ELECTRONOTES ELECTRONIC MUSIC CLUB SUPPLEMENT

Supplement to Vol 2 #16 General Distribution

Prepared by Bernie Hutchins, Tom Wells, and Simon Benlevy

HYBRID VOLTAGE-CONTROLLED - DIGITAL ELECTRONIC MUSIC SYSTEMS CONSIDERATIONS

Report submitted to Prof. W. Ku, Elect. Eng. 4475 term project December 1972

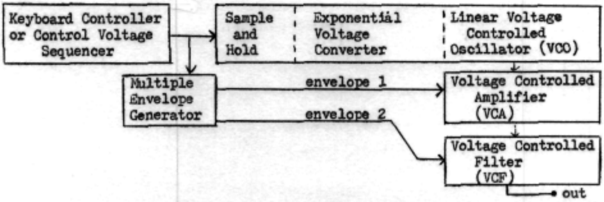
CONTENTS

INTRODUCTION: Electronic music systems since about 1965 have been mainly concerned with voltage controlled devices.[1] Recently, digital devices and the availability of digital IC's and MSI have made digital music systems practical,[2,3,4] and even some LSI is being used in organ designs.[5,6] Use of digital systems along with the present voltage controlled synthesizers is being considered.

This report will consider some aspects of voltage controlled systems, and interfaces with digital systems. It is intended that an overall electronic music system be presented, and toward this end, it will be mainly voltage controlled, i.e. generally analog, since the digital counterparts of various electronic music modules are not developed yet, or are impractical. Where analog modules are used, the corresponding digital approach is often presented as an appendix to this report.

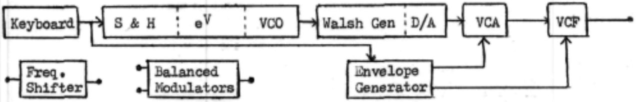
Digital IC's have been used for various purposes in electronic music. Digital controllers known as sequencers or tune computers have been used to provide a slow series of control voltages representing different musical notes in sequence.[7,8] Various methods of waveform generation have been used, mainly on a segment-by-segment basis, and will be mentioned at the beginning of part 4 of this report. The real tour-de-force of digital methods in electronic music has been in the generation of appropriate musical scales by digital countdown of one master oscillator, e.g. a twelve tone equally tempered scale by dividing one upper frequency by numbers in the range of 100-300, these numbers approximating the ratio $1{:}2^{1/12}$. These methods will be discussed, but not used. The main digital method we will discuss here is the synthesis of waveforms by Walsh-Fourier methods.

Conventional Voltage Controlled System: A conventional V.C. electronic music system in indicated below in its most basic form:



The VCO must have exponential response to the control voltage in order that equal musical intervals are generated for equal changes of control voltage. Recall that each octave of an equally tempered scale is 12 notes, equally spaced, and more fundamentially, each octave is twice the frequency of the one below, hence the need for an exponential rather than a linear voltage to frequency relation. This permits two exponentially controlled VCO's to track each other at equal intervals (equal frequency ratios), and also, resistors in the voltage divider string in a keyboard can all be the same value. An exception is made for the generation of musical timbre by means of FM sidebands[9]. The purpose of the sample-and-hold circuit is to provide a control voltage for the VCO during the time that the sound envelope is intended to decay. For example, after the key on a keyboard controller is lifted, we want a finite, non-zero decay time during which the pitch should not change. Various types of voltage envelopes are available, and can be supplied to a VCA for amplitued shaping, or to a VCF to control overtone content with time. This latter time dependent control over musical timbre is thought to be fundamental to the "character" of the synthesized sounds, i.e., it makes the sound musical.

PART 1: Discussion of the overall system: The system shown below is a slight expansion of the basic V.C. system described above. We have added a Walsh function generator between the VCO and the VCA. This provides the waveshaping. Also included are some additional balanced modulators, and additional equipment for a frequency shifting device of simple design:



The basic source of signal is the VCO, since the VCO is the most generally useful for musical applications at this time. This is because it is more generally tunable, and the addition of a voltage sample-and-hold permits the storage of "frequency information" by the stored voltage. The VCO we are presently using is shown in Fig. 1, pg. 3 of Electronotes Newsletter #15 (EN#15), Nov. 30, 1972. Other VCO's perhaps more suitable are given in EN#13, pg. 5 (Oct. 10, 1972) and in EN#16, sect. 4a (to be published). The main feature we require is that the VCO can drive TTL, and that it have range well above the audio, since it must be counted down by a factor of 32 by the Walsh function generator. A discussion of the attractive alternative of scale generation by digital means is discussed in Appendix A. The exponential converter is shown in fig. 4, pg. 3 of EN#15, and its operation is described in Appendix B of this report. The sample-and-hold circuit and the envelope generator are shown in fig. 6 of EN#15, pg. 4. Digital methods of envelope generation and gain control are discussed in Appendix C.

The basic VCA circuit is simply a analog multiplier using a single chip IC. This module is described in EN#12, pg. 4, Sept. 20, 1972, and a discussion is also given in Appendix D, which also discusses a D.C. version that is used for control purposes in the VCO and VCF. The multiplier module serves as a VCA, but also works as a balanced modulator, a frequency doubler, and for amplitude modulation as desired.

The three remaining modules will be discussed separately. The VCF, fig. 2, combined with a quadrature VCO, fig 3 of EN#15, pg. 3 is described in part 2 below. The frequency shifting interconnection is described in Part 3 below, and the Walsh generation process is described in part 4.

PART 2, The VCF/VCO Module: First of all, it is appropriate to state why we are considering the use of a filter in a system we are orienting toward additive rather than subtractive synthesis. That is, we intend to build a waveform from harmonics (Walsh harmonics in this case) rather than start with a waveform rich in harmonics and then filter it down to what we want (subtractive synthesis). However, the additive synthesis by Walsh functions is less familiar, and hence standard filtering of the waveforms generated can be used in the standard way. In a true Walsh-synthesis process, each Walsh function output to the D/A converter would be controlled with time, and this would necessairily be more involved than the VCF, and thus the additive synthesis is to be thought of as more of a preset and more permanent process, while the use of the VCF might provide for more rapid changes during live performance for example. Finally, the VCF could be used directly with the VCO in the standard manner, or used as a more general filter for processing live sounds, etc.

The VCF circuit is shown in Fig. 2 of EN#15. It is of the State-Variable type, and the operation of this filter is described simply by D. Rossum in part 8a of the same issue, and other references are given on page 6 of that issue. The main interest in this filter is its stability and the simultaneous High-Pass, Band-Pass, and Low-pass outputs.

The filter is voltage controlled by two D.C. mulitