CSIRAC TIMING

Background

In 1948 Trevor Pearcey and Maston Beard of the Australian CSIR (Council for Scientific and Industrial Research) commenced the design of a stored program electronic computer. The computer – the CSIR Mk1 (later renamed CSIRAC) ran its first test program in late 1949.

CSIRAC was Australia's first computer and one of the first in the world. The computer had a long working life and was decommissioned in late 1964. However, it survives intact (although not operational) as the sole survivor of the first generation of stored program machines.

CSIRAC was designed as a bit-serial machine using mercury delay lines to implement its main store. This provided a design which was logically elegant and economical to construct.. During its long working life the computer underwent several modifications to its timing in order to optimise execution speed. This paper, written by the engineer who maintained CSIRAC for much of its life, and who devised and implemented a major improvement in its performance, gives an insight into the technical complexities of this remarkable machine.

Introduction

There have been many descriptions of the units that existed within the design of CSIRAC, the most important, of course, being the writings of the original designers, Maston Beard and Trevor Pearcey, and at a later date Frank Hirst. One area that does not appear to have been recorded in detail relates to the timing, an important feature in a serial type machine such as CSIRAC.

The fact that 16 words circulated in each logical mercury delay line each being stored or accessed by a (unique) address ranging from 0 to 15 has appeared frequently. The only other detail that needed to be known by any user of CSIRAC was that address positions 12, 13 and 14 did not allow the speed-up process to occur, and this has often been mentioned.

It was only when a program had an output that depended upon the real time processing of some commands that precise timing details had to be known; the Music program was the only one in this category.

In the following description of the timing process the features of CSIRAC's operation have been included, hopefully in sufficient detail to allow an understanding of the timing process.

Description

The computer known as CSIRAC is a *serial* digital machine where the digits of a number move one after another along a single digit trunk.

It operates in the binary scale where a "1" is represented by a positive pulse of 1.0 microsecond duration and approximately +50 volts amplitude and the absence of a pulse representing a "0".

These digit pulses are spaced 3 microseconds apart as determined by the basic clock frequency of 333 kHz. A group of 20 binary digits is used to define a *word* which can either be a number or an instruction, depending on how it is used.

The time taken to serially transmit a word is 60 microseconds and this period is known as a *minor cycle*.

Mercury Delay Line Memory

The computer has a mercury delay line memory which is an ultrasonic type storage system. The words are ‘stored’ by keeping them circulating around a closed loop consisting of electronic circuitry that includes a delay line containing mercury.

At the input end of the delay line the digit pulses modulate a 10 MHz carrier and these RF pulses are applied to an X-cut quartz crystal which transforms them into ultrasonic waves. These waves travel relatively slowly down the column of mercury, impinging on an identical crystal located at the output end where they are transformed back into RF pulses. After amplification, detection, shaping, and timing adjustment they are again identical to the original digit pulse train and ready to traverse the loop again. By these means, while the equipment performs fault-free and the power is maintained, information can be retained indefinitely.

The amount of information that can be stored in each delay line loop is determined by the length of the mercury column and the digit spacing. The mercury delay lines used in CSIRAC have a 54 inch column of mercury and a digit pulse spacing of three microseconds. The delay in the mercury column is some 958 microseconds and electrical delay lines in the external circuitry bring the total delay to 960 microseconds. This allows 16 words, a total of 320 digits, to be stored in each loop, and this period of 960 microseconds is known as a *major cycle*. The design allows for 32 physical mercury delay lines to exist, and with interspacing allowing two logical loops per delay line loop a total of 1024 words of storage is possible.

The three microsecond digit spacing was chosen to assist in the design of reliable circuitry for the series of gates, flip-flops and number of stages in the arithmetic functions of the computer operation using the valve types available at the time.

In the memory electronic circuitry the read-in and read-out gates were relatively simple. The bandwidth of the mercury delay line and its associated amplifier was such that within the memory system a digit spacing of only 1.5 microseconds would be possible, and by an interspacing feature two logical loops, one called ‘direct’ and the other ‘interspaced’, each of sixteen words, could circulate around each physical delay line loop, one interlaced with the other. A doubling of the memory capacity was attained with a relatively small increase in circuitry without compromising the performance of the rest of the computer.

Clock Pulse Generator

The main clock generator of the computer produces a continuous train of positive pulses 50 volts in amplitude and one microsecond duration occurring every three microseconds; they are called PP′ pulses. They are used throughout the computer, outside of the memory area, in the generation of the digit pulses that make up the computer words. The other important function they perform is in gating circuits to restore the shape and timing of the digit pulses where they have started to become deteriorated. A pulse called a P pulse is produced at every twentieth PP′ position and its purpose is to denote the start time of a word; its period of sixty microseconds is called a *minor cycle*. It is used to start the generation of waveforms that are used in dealing with one or more words within the computer operation.

Coinciding with every sixteenth P pulse, a C pulse is generated and it is used to indicate the starting time of the circulation period around each memory loop. This C pulse period of 960 microseconds is known as a *major cycle*.

This set of pulses allows synchronisation to be maintained throughout all parts of the computer.

Memory Interspace Feature

To describe the method of interspacing within the memory system an explanation of the decoding of an instruction is needed. The twenty digits making up an instruction are identified as p1 (the least significant) to p20 (the most significant). In the serial transmission of these twenty digits the first in the train is the least significant. Although not relevant at this point it is a necessary condition for multiplication. The five digits p1 to p5 specify which of the thirty-two destinations, and the p6 to p10 the source required, in the particular operation to be executed. The digits p11 to p20 are used to identify which of the 1024 possible locations available in the mercury delay line or magnetic disc memories is required. In the delay line memory the digits p16 to p20 specify which of the 32 physical delay lines is accessed; the p15 digit selects the direct or interspaced word and the p11-p14 digits determine which of the 16 time positions it occupies.

As mentioned earlier, within the memory section of CSIRAC there are digits with a 1.5 microsecond spacing. For clocking and reshaping these pulses there is a *memory clock* of twice the main clock frequency, synchronised by PP′ pulses so the memory is in synchronism with the rest of the computer. The memory clock pulses are of similar amplitude to the PP′ pulses but of 0.75 microsecond width (Fig. 1). The memory also has its own input and output trunks to handle the interspaced digits. An important point is that the output trunk copies from the memory loop 1.5 microseconds earlier than normal.

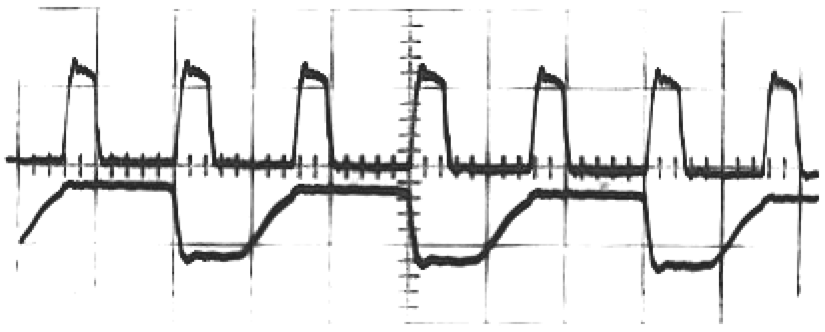


Figure 1. Memory Clock (Top) and PP pulses
(From actual photograph 1964)
Scale: Vert. 50V/div. Hor. 1μs/div.

When reading into a direct word position the digits appearing on the main computer input trunk are gated by the appropriate waveforms and then delayed by 1.5 microseconds before entering the memory input trunk and then eventually go into the desired delay line. An interspaced word read into the same time position and delay line, goes from the main input trunk via gates but is undelayed when entering the memory input trunk, and so its digits appear 1.5 microseconds earlier than the direct ones and are interspaced between them in the delay line.

When reading out of a direct location the digits on the memory output trunk, which started 1.5 microseconds late, are now in synchronism again, and are gated onto the main computer output trunk. The interspaced location digits when read out to the memory output trunk are 1.5 microseconds early and are delayed 1.5 microseconds when being gated onto the main computer output trunk.

By these means the computer can just treat the memory as though there are 64 delay lines each containing 16 words; unfortunately the engineer was not quite so lucky.

The Computer Cycle Operation (MODE 0)

A program for CSIRAC consists of a set of commands, each being a 20 binary digit word, and these are stored in consecutive locations in the mercury delay line memory. Each command results in the transfer of a 20 digit number in one of the dozen or so registers via a *source gate* into a register via a *destination gate*. The pulses when moving from their source register may undergo a transformation due to the characteristics of the source gate and similarly as they enter the destination register via its destination gate. There are 32 different source gates and the same number of destination gates available although there are only about a dozen different registers. This is a result of some registers having more than one gate associated with them; an example is the 'A' register which has 7 different source gates and 6 different destination gates attached to it.

CSIRAC has a set sequence of operations known as a *Computer Cycle* that is repeated until all the commands of a program have been executed. This Computer Cycle is controlled by a set of gating waveforms generated by the *Sequence Control Unit*. Other units needed are the *Sequence Register* which holds the memory address of the next command to be executed, and to decode this address there is a *Line Selector* which selects one of the 64 logical delay lines and a *Time Selector* to decode the minor cycle time that it occupies.

The *Interpreter* receives the 20 digit command from the memory when it becomes available, and at the appropriate time the address digits are read to the Time and Line selectors and the source and destination information is sent to the *Source Selector* and *Destination Selector*. At a time determined by its input the Time Selector produces a one minor cycle waveform that joins the output trunk to the input trunk; one word length of information now passes from the selected source to the selected destination.

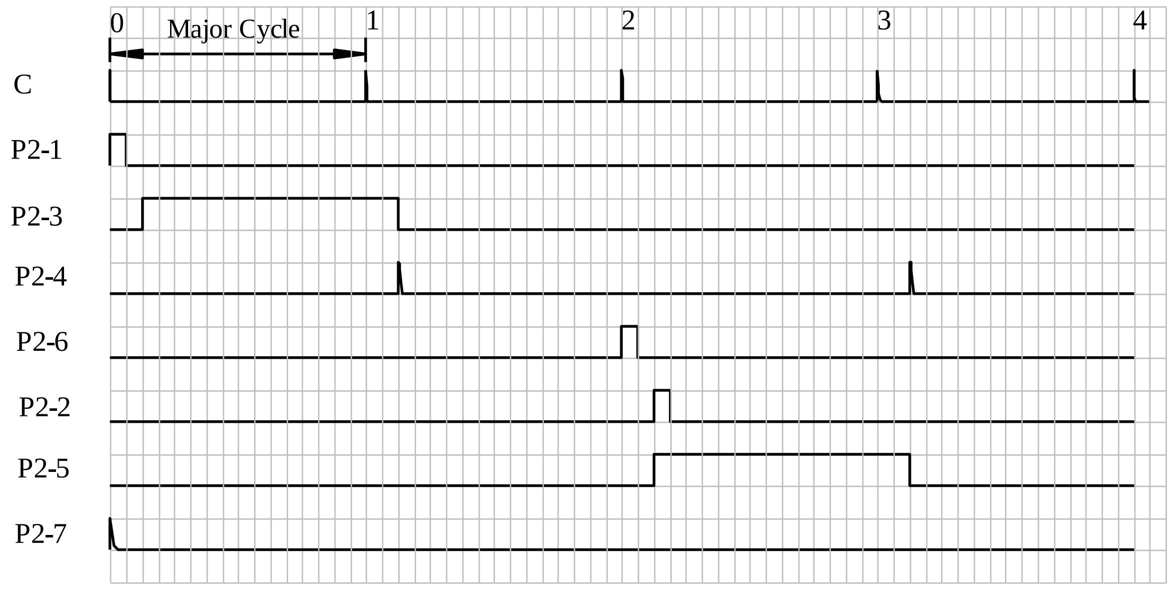


Figure 2. Sequence Control Unit waveforms

The Sequence Control gating waveforms are shown in Fig. 2, and the complete sequence was originally designed to occur over a period of 4 major cycles, the original period of the Computer Cycle. For convenience a major cycle period is occasionally referred to as one millisecond instead of 960 microseconds. This set of waveforms is repeated continuously unless a special operation needs longer than 4 major cycles for its completion. For testing purposes, provision is made to stop after one cycle each time the unit receives a start signal.

The sequence of events during a computer cycle is as follows:

(a) During the one minor cycle P2-1 waveform the p11 to p20 digits of the sequence register are transferred to the Time Selector and the Line Selector.

(b) There is then a one minor cycle gap to allow for the relatively slow rise time of the Line Selector outputs and then during the major cycle P2-3 waveform the selected memory loop gate is activated during the minor cycle determined by the Time Selector and a copy of the command is transferred via the output and input trunks to the Interpreter.

(c) When the transfer is completed the Time and Line selectors are cleared by P2-4.

(d) During the one minor cycle period of P2-6 the p11 to p20 digits of the command are transferred from the Interpreter to the Time and Line selectors. Also during this period the static register within the Interpreter which receives p1 to p10 digits of the command is set up and at the end of P2-6 this information is sent via a cable to the Source and Destination selectors. Again a one minor cycle period is allowed for the relatively slow rise time due to cable capacitance.

(e) At the start of the P2-5 period that occurs after P2-6, the P2-2 waveform adds a p11 digit to the Sequence Register to indicate that the next command is in the next memory location. During the P2-5 major cycle period, at the time specified by the command address, the one minor cycle waveform produced by the Time Selector joins the output trunk to the input trunk. The output trunk has access to the selected source register through its gate, and the input trunk is joined to the selected destination register through its gate. So for the one minor cycle the two trunks are joined, one word is transferred from source to destination and the command has been executed.

(f) The Time and Line selectors are again cleared by P2-4.

(g) The P2-7 pulse clears the Interpreter.

The Initial Speed-up and Timing (MODE 1)

When looking at the waveforms in Fig. 2 it can be seen that between the end of P2-3 and the next waveform P2-6, is a period of 14 minor cycles; this is necessary so that the timing can be maintained by starting P2-5 two minor cycles after a C pulse. However when the position in the memory allows the command to be transferred to the Interpreter at least one minor cycle before the next C pulse after the start, P2-3 can be terminated at the end of the transfer period and the following waveforms can start one millisecond earlier. Similarly if the execution time of the command during P2-5 is early enough then the last major cycle is not needed. A relatively simple circuit modification was all that was necessary to achieve this increase in speed of operation and appropriately the pulse to initiate it, P13-2, is known as the *Speed-up pulse.* In the initial design of CSIRAC the computer cycle was always 4 milliseconds, but Trevor Pearcey mentioned in his early literature that he expected it would be reduced to 2 milliseconds at a later date.

In 1952 when CSIRAC was operating in a 'closed shop' manner a toggle switch on the operator's panel labelled 'speed-up' was installed. When operating in the speed-up mode and a fault was suspected it was convenient to eliminate the speed-up procedure as the reason by switching it off. It was found to be perfectly reliable and so the switch was left in the 'on' position.

When the timing of commands is important it is necessary to know the exact position relative to a C pulse that the p11 to p14 time digits reference. The first minor cycle period of waveform P2-3 is selected when the time digits are 1011, i.e. 11 in decimal notation. The next minor cycle is selected by 1010 (10), and so on down to 0000 (0); then follows 1111 (15), 1110 (14), 1101 (13) and finally 1100 (12) which is the last minor cycle of the P2-3 major cycle. In Fig. 3 it can be seen that all but the 12, 13 and 14 configurations of the time digits select minor cycles that finish at least one minor cycle before the next C pulse and so in the majority of cases speed-up is possible. This minor cycle delay is necessary to prevent any clash between the pulses terminating the P2-3 and the C pulse. Thus the time taken for a computer cycle can be 2, 3 or 4 milliseconds depending on the memory location of the command and the time digits within the address section of the command.

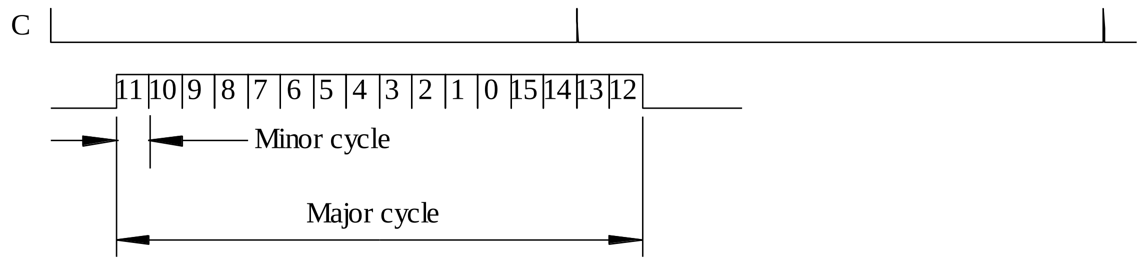


Figure 3. Time positions within P2-3 Major Cycle period

The aim of deciding on a timing sequence would be to have the three positions where speed-up did not occur to be the least likely to be used. The delay line memory and the magnetic disc memory were not of consequence in this decision, rather one would juggle the program around to minimize the effect whatever three locations were involved (their being sequential would be the important factor). The only other register that held more than one word was the D register which was a single mercury delay line identical to the delay lines in the main memory. The D register possessed full arithmetic facilities and not having the ‘inter-space’ feature it held 16 words (D0, D1...D15). Any commands referencing this register had to indicate the particular word required by its address time digits.

The D register was used frequently by programs and the convenient way was to use D0 then as more registers were needed D1 then D2 etc.; the exception was D15 which was used as the main link register to subroutines, and most library routines had this D15 register incorporated in their coding. The timing used, as shown in Fig. 3, satisfies the points made above, and the reason that the time positions appear in a descending order can be seen as a function of the Time Selector, explained later and illustrated in Fig. 4.

Timing Logic Change lets CSIRAC run faster

In early 1962 the operating speed of CSIRAC did not compare very favourably with that of other machines then available. The implementation of an English language type compiler called Interprogram, created by Geoff Hill in 1960, had greatly extended the range of users due to the ease with which it could be mastered.

To reduce the computer cycle time would give CSIRAC added zest, but with the present logic of the timing synchronism that was not possible. The Time Selector contained a four stage static register that received the time digits p11 to p14. In the first section of the computer cycle they came from the Sequence Register during the P2-1 minor cycle, then after the selected command from the memory had been copied to the Interpreter the static register was cleared and ready to receive the p11 to p14 digits from the Interpreter during the P2-6 minor cycle.

To select the time position required the output from each stage of the register had to be in the high or ‘1’ state, and this condition was tested, first under the P2-3, and then later under the P2-5, major cycle periods; each started two minor cycles after a C pulse and so were always in synchronism with the basic timing. The content of the Time Selector register was incremented by a P pulse adding a ‘1’ to the p11 stage after the end of each minor cycle during the P2-3 period. If the register output was not 1111 (15 decimal) for the first minor cycle it would increase at each minor cycle until that value was reached and resulted in an output waveform lasting one minor cycle. This minor cycle pulse called P13 was used to join the Output and Input trunks and so a transfer of information of one word length would take place at the required time. A pulse formed by the differentiated end of P13 and known as P13-2 was used to initiate speed-up when possible (as explained earlier). The same sequence of events takes place under P2-5.

