kHz. Thus for every second, 16 000 12-bit numbers (commonly called words) are stored. Since each 12-bit word occupies two 8-bit addresses, 32 000 memory addresses are required. For sound reinforcement work, 250 millisecond delays are sufficient, so that only 8000 addresses are required.

An 8 K segment of memory (where K now stands for 1024 as is usual in digital computers) might run, for example, from the address $6000 to $7FFF. Though this is physically a linear segment, we have to effectively make it behave like a loop. The A/D and D/A converters have to step through the entire range of addresses of this memory segment. In order to make the segment look like an endless loop to the A/D and D/A converters, we have to reset the address for each converter, when it reaches the last address of the segment at $7FFF, back to the first address at $6000. This can easily be done by software. The segment then functionally becomes an endless loop, as far as the converters are concerned.

DELAY TIME

The actual delay time depends on two factors: the length of the memory segment and the relative addresses of the memory locations accessed by the A/D and D/A converters. As the audio samples are read into the memory by the A/D converter, each new sample is stored in the memory two addresses higher than the previous sample. For example, if the memory segment \$6000 to \$7FFF is used, let us consider a sample which is stored at \$6000 and \$6001, since a 12-bit sample requires two 8-bit locations. The next sample will be stored at \$6002 and \$6003 and so on. When the A/D converter reaches \$7FFE and \$7FFF, the software will direct the A/D converter to go back to \$6000, thus making an endless loop and storing a new set of samples in the memory.

It can be seen that when the A/D converter has stored an audio sample at

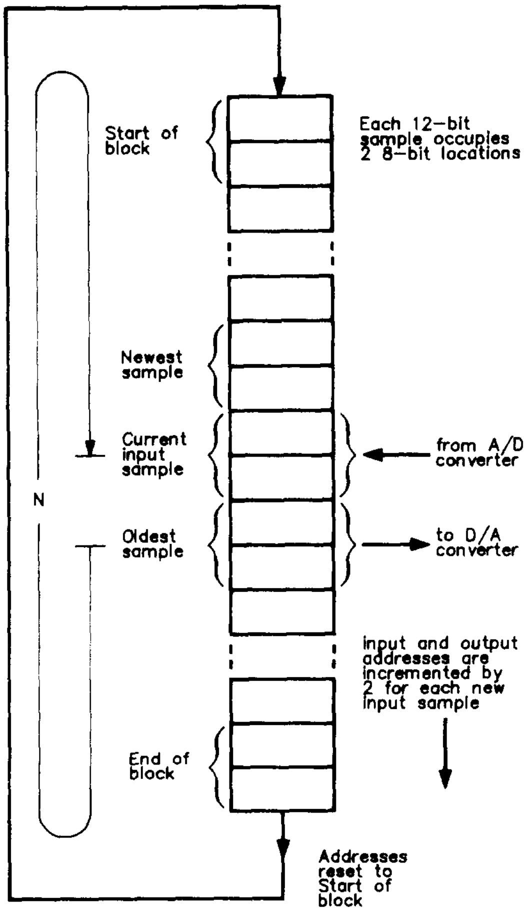
Start of block
Each 12-bit sample occupies 2 8-bit locations
Newest sample
Current input sample
from A/D converter
N
Oldest sample
to D/A converter
Input and output addresses are incremented by 2 for each new input sample
End of block
Addresses reset to Start of block

Fig. 2. How the memory block in RAM forms an endless loop.

any pair of addresses A and A + 1, the address A + 2 and A + 3 will contain the 'oldest' sample stored previously. Hence if the D/A converter extracts the outgoing samples from A + 2 and A + 3, these samples would have been delayed for the full length of time taken by the converters to go round the whole loop. On the other hand, the sample at addresses A − 2 and A − 1 is the most recent sample stored previously compared to that at A and A + 1, and if the D/A converter extracts this sample it will provide the shortest possible delay time.

Thus, by varying the length of the memory loop, i.e. the length of the memory block used for the loop, we can vary the maximum delay length that can be obtained. For a fixed memory loop length, we can change the delay time merely by changing the relative positions of the A/D and D/A converters as they traverse around the memory loop. If we think of the A/D converter as following the D/A converter, the more closely it follows, the longer will be the delay time. For the maximum length, the A/D converter should be just one address behind the D/A converter, while for the shortest possible delay, their positions should be reversed. Figure 2 shows how the memory block is arranged to form a loop, with the D/A converter retrieving samples from the addresses just ahead of the A/D converter, thus giving the longest possible delay time for the memory block.

THE DELAY ROUTINE

The assembly language program to generate the time delay is basically very simple. We first consider a basic program for a single-output time delay. The purpose of the program is to store the incoming A/D digital samples in the memory block and retrieve the outgoing delayed samples from the block in an ordered and increasing sequence of addresses. In the main part of the program loop, one 12-bit sample is input from the A/D converter and then stored in a certain pair of 8-bit locations in the block. A 12-bit sample is then retrieved from another pair of 8-bit locations at an address which is at a fixed offset from that of the incoming sample. Each of the two addresses is incremented to point to the next pair of 8-bit locations in the block. When the end pair of addresses of the block is reached by either the A/D or the D/A converter, the addresses are reset to the start of the block.

Thus the time between the sampling of successive samples is determined by the time spent in one loop, which should be made as short as possible. For a 16 kHz audio bandwidth, a sampling rate of a least 32 K samples per second is necessary. If the 6502 were used to process the audio samples, the sampling rate would be severely limited by its processing speed. This is due not only to the 8-bit limitation of the 6502, but also to its inability to do indexed addressing over a range greater than 256 addresses. Indexed addressing allows the microprocessor to efficiently increment addresses over a memory block, such as is required in the time delay program. This indexed addressing limitation imposes a severe constraint on the length of the memory block which the 6502 can effectively handle, limiting it to 256 addresses.

A time-delay program for the 6502 using the present technique has been previously reported.[3] The loop required 45 Apple II clock cycles and hence, at the Apple II clock frequency of 1·023 MHz, took 43·99 microseconds. With this loop delay, the sampling rate was $2{\cdot}273 \times 10^4$ samples per second, giving an audio bandwidth of 11·37 kHz. Though this is acceptable, the block length of 256 addresses limits the longest possible delay time to 43·99 microseconds × 128 which equals 5·63 milliseconds. (The loop delay is multiplied by 128 and not 256 because each 12-bit sample requires two addresses.) This is quite inadequate for most audio work. Longer delays would require longer blocks of memory and would result in much lower bandwidths if the 6502 were used, as slower methods of incrementing the addresses would have to be used.

THE 6809 MICROPROCESSOR

The 6809 microprocessor, being able to handle 16-bit data transfers more easily, is better suited to processing the 12-bit audio samples. In addition, the 6809 has superior indexed addressing modes compared to the 6502 and is not restricted to a 256-address block of memory for indexed addressing. A time-delay program was written for a 6809 coprocessor board to take advantage of its features. The 6809 coprocessor board was manufactured by Stellation Two, and is called ‘The Mill’.[4]

The 6809 MPU on the board is a true coprocessor in that it can run simultaneously with the 6502. Both MPUs are regulated by the Apple II clock. The 6809 becomes active only when the coprocessor board is given an explicit instruction to activate it. The 6502 can still function alongside the 6809, and actually executes its instructions during the clock cycles when the 6809 is idle. Hence it is possible to have the 6502 take care of some subsidiary tasks while the 6809 performs the main task of inputting, storing and outputting the delay samples. In addition, having the 6502 as a simultaneous coprocessor allows us to use it to change the delay parameters while the 6809 processes the audio samples, so that delay times can be instantaneously altered while the program is executing.

The time delay program for the 6809 utilises the audio input/output board with the AD574 A/D converter and the AD565 D/A converter hence providing for one input audio channel and one output audio channel. The addresses of the two PIAs corresponding to the two analog chips depend on the Apple II slot in which the board is plugged as follows:

PIA1 (AD574) $C0x0 – $C0x3 or $C0x8 – $C0xB
PIA2 (AD565) $C0x4 – $C0x7 or $C0xC – $C0xF

where x is a hexadecimal number equal to the slot number (0 to 7) plus 8. For example, if the board is in slot 4, x would be equal to $C.

The memory block in which the audio samples are stored begins at $3000. The address of the end of the block is not explicitly given in the program, but is itself stored in the addresses $00EE and $00EF. This makes it easier to change the end address, and hence the block size, as desired. For the present experiments, the end of the block was set at $6FFF, giving a block length of $4000.

Figure 3 shows the flowchart of the program. First the PIAs have to be 'initialized' so that they are properly set up to work with the AD574 and AD565. The X and Y index registers are used as input and output pointers respectively to store and retrieve the digital samples. The input pointer points to (i.e. contains the address of) the pair of locations which are to receive the incoming 12-bit sample from the A/D converter. The output pointer points to the locations from which the output samples are to be retrieved by the D/A converter. The X and Y index registers could, for example, be initialized to $3000 and $3002 respectively to give the maximum possible delay time for the given block length. Shorter delay times can be obtained either by increasing the difference between the X and Y index registers, or by decreasing the block size which is defined by the end-of-block address in the addresses $00EE and $00EF.

The digitized audio sample is then read in from the AD574 via its respective PIA. This sample is immediately stored in an address which is pointed to by the input pointer. The input pointer is then incremented so that it points to the next pair of locations, ready for the next input sample. The delayed output audio sample is then retrieved from a pair of locations pointed to by the output pointer and output through the AD565 via its respective PIA. The output pointer is also incremented so that it points to the next audio sample to be output.