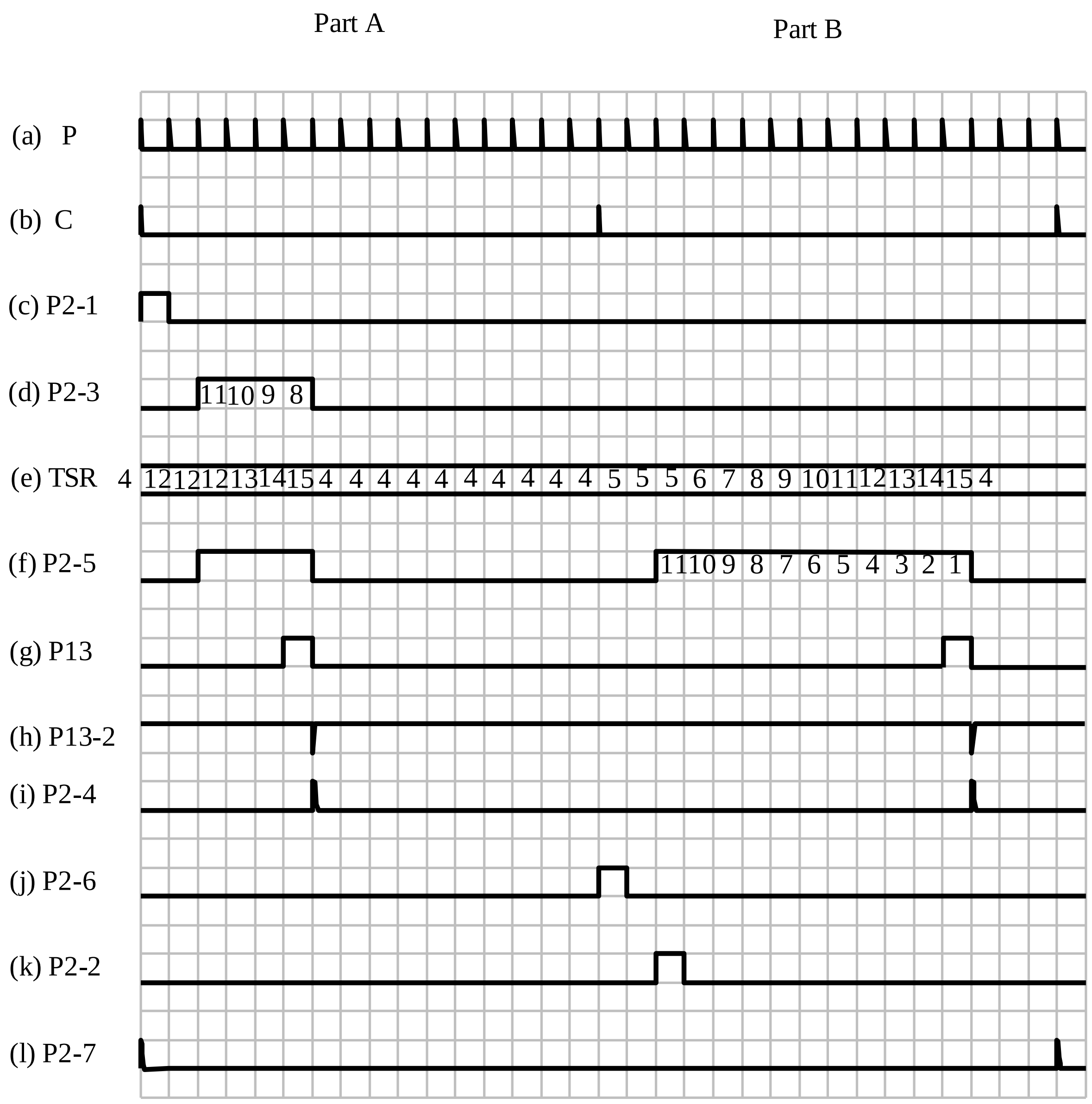
The P2-4 clear pulse presets the 4-digit Time Selector register to 0100 (4 decimal) and when the time digits were read in under P2-1 or P2-6 they were added to this value 4.

Fig. 4 illustrates the timing for a computer cycle where the command is located in time position 8 of one of the delay line loops and the command itself has an address time position of 1 (i.e. the command digits p11-p14 are 0001). The actual command can be written n, 8 0 1 D B where n,8 is the memory location of the command and 0 1 D B is interpreted as 'copy the contents of the register D1 to the B register'.

Fig. 4(a) shows P pulse positions and Fig 4(b) shows C pulse positions for timing.

Fig. 4(c) P2-1 occurs in the first minor cycle of the computer cycle and during this time the Sequence Register contents (p11-p14) are added to the Time Selector register.

Fig. 4(d) P2-3 waveform and later P2-5 waveform (Fig 4(f)) each have their minor cycle time positions numbered.



TSR - Time Selector Register

The command represented is n,8 0 1 D B

Figure 4. Timing with speed-up occurring in both parts of computer cycle

PART A.

Fig. 4(e) shows the Time Selector register contents. Prior to the C pulse time the value is 4 as left by the previous P2-4 clear pulse. After the Sequence Register contents (value equals 8 as this is the time position of the next instruction in memory) are added during P2-1 the value is 12 and remains same until the start of the second minor cycle under P2-3 when a P pulse changes it to 13, then a minor cycle later to 14, and then to 15 at the start of the fourth minor cycle under P2-3; at the end of that minor cycle the register is returned to 4 by clear pulse P2-4 (Fig. 4(i)).

Fig. 4(g) While the register contents were 15, P13 was produced.

Fig. 4(h) The end of P13 is differentiated to produce P13-2 (the speed up pulse) which ends P2-3 (Fig. 4(d)).

Fig. 4(i) P2-4 is produced by the differentiated end of P2-3.

PART B.

Fig. 4(e) After being cleared at the end of P2-3 the Time Selector register contents remain at 4 until at the next C pulse time P2-6 (Fig. 4(j)) commences and the time digits within the command are transferred from the Interpreter (value = 1) and added in making the contents 5. The value remains 5 until the start of the second minor cycle under P2-5 when a P pulse makes it 6 and so on at the start of each successive minor cycle until at the 11th minor cycle it becomes 15 and the output generates P13 until the next P pulse arrives to end the output from the register. P13-2 and P2-4 are generated as happened at the end of the P2-3 sequence.

Fig. 4(k) P2-2 occurs at the start of P2-5 and is used to add one to the Sequence Register.

Fig. 4(l) P2-7 is generated at a C pulse time that coincides with the start of a computer cycle.

The Time Selector is used twice during each computer cycle, in the first half it selects the minor cycle that the current instruction occupies in a memory loop. Then during the second half, when the execution of that command takes place, it has to select the minor cycle for the transfer of information from the source to the destination. Even when no address was actually needed, as for many sources and destinations, time position ‘0’ still has to be selected.