The storage of the input sample and the retrieval of the delayed sample constitute the main body of the program loop. At the end of the loop, before the program returns to the beginning of the loop to process the next set of audio samples, it has to determine whether the end of the memory block has been reached. For example, if the memory block extends from $3000 to $6FFF, then the input and output pointers will each in due course exceed the last address in the block. At this stage, each pointer has to be steered back to

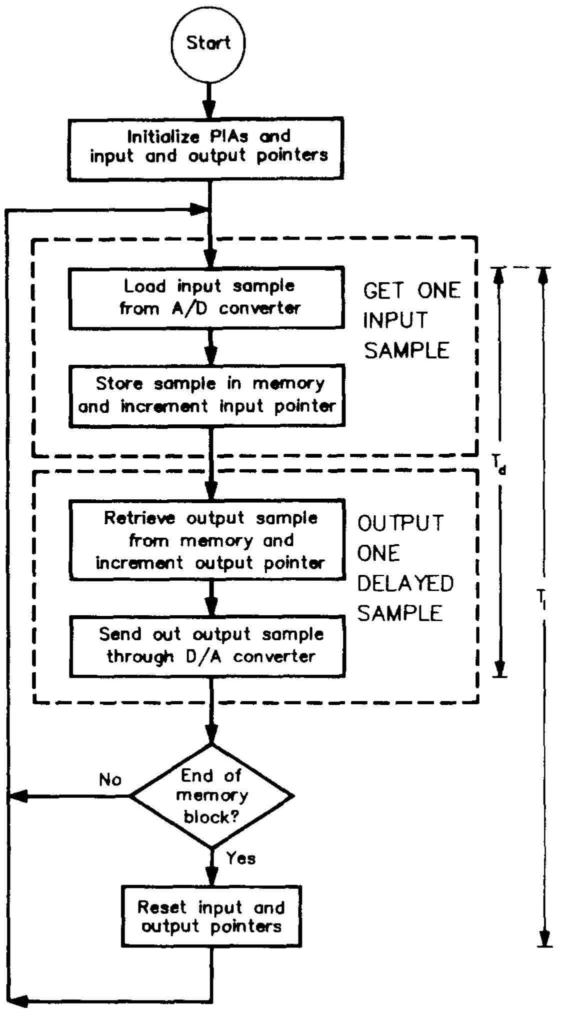
Start
Initialize PIAs and input and output pointers
Load input sample from A/D converter
Store sample in memory and increment input pointer
GET ONE INPUT SAMPLE
Retrieve output sample from memory and increment output pointer
Send out output sample through D/A converter
OUTPUT ONE DELAYED SAMPLE
T_d
T_l
End of memory block?
No
Yes
Reset input and output pointers

Fig. 3. Flowchart of single-output 6809 delay program.

the beginning of the block so that the block effectively forms an endless loop. Hence at the end of each program loop, a test must be performed to determine if each pointer has reached the end of the block. If it has, it is 'reset' to the beginning of the block.

There are two programming complications here which we will briefly mention. When each pointer reaches the block end, the program needs a little extra time to do the resetting. Now, this may cause the program loop to be a little longer than usual. Hence we have to ensure that the loop durations are precisely the same, whether the block end has been reached or not, otherwise every time the end of a block is reached, there will be a slight 'hiccup' which will slightly distort the audio waveform. The second complication arises because the input and output pointers do not reach the end of the block simultaneously. Thus each index register has to be tested and reset separately, thus complicating the end-of-loop 'housekeeping'. In the actual machine language programs, all end-of-loop housekeeping routines were optimised to take care of these problems.

CALCULATION OF TIME DELAY

The total time spent in one loop can be determined by adding up the time taken up by each instruction in the loop. Each successive sample is delayed with respect to the previous one by precisely the time spent in one loop. The time delay is the time spent in one loop T_l multiplied by the offset in terms of samples between the address into which the input samples are stored and the address from which the delayed output samples are retrieved. To this must be added the small time difference T_d between the instruction loading the input sample from AD574 and the instruction storing the output sample to the AD565. Both T_l and T_d are shown in Fig. 3. Hence the total delay time T is given by

$$T = (N * T_l)2 + T_d \tag{1}$$

where

T_l is the time spent in one loop
T_d is the time between the input and output instructions and
N is the offset between the addresses accessed by the AD574 and the AD565.

In order to set up the correct parameters in the time-delay program for the desired delay, the user would have to calculate N (which is shown in Fig. 2) as defined in eqn (1) above. The value of N thus obtained gives the offset between the address for the storage of the input samples and the retrieval of the output samples. If, for example, the input address is initially defined as \$3000 and the memory block ends at \$3FFF, then for $N = 1024$ or \$400, the output address should initially be set at \$3C00. If we consider the block to be an endless loop with \$3FFF joined back to \$3000 then we can see that \$3C00 is \$400 addresses behind \$3000.

Note that N is obtained by starting from the AD574, going sequentially backwards along the addresses in the memory block until the AD565 is reached, treating the block like an endless loop. The largest possible value for N is the length of the memory block, obtained when the AD574 is directly behind the AD565, as in Fig. 2.

For the 6809 program, the following values were obtained:

$$T_1 = 37 \text{ clock cycles or } 36{\cdot}17 \text{ microseconds}$$
$$T_d = 28 \text{ clock cycles or } 27{\cdot}37 \text{ microseconds}$$

Thus for a memory block which is 1024 addresses long, the longest possible delay time is 37·065 milliseconds. Longer blocks will give proportionately longer delays.