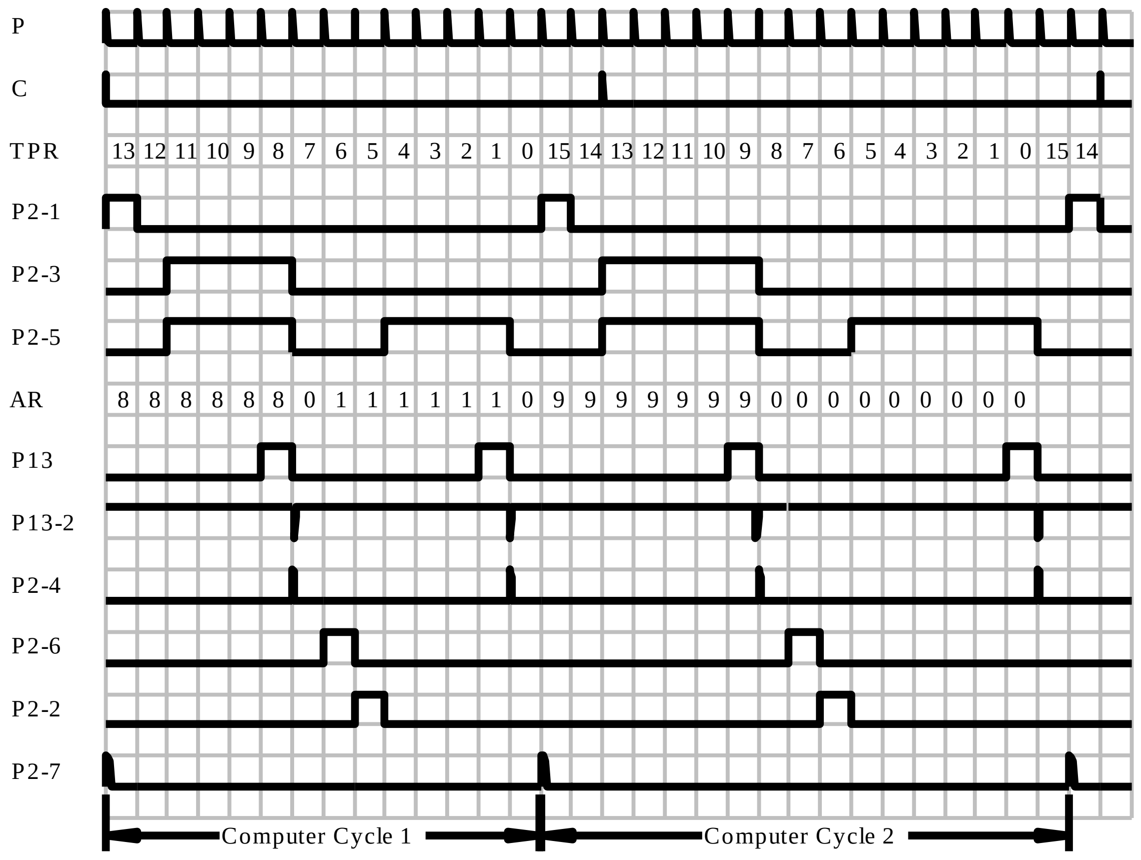
Fig. 4 shows that the time selector has only one register; it is preset by P2-4, then the time position digits are added in and then incrementation each minor cycle finds the required minor cycle. P2-4 then presets it again ready for the same procedure to take place during the second half of the computer cycle. As the incrementing at each minor cycle has to start at a known period from the timing C pulse, the control waveforms for the second half start with P2-6 which is initiated by a C pulse, the other waveforms follow as shown on Fig. 4.

MODE 2 Operation

To be able to reduce the computer cycle time a different method of selecting the required minor cycle time was needed. The C pulse still had to be the timing pulse for determining the minor cycle time position number; there were 16 of them, each 60 microseconds long, and the circulation time around each memory loop was 960 microseconds, the period of the C pulse.

A Time Selector was designed that had two registers. The first, a ‘time position’ register preset by a C pulse, then incremented at minor cycle intervals by a P pulse, would be used to test for the time position minor cycle. The C pulse would always preset it to 13 which was the existing value of the minor cycle after a C pulse (see Fig 3). This maintaining of the existing time positions would be necessary so that programs already in existence would still run correctly if they included commands that were address dependent such as the ‘shift left’ instruction.

The second register, the ‘address register’, would be preset by P2-4, but now the preset value would be zero, and during the P2-1 period the Sequence Register address digits would be read in, unchanged, to this register. During the P2-3 period the constant value of the address is compared with the changing value in the time position register and during the minor cycle when they are identical P13 is generated and P13-2 and P2-4 occur as before. At this point, after P2-4, we still have our time position count progressing in its register so we can proceed with the second half of the computer cycle. After one minor cycle P2-6 is initiated by a P pulse rather than a C pulse and so it becomes possible to complete a computer cycle in less than a millisecond, but remember on most occasions the next command is in the next position in memory and so the time that a command is read from the memory during P13 until the next command can be read during P13 is 15 minor cycles or 900 milliseconds. To achieve this speed it is also necessary to optimize the command’s address with respect to the command’s memory location address. Each memory location has its own 8 sequential values for the command address to achieve maximum speed of execution (or minimum computer cycle time).



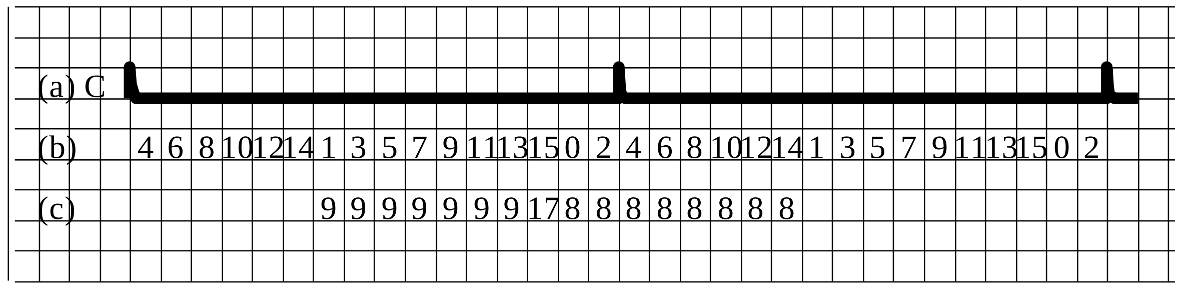
TPR - Time Position Register
AR - Address Register

The two commands are:
n,8 0 1 D B
n,9 0 0 D A

Figure 5. Timing of two commands in successive memory locations using MODE 2 operation

Fig. 5 illustrates the timing of two successive computer cycles; the first command is in a memory location of 8 (i.e. the p11 to p14 address digits of the location are 1000) and the second command is in location 9. The address digits of the first command are 0001 and those of the second command are 0000. The two commands can be written
n,8 0 1 D B (explanation given earlier)
n,9 0 0 D A where n,9 is the memory location of the command and 0 0 D A is interpreted as 'Copy the contents of the register D0 to the A register'.
For convenience the computer cycle is shown starting at a C pulse time, but it could start at the beginning of any minor cycle (i.e. any P pulse time).
P2-6. This now occurs one minor cycle after the end of P2-3, and causes the time digit of the command to be added to the Arithmetic Register.
P2-2. This now occurs immediately after P2-6, is one minor cycle long and causes a P11 to be added to the Sequence Register.
P2-7. This now occurs coincident with the start of P2-1, is generated by a P pulse and clears the Interpreter.
A second Time Selector (to be used with the original one) was constructed incorporating the above logic and testing over a period of time showed that it could be as reliable as the original Time Selector. Some minor changes to the Sequence Control unit involving the P pulse to be used instead of the C pulse were needed for the new Time Selector operation. All of the changes were incorporated by the use of relays activated by a single switch on the operator's panel so that one could change from the original mode to the new one with ease. The only output from the Time Selector was the P13 minor cycle pulse so the source of this output was

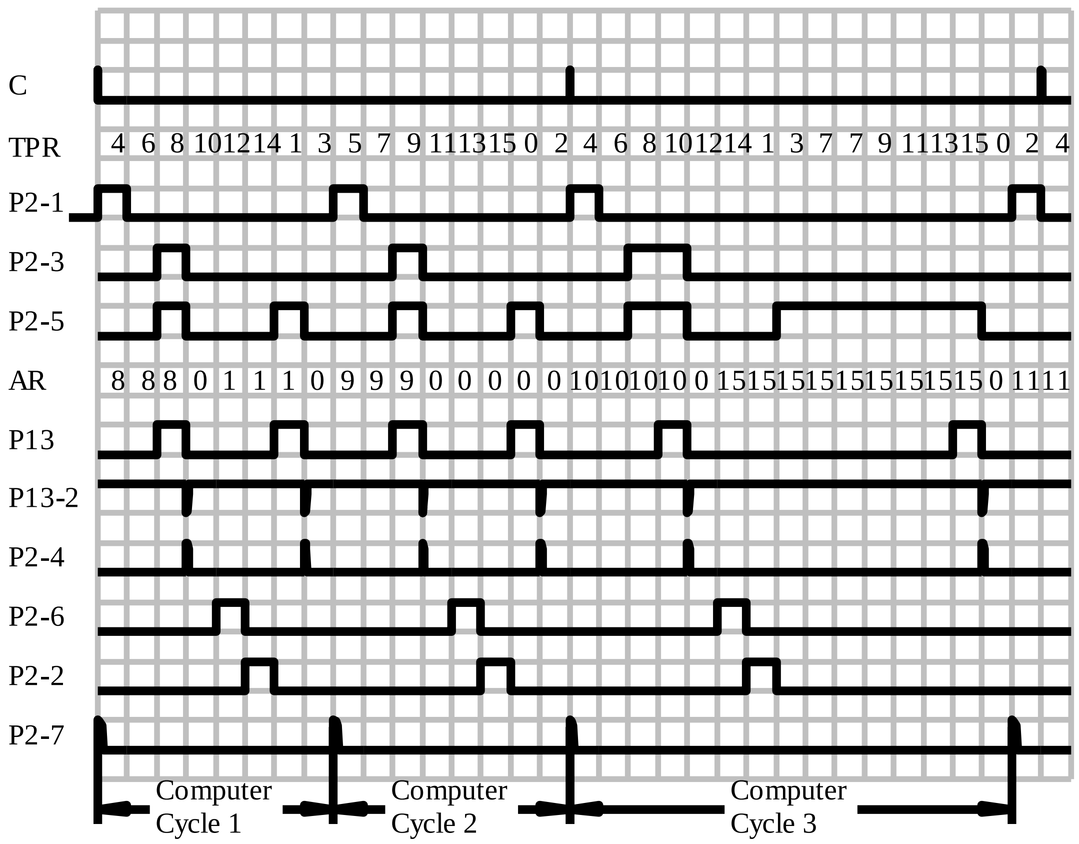
also changed by relay as above. When using the computer with the switch in the ‘off’ position the original Time Selector was in use and this was now referred to as MODE 1 operation. With the switch in the ‘on’ position the output from the new Time Selector was in use along with the Sequence Control Unit changes and this was referred to as MODE 2 operation.
During the design of the second Time Selector it was seen that when optimized addressing was possible in a sequence of commands the time to complete each individual computer cycle was only 8 minor cycles. However even with the new version of the Time Selector, successive locations were 15 minor cycles apart.


(a) C
(b) 4 6 8 10 12 14 1 3 5 7 9 11 13 15 0 2 4 6 8 10 12 14 1 3 5 7 9 11 13 15 0 2
(c) 9 9 9 9 9 9 9 17 8 8 8 8 8 8 8 8

(a) C pulse positions
(b) Mode 3 time position register count
(c) No. of minor cycles to next consecutive memory location

Figure 6. MODE 3 Timing

MODE 3 Operation
By using the same time position register, but changing the arrangement of its connections to the coincidence detector, relative to those from the address register, it was possible to get a forwards count incrementing by two. Fig. 6(b) illustrates the exact sequence obtained; the even numbered memory locations had their next memory location 8 minor cycles ahead, and for the odd numbered ones they were 9 minor cycles ahead, except for location 15 where location 0 was 17 minor cycles away, see Fig. 6(c).
Figure 7 illustrates the waveforms for the computer cycles to execute three commands (the first two being the same ones illustrated in Fig. 5), but using this new configuration of the time position register. The three commands can be written
n, 8 0 1 D B
n, 9 0 0 D A
n,10 0 15 PE PD where 0 15 PE PD is interpreted as ‘Add a digit in the P11 position to the contents of the D15 register’.
These three instructions or commands are the last three (prior to the exit command) in a double precision multiplication routine used in the original loan repayment program run on CSIRAC in Melbourne. The result finished in D0.D1 and was copied to A.B prior to exit from the routine back to the main program.



TPR - Time Position Register
AR - Address Register

The three commands are:

n, 8	0 1	D	B	Copy contents of D1 to register B
n, 9	0 0	D	A	Copy contents of D0 to register A
n, 10	0 15	PE	PD	Add a P11 digit to the value in D15

Figure 7. Timing of three commands in successive memory locations using MODE 3 operation

It can be seen that the three commands can be completed in a period of two major cycles; this mode of operation was known as MODE 3.

Every program that was to be run under MODE 3 operation of CSIRAC had to have its ‘left shift’ instructions modified to allow for the new timing sequence, but this was a small price to pay for those users who had compute-bound programs.

The changing of connections to the coincidence detector to switch from MODE 2 to MODE 3 was achieved by relays operated by a switch on the operator’s panel adjacent to the MODE 1/MODE 2 switch. To operate in MODE 3 the first switch had to be in the MODE 2 position so that the new Time Selector and Sequence Control changes were in use, and then with the MODE 3 switch ON the fastest Computer Cycle timing would be available.

The two new modes of operation of CSIRAC were available from July 1962. MODE 2 was most used as the Interprogram feature would run in this mode but not in MODE 3, and the general user would get a worthwhile increase in speed, even without optimizing, and of course an even greater increase with optimization. MODE 3 found most favour amongst those users with programs that involved repetitive time-consuming calculations.

Table 1 illustrates the progressive reduction in command times achieved when the techniques described above were applied.

MODE 0.	All commands take 4 ms.
MODE 1.	Multiplication and Left Shift take 4 ms. Other commands take 2 ms + 1 ms if the address is 13, 14 or 15 (mod 16) + 1 ms if the store location is 13, 14 or 15 (mod 16) If both + 2 ms. Thus commands may take 2, 3 or 4 ms.
MODE 2.	Multiplication takes 3-4 ms, depending on optimization Left Shift takes 1-3 ms, depending on optimization and the number of Left Shifts. Other commands: if optimized take 1 ms, otherwise < 2 ms.
MODE 3.	Multiplication and Left Shift as for MODE 2. Other commands: if optimized take 0.5 ms, otherwise < 1.5 ms.