The lower limit of the sampling time depends on the speed with which the A/D and D/A converters can operate, which is in effect determined by the conversion speed of the A/D converter, since the D/A converter is much faster. The AD574 A/D converter, which is a successive approximation A/D converter, has a specified maximum conversion time of 35 microseconds. Hence a loop time of less than 35 microseconds is not desirable, since a complete A/D conversion in less than that time cannot be guaranteed. As it happens, the loop time of 36·17 microseconds is just above this lower limit.

The audio bandwith which can be handled depends on the sampling frequency which in turn depends on the time spent in one loop. For $T_1 = 36{\cdot}17$ microseconds, the sampling frequency is 27·64 thousand samples per second, and hence the audio bandwidth is 3·82 kHz, which is quite sufficient for many audio applications. The measured audio delay times and bandwidths obtained experimentally were very close to the theoretical figures.

MULTIPLE DELAY OUTPUTS

Using the audio dual output board which provided another two AD565 D/A converters, it was possible to provide another two audio outputs to give three outputs, each with an independent delay time. Figure 5 shows a block diagram of the three-output delay system using the 6809. With three audio outputs and three independent delay times, the machine language program had to be modified and expanded. The simplified flow-chart for the three-output machine language program is shown in Fig. 4. The input and output process blocks are essentially similar to those within dotted lines for the single-output program in Fig. 3.

The additional instructions required to output the audio samples through the additional two AD565 D/A converters, compared with the earlier program, result in a longer loop. In addition, the 'housekeeping' at the end of each loop becomes more complex. This is due to the fact that we now have four addresses to increment and reset when the block end is reached, instead of two as before. In addition, the 'housekeeping' for each address has to ensure that the program loops which reach the block end are of the same

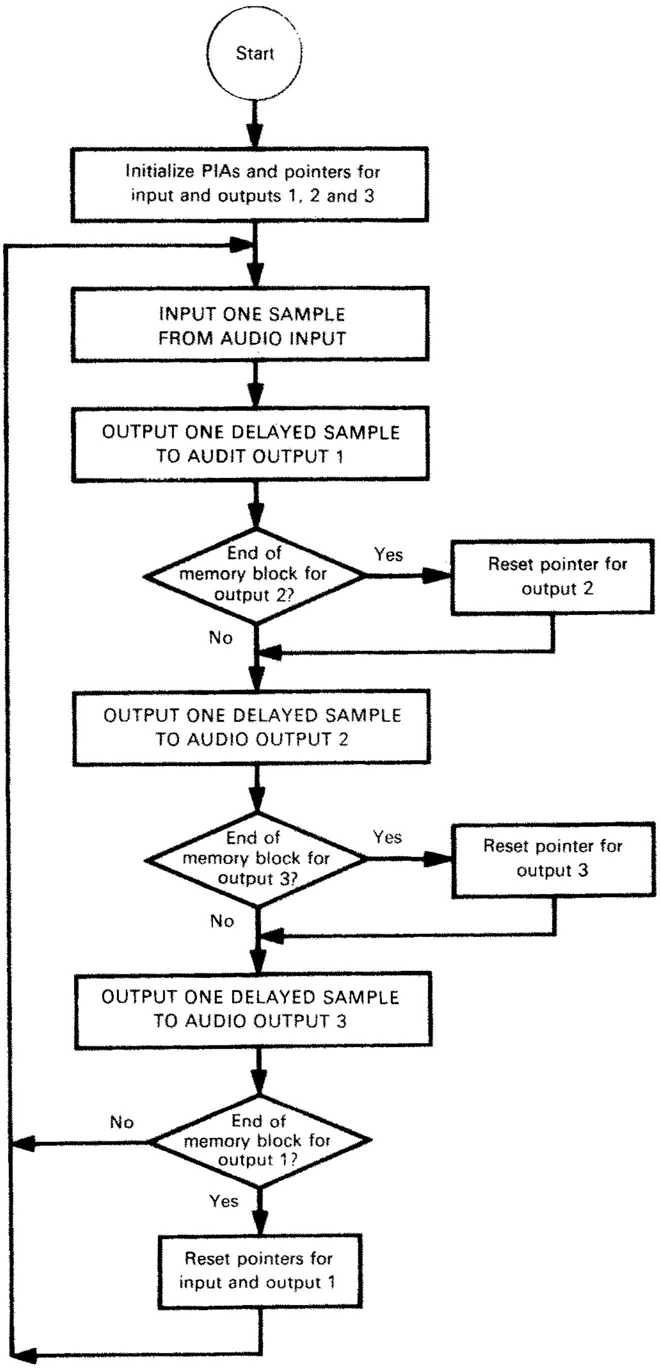
Start
Initialize PIAs and pointers for input and outputs 1, 2 and 3
INPUT ONE SAMPLE FROM AUDIO INPUT
OUTPUT ONE DELAYED SAMPLE TO AUDIT OUTPUT 1
End of memory block for output 2?
Yes
Reset pointer for output 2
No
OUTPUT ONE DELAYED SAMPLE TO AUDIO OUTPUT 2
End of memory block for output 3?
Yes
Reset pointer for output 3
No
OUTPUT ONE DELAYED SAMPLE TO AUDIO OUTPUT 3
No
End of memory block for output 1?
Yes
Reset pointers for input and output 1

Fig. 4. Flowchart of three-output 6809 program.

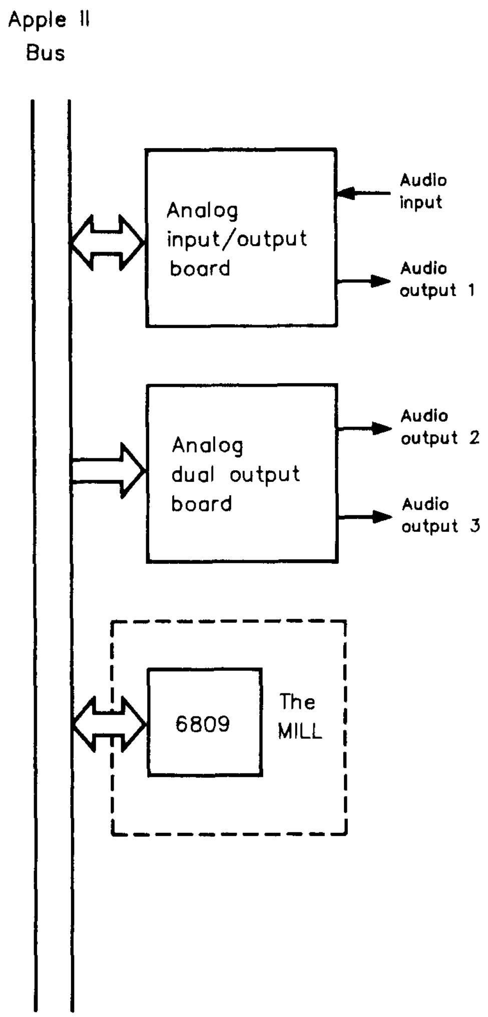

Fig. 5. Block diagram of three-output 6809 delay system.

duration as the regular loops, by using dummy instructions whenever necessary to equalise all loop durations.

The writing of the program for three outputs was greatly facilitated by the fact that the 6809 not only has two index registers, but also has two similar registers, the U and S registers meant to be stack pointers but which can also be used as extra output pointers for the two extra outputs. However, the longer and more complex program loop results in a loop duration nearly three times that for the single-output program. Consequently, the sampling rate falls drastically and the audio bandwidth is only slightly more than one-third of the one-output case i.e. only 5·22 kHz as compared to 13·82 kHz.

TABLE 1
Parameters for the Three Different Programs, 6809 Processor

Program	T_d *(microseconds)*	T_l *(microseconds)*	*Bandwidth (kHz)*	*Time delay (milliseconds)*
1 output	27·37	36·17	13·82	148·2
2 outputs	50·83	65·49	7·63	268·3
3 outputs	80·16	95·80	5·22	392·4

This low bandwidth is of practical value only in situations where speech-quality audio is sufficient.

For comparison purposes, a machine language program was written giving only two audio outputs. The program was similar in principle to the three-output program but used only two of the three AD565 D/A converters. In consequence, the delay loop time was intermediate between the one-output and three-output cases, and hence the audio bandwidth was 7·63 kHz, midway between the other two cases.

Table 1 shows the bandwidths attained and the longest delay times possible for a memory block of length $1000(or decimal 4096) for each of the three different programs. As the program loop length T_l gets longer, the sampling rate and the bandwidth go down. However, the delay obtained for a fixed number of loops goes up proportionately.

The 6809 coprocessor is activated by the 6502 to run the 6809 machine language programs through a special control register on the 6809 coprocessor board. For the single-output program, the duration of the desired time delay is determined by the block length. Hence the end-of-block address must be calculated beforehand and stored in the addresses $00EE and $00EF.

CONTROL OF THE DELAY TIME

With this microprocessor-controlled delay time system, precise control of the time delay duration is easily achieved through software, making remote control of the delay time possible. This is done by changing the time spent in the memory by a digitized sample before it is retrieved and output to the D/A converter. If we consider, for example, the single-output 6809 delay program, this length of time depends on the following two factors:

1. The offset between the input and output addresses.
2. The length and hence the end address of the memory block.

In the program, the initial input address is set at $3000 and the initial output address at $3002, so that the maximum delay is obtained for a given memory block length. Once the initial addresses are set, the offset between them is maintained as long as they remain in the program loop. To change the delay time by changing the offset, a different value could be substituted for the initial output address.

The second method is to change the length of the memory block. In this particular program, the end address can be changed by changing the contents of the two addresses $00EE and $00EF which store the 16-bit end-of-block address. A short routine can be written in 6502 machine code so that the 6502 changes the contents of these two addresses as desired. The 6502 facilitates this real-time change in the end-of-block address and hence of the time delay since it is running simultaneously as a coprocessor to the 6809. This capability was successfully verified experimentally. A program was also written in BASIC to provide a user-friendly environment by which the delay times could be changed easily and automatically by simply entering the required times.