Table 1. Command times for each operation mode

How CSIRAC was able to play Music

One of the destinations available in the CSIRAC command structure was a small audio amplifier driving a 5-inch speaker. While this was not a register as such, it could accept a 20 bit digit train applied to the amplifier input for a minor cycle period like the other destination registers.

When a number was sent to this destination, known as the hooter, and represented by the letter P in the written form of CSIRAC's commands, a 'click' was produced by the speaker. The greater the number of '1' bits in the word coming from the source the louder the 'click', and the command with the P destination was usually in a small loop to make sure it was audible and of sufficient duration. The use of this command could be to signal that the end result of a program had been reached, or perhaps a point in a computation needed a parameter to be set on a register on the console switchboard; its use was up to the requirements of the programmer.

The sound produced by the speaker was probably the reason for it being known as the 'hooter', but it was not unpleasant, the digital pulses presented to the audio amplifier being altered by its circuit characteristics and fed to the speaker as a rounded sawtooth waveshape.

Some time prior to August 1951 Geoff Hill, who worked with Trevor Pearcey on the original design of the programming techniques for CSIRAC, saw the potential for music to be generated by exploiting the different tones produced by the speaker when different loops, which included the speaker command, were used in a program. The program, which was demonstrated some time in August 1951, has not survived but a 1953 version of it was located during the archiving process of CSIRAC records. Also found during the archiving was a music program written by Professor Tom Cherry during 1957 when he was associated with CSIRAC in its early days in Melbourne. I can remember being approached for details of the timing of CSIRAC and Professor Cherry's comprehension of it is amply demonstrated in his most elaborate version of a Music Program.

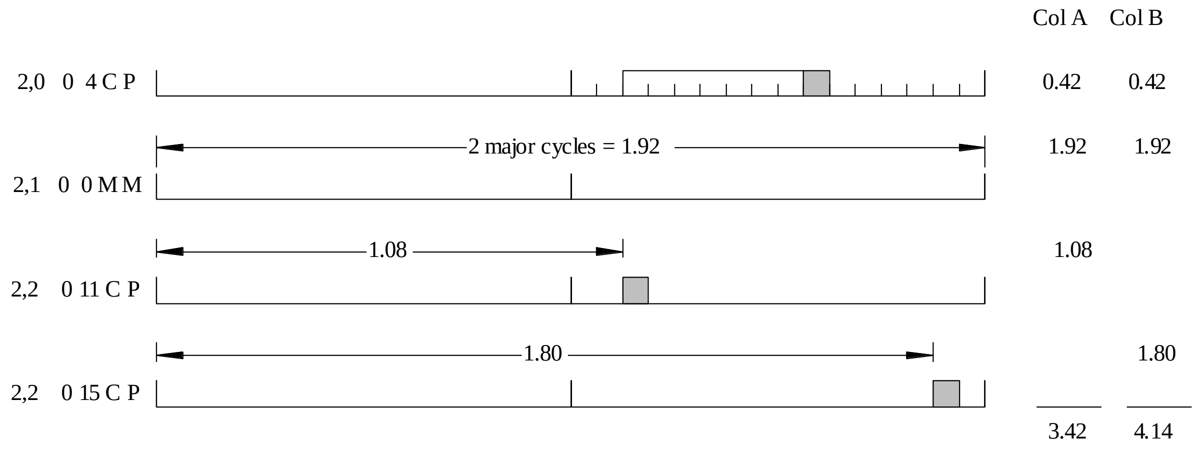
As mentioned earlier, if the speaker command is included in a loop along with one or more other commands to vary the timing, then when the loop is executed by the computer a tone is produced by the speaker. The tone, or pitch, is determined by the period of successive executions of the speaker command, and its duration depends on the number of times the loop is traversed. Designing a series of such loops producing different tones that were the same as, or in reality close to, those of the musical scale would form the basis of a program that could produce a music-like output from the speaker.

In designing a loop to produce a particular tone by adjusting the period between speaker commands, intervening commands with computer cycle-time variations were used.

The computer cycle consists of two sections: during the first, Part 1, the command is copied from its memory location to the Interpreter. This takes 1.92 milliseconds if the location is 12, 13 or 14 (modulo 16) or 0.96 milliseconds for the other thirteen locations. In Part 2, when the Interpreter is decoded for execution to take place, the value of the address within the command plays the same part as the memory location did in Part 1 in determining the time required to complete the computer cycle.

Thus command times of 1.92, 2.88 or 3.84 milliseconds could be achieved, along with their multiples or combinations, the minimum variation being one major cycle or 0.96 milliseconds. This only allows the design of about half a dozen tones that are near enough to the musical scale over the two and a half octave range that is possible with CSIRAC.

The period of the tone is actually the time between the start of the minor cycle when the digits from the source are sent to the speaker, at the execution time of one speaker command, and the start of the minor cycle time during the next speaker command when the source digits go to the speaker again. The address contained within a speaker command determines which minor cycle is chosen for this transfer of digits from source to speaker. By varying the address of the second of a pair of speaker commands the period can be varied by 0.06 millisecond steps up to 0.72 milliseconds as the execution time of the speaker command varies from its earliest possible time in the computer cycle to the latest possible time.



All figures represent time periods in milliseconds

Figure 8. Variation of period by altering the address within a speaker command

In Figure 8 the timing is shown for three commands that are in memory locations 64, 65 and 66; these memory addresses are usually written in the scale of 32, so the commands are in locations 2,0 to 2,2. In 2,0 the command 0 4 C P exists, which is interpreted as; send the contents of the word in source register C to the destination P (the speaker) at time position 4. In 2,1 the command 0 0 M M is a null command and just adds 1.92 milliseconds to the period. In 2,2 is another speaker command 0 11 C P that sends the contents of register C to the speaker at time position 11.

In the timing diagram of Figure 8 the minor cycle where execution takes place is shaded and the time segments in milliseconds are shown under Col A giving a total period of 3.42 milliseconds for the sequence described above. If we change the address within the command in memory location 2,2 so that it becomes 0 15 C P the minor cycle where execution takes place occurs later in the computer cycle and the period is increased to 4.14 milliseconds as shown under Col B.

Figure 9 shows a section from the 1957 Music Program by Professor Tom Cherry. It is an eight command loop that is entered when the note A3 is required during the program. This note on the Equal Tempered Scale has a frequency of 220 Hz, its period being 4.545 milliseconds. The note produced by this loop has a period of 4.56 milliseconds giving it a frequency of 219.3 Hz, and as it is a pure note it would sound in tune to most ears. The position within the memory where the loop is stored is from location 322 to location 329; in the 'scale of 32' that is from 10,2 to 10,9.