Some commercial stand-alone units do allow this dynamic control of the delay time, usually via some analog parameter of limited precision such as resistance or voltage. Using the 6809 program, precise control of the time delay in accordance with some rapidly changing real-time parameter is also possible. This could lead to interesting psycho-acoustic experiments on phenomena such as the Haas effect which were not possible before. Sound reinforcement systems which changed their delay times dynamically in response to changing acoustic conditions and sound intensities would also be possible.

THE 68008 MPU BOARDS

The longer loop delay times obtained for the 6809 three-output program resulted in a considerably lower audio bandwidth which could be handled by the system. Thus it was decided to speed up the time delay program by using two different 68008 coprocessor boards for the Apple II. Both these boards are available from Stellation Two, the same company which markets the 6809 Mill coprocessor board. These two boards are the McMill board and the Q-68 board.

The 68008 MPU is designed to run at a much higher clock speed than the 6809 or 6502. Thus in both the McMill and Q-68 boards, the 68008 is driven from the 7·16 MHz clock signal available from the Apple II bus, and not the normal 1·023 MHz Apple II clock. Whenever the 68008 in both boards has to access the Apple II bus, it is slowed down considerably to the clock rate which the bus can handle. The major difference between the McMill and Q-68 boards is that the McMill does not have any on-board RAM, while the Q-68 can carry up to 8 K bytes of RAM on-board.

This distinction is quite important when the speed of processing is considered. The advantage of faster processing which the 68008 offers may be nullified if it has constantly to access the Apple II's memory. This is due to the fact that transactions done via the Apple II bus have to operate at the Apple II's regular clock speed, and thus will slow down the 68008 considerably. Hence the McMill, in spite of its much more powerful MPU has no speed advantage over the 6809 if it has to constantly refer to the Apple II memory. Its superior power is really utilised only in its own internal operations.

A 68008 time delay program for a single audio output was written for the McMill. This program is similar to the single-output program for the 6809. It was not possible to calculate the precise time taken for one loop because the clock speeds of the instructions which access the Apple II bus were not clearly specified in the McMill manual. However, the loop time was experimentally determined by observing the total time taken for the loop to execute a large number of times, and then dividing the observed time by the number of loops performed. This gave an experimental value of 94·44 microseconds for the loop time, from which the expected audio bandwidth was calculated to be 5·29 kHz.

This performance is much worse than that for the 6809 program because the 68008 has constantly to access the Apple II bus. Even if the 68008 program does not explicitly refer to Apple II memory addresses, it has to access the Apple II memory simply because the 68008 program itself is stored in that memory. Before each instruction can be executed by the 68008, it has to be fetched by the 68008 from the Apple II memory and brought into the 68008 to be decoded and executed.

THE Q-68 68008 DELAY PROGRAM

Like the McMill, the Q-68 has a 68008 on-board which can run simultaneously with the Apple II's 6502.[5] However, the Q-68, unlike the McMill, is capable of carrying up to 8 K bytes of RAM on-board. A time delay program was written in 68008 machine language for the Q-68 to provide a single delayed output. The flowchart of the program is similar to that of the single-output delay program for the 6809. Unlike the program for the McMill, the memory block for the delay program was defined to be on-board the Q-68 itself. As only 2 K RAM was available on-board the Q-68 used, the memory block could not be longer than 2 K. However, the program could be modified easily for the maximum 8 K which the Q-68 is capable of holding.

The 68008 program, when loaded from the Apple II disk drive, is initially placed in the Apple II memory. It would be quite possible for the 68008 program to be run while it is in the Apple II memory, but as in the case of the McMill board, this would negate the advantage of putting the memory block for the delayed samples in the Q-68's on-board memory, since the 68008 would still have constantly to access the Apple II bus to fetch the program instructions. Hence the program is initially placed in two segments of Apple II RAM starting at addresses $1000 and $2000. The segment at $2000 is the delay program proper; the short segment at $1000 has the

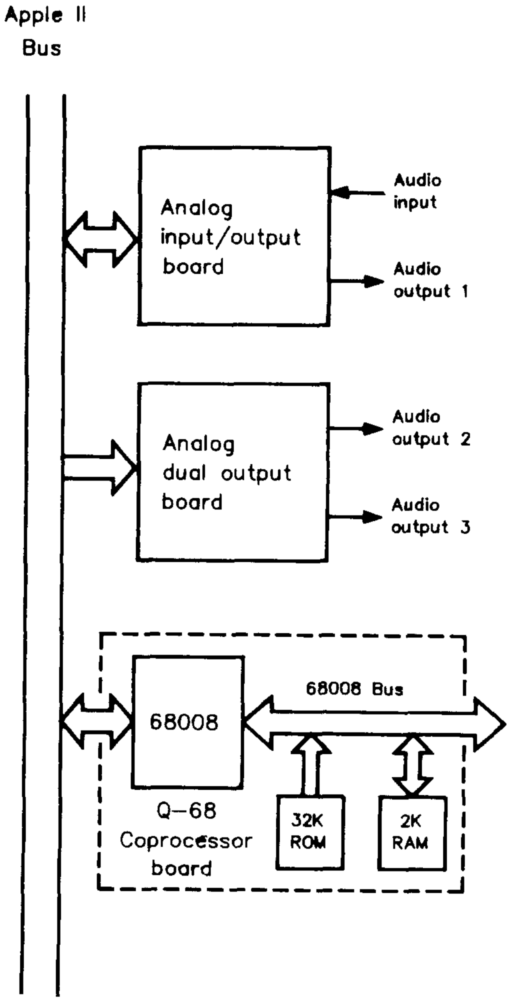

Fig. 6. Block diagram of three-output 68008 delay system.

function of moving the main segment at \$2000 to the Q-68 memory. Thus when the 68008 begins executing the program from Apple II RAM, it encounters the portion at \$1000 first, moves the segment at \$2000 to the Q-68's on-board memory, and then begins executing the main delay program from the on-board memory.

The on-board 2 K RAM occupies the addresses from \$18000 to \$187FF. The 68008 program itself is moved into the RAM to begin at \$18000. The memory block is defined to begin at \$18180. The end-of-block is initially stored in the Apple II RAM at \$1800, but is then moved into the 68008 D3 register. The end-of-block address is defined as the last address on board, i.e. \$187FF, so that the length of the block is \$67F or decimal 1663.

The computation of the delay time was slightly complicated by the fact that the clock cycle of the 68008 would be slowed down whenever it accessed the Apple II bus. This bus access was still necessary in order to input audio samples from the AD574 and output delayed audio samples to the AD565. When the 68008 is not accessing the bus, it runs at a clock speed of 7·16 MHz, giving a clock period of about 0·14 microseconds. From information about the 68008 instruction cycles in the 68008 manual, it was deduced that of the 16 clock cycles taken up in accessing the bus, four were at the Apple II bus frequency of 1·023 MHz.

To provide a comparison with the 6809, delay programs were also written for two and three audio outputs. Figure 6 shows the block diagram of the three-output delay system using the 68008. The flowchart for the three-output program is functionally identical to that for the three-output 6809 program. The corresponding values of T_d and T_i as previously defined were calculated for all three 68008 programs, and the audio bandwidths obtained from these values. As Table 2 shows, the improvement is most marked for the two and three-output cases, where the bandwidth is almost doubled.

TABLE 2
Parameters for the Three Different Programs, 68008 Processor

Program	T_d	T_i	*Bandwidth*	*Time delay*
	(microseconds)		*(kHz)*	*(milliseconds)*
1 output	13·41	34·92	14·29	143·0
2 outputs	27·37	37·43	13·36	153·3
3 outputs	41·34	51·39	9·73	210·5

CONTROL OF THE 68008 DELAY TIME

For the 68008 program, unlike the 6809 program, the delay time cannot be directly altered by the 6502 while the 68008 program is running. This is due to the fact that when the 68008 program is running, the end-of-block address is in the on-board memory which is not directly accessible to the 6502. Thus in order to change the delay time, the new delay value (e.g. the new end-of-block address) will have to be first loaded into the Apple II memory, and then transferred to the on-board RAM so that the corresponding pointers in the 68008 program are altered. The 68008 will itself have to do this transfer, since the 6502 cannot do it, as it has no access to the on-board memory.

One way of doing this is to initiate an interrupt on the 68008, so that the delay program will be briefly stopped to enable the 68008 to transfer the new address to the on-board memory. When the 68008 resumes the time delay program, it will be with the new block length and hence the new time delay duration. This method of changing the time delay was verified experimentally and found to be feasible. As for the 6809, the time delay can thus be dynamically varied under software control or by external parameters. It would be even more desirable for the 6502 to access the on-board memory directly, which is possible but would require major alterations to the circuitry of the Q-68.

CONCLUSIONS

It has been demonstrated that the dynamic control of delay time duration is possible, using an inexpensive Apple II microcomputer and simple additional hardware together with coprocessors such as the 6809 and 68008. In addition to providing an inexpensive method of creating digital audio time delays for owners of such microcomputer systems, it opens up many possibilities for audio research and applications in which the time delay can be dynamically and continuously changed by remote control or under software control, or even under the real-time control of dynamically varying external parameters. We believe that the system we have described could be of use to many workers in audio applications and research.

REFERENCES

1. *Analog Devices data acquisition databook 1984, vol. 1: integrated circuits.* Analog Devices, Norwood, MA, USA, 1984.
2. *Apple II Reference Manual.* Apple Computers, Cupertino, CA, USA.
3. Tan, B. T. G. & Tay, L. P., 12-bit digital audio time delay using the 6809. *Microprocessors and Microsystems*, **10** (1986) 500–5.
4. *The Mill—principles of operation.* (Manual for 6809 coprocessor board). Stellation Two Inc., Oakview, CA, USA, 1981.

5. *QPAK-68 System Reference Manual.* Stellation Two Inc., Oakview, CA, USA, 1983.
6. *MC68008 Advance Information.* Motorola Semiconductors, Austin, TX, USA, 1985.