These memory locations are such that 'speed-up' occurs in the first half of all computer cycles in the loop, and the four speaker (P) commands have addresses that are different but all allow speed-up to occur during the second half of their computer cycles. All four speaker commands thus have two major cycle, or 1.92 millisecond, computer cycle times. Between each speaker command is a null type command except for the loop return command in 10,9 which returns control to 10,2. The number of times the loop is traversed is dependent on the value set in the lower half of the Sequence Register and is determined by the note duration required.

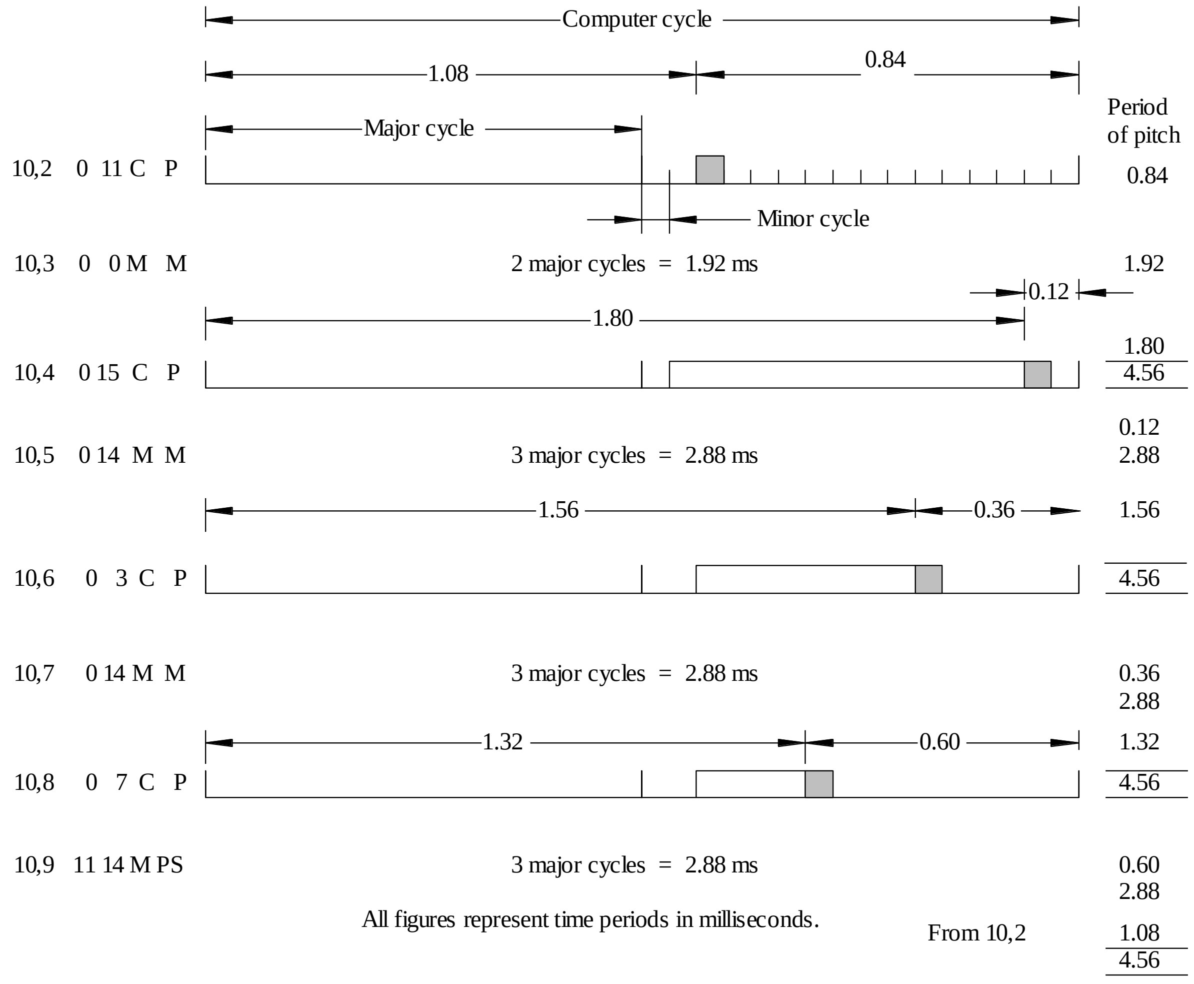


Figure 9. Using a loop to obtain desired period

The null commands and the loop return command have addresses that do not allow speed-up to occur, except for that in 10,3; so three have computer cycle times of 3 major cycles, 2.88 milliseconds, and 10,2 has a computer cycle time of 2 major cycles, 1.92 milliseconds. If we now consider the loop as four pairs of commands, each pair having a speaker command and a null command, then we get the first pair 10,2 and 10,3 that take a total time of 4 major cycles, giving a period of 3.84 milliseconds, and the other 3 pairs each taking 5 major cycles, giving a period of 4.80 milliseconds. If each speaker command had a zero address the note produced would be a mix of two frequencies, 208.3 Hz and 260.4 Hz, not close to what we want and most likely discordant.

The total time for the loop, however, is three periods of 4.80 milliseconds plus 3.84 milliseconds, a total of 18.24 milliseconds in which to produce four speaker pulses. By varying the address within each speaker command it is possible to decrease the longer periods of 4.84 milliseconds, and increase the shorter period of 3.84 milliseconds, so that there is a constant period of 4.56 milliseconds between the times the speaker receives a pulse. This produces a pure note of 219.3 Hz which is very close to the required note A3.

Figure 9 illustrates the effect changing the address within each speaker command has on the execution time that the digits go to the speaker. These times are shown by a shaded minor cycle period in the second half of the computer cycle time. Starting with the first command in 10,2 the digits from the C register go to the speaker at time position 11 and a time of 0.84 milliseconds elapses before the end of the computer cycle. The command in 10,3 takes 1.92 milliseconds for its computer cycle to take place, and in 10,4 the command is 1.80 milliseconds into the computer cycle when the digits again go to the speaker at time position 15. These three time periods are shown under the column headed ‘Period of pitch’ and their total is shown as 4.56 milliseconds. The effect that the address has can be followed through the succeeding commands, and for the fourth period total, after the 2.88 milliseconds of the command in 10,9 we include the 1.08 milliseconds elapsed time of the computer cycle of the command in 10,2 again producing a period of 4.56 milliseconds.

When the Music program was written in 1957 only the original speed-up (MODE 1) was available giving a minimum computer cycle of 1.92 ms. Perhaps this was fortunate in a way, as under MODE 2 conditions where the minimum cycle time was 0.9 ms with the promise of higher frequency tones being available, planning a loop to produce a desired tone would have been more complex. Although the timing relative to a C pulse was still the same, the start of a computer cycle now coincides with a P pulse rather than a C pulse. This results in loop returns and other non-sequential commands being more difficult to incorporate, and with the speaker addresses to achieve minimum cycle time limited to a choice of eight, rather than twelve, a music program may never have been finished if MODE 2 had been available.

R. Bowles
4.2.03