APPENDIX: THE 6809 AND 68008 COPROCESSOR BOARDS

The MILL 6809 coprocessor board

The Mill is a coprocessor board which plugs into one of the Apple II slots and provides a 6809 microprocessor which can run simultaneously with the Apple II's 6502. The 6809 is controlled by a special control register on the Mill itself.[4] This control register has 8 bits, each of which controls a particular function of the 6809. The 8 bits of the control register are allocated 8 addresses in the Apple's memory map from SLOT + 0 to SLOT + 7, where SLOT is the base address of the slot in which the Mill is inserted. Each of these addresses will access one bit of the control register. For example, the base address of slot 4 is $C0B0.

The activating of the 6809 is controlled via bit 1 of the Mill's control register. If bit 1 is set to a '1', the 6809 will be activated, while if bit 1 is cleared to '0', the 6809 will be halted. Bit 1 can be set or cleared by writing either a '1' or '0' respectively into the most significant bit of the address SLOT + 1. Hence the 6809 can be started or stopped by writing a suitable value into the address SLOT + 1.

The Q-68 68008 coprocessor board

The Q-68 is a plug-in 68008 coprocessor board for the Apple II which has on-board RAM and ROM in addition to the 68008. The 68008 is actually capable of addressing up to 1 048 576 (i.e. 1024 × 1024 or 1 M) addresses.[5] This total can be divided into 16 'pages' of 64 K addresses each. In the Q-68, the lowest 64 K addresses are devoted to the entire memory map of the Apple II, and the next 64 K addresses are devoted to memory carried on the Q-68 board itself. The other 14 pages are available for off-board expansion, and the Q-68 board carries an external connector for this purpose. The 1 M address memory space means that the memory addresses are 20 bits long and have to be expressed as five hex digits i.e. from $00000 to $FFFFF. The on-board memory comprises a 8 K ROM which can be expanded to 32 K and 2 K RAM which can be expanded to 8 K.

For the lowest 64 K with which the Q-68 addresses the Apple II memory, the 68008 views the RAM, I/O and ROM in the Apple II differently from the 6502. The 6502 places the Apple RAM in the lowest 48 K addresses from

$0000 to $BFFF, has the I/O in the 4 K from $C000 to $CFFF and leaves the top 12 K from $D000 to $FFFF for ROM or for extra RAM. These Apple II addresses are altered by an address 'translation' so that the 68008 sees them at different addresses. The translation involves a 'swop' between two 1 K blocks of addresses as seen by the Q-68. The 1 K block addressed as $0000 to $03FF by the Apple II is addressed as $0800 to $0BFF by the Q-68 while the 1 K block addressed as $0800 to $0BFF by the Apple II is addressed as $0000 to $03FF by the Q-68. The address translation is performed by hardware on board the Q-68.

The 68008 on the Q-68 can be controlled by three software switches on the board each of which has a separate address and function as follows:

$C0n0	Turn 68008 off
$C0n1	Turn 68008 on
$C0n3	Cause Level 7 interrupt

where n is the number of the slot the Q-68 is inserted into plus 8. The Level 7 interrupt is able to stop the current program the 68008 is executing and make it execute another program (the interrupt routine). When the interrupt routine is completed, the 68008 resumes execution of the program which was interrupted.

Each time one of the above addresses is accessed, the corresponding function will be activated. Before the 68008 is turned on, the starting addresses of the machine language program to be executed should be stored as a 32-bit number in the four addresses from $00004 to $00007. The stack pointer also has to be stored as another 32-bit number in $00000 to $00003. The stack pointer is needed to define a section of memory called the 'stack' which is used by the 68808 in executing its instructions.

Time taken by the 68008 to access the Apple II bus

The following assumptions were made in the computation of the time taken to execute instructions which accessed the Apple II bus:

1. When one 12-bit audio data sample is moved from one address to another, two complete memory read cycles are required to fetch the data and two complete memory write cycles are required to store the data since the bus only handles 8 data bits in each single memory read or write cycle.
2. Both memory read and write cycles are assumed, as given in the Motorola 68008 manual,[6] to take four clock cycles.
3. From the 68008 read/write timing diagrams given in the 68008 manual, each 8-bit read or write cycle puts the 8-bit data on the bus for

two of the four clock cycles. Thus two of the four clock cycles are assumed to be at 7·16 MHz, while the other two during which the bus is accessed are assumed to be at 1·023 MHz, the Apple II's clock frequency.

Thus, for example, when the 68008 retrieves one 12-bit sample from its on-board RAM and sends it out to the AD565, the following sequence occurs:

1st read cycle (4 clock cycles): 8 bits retrieved from RAM
1st write cycle (4 clock cycles): 8 bits sent to AD565
2nd read cycle (4 clock cycles): 4 bits retrieved from RAM
2nd write cycle (4 clock cycles): 4 bits sent to AD565

Of the 16 clock cycles, the 8 clock cycles in the read cycles are at 7·16 MHz since the bus is not involved. For the write cycles, two out of the four clock cycles are at 1·023 MHz as the data has to go on the bus to reach the AD565. Thus four out of the 16 clock cycles are assumed to be at 1·023 MHz and the time taken for the instruction is accordingly calculated. The time taken for an input operation when an audio sample is input from the AD574 and stored in the on-board RAM is similarly calculated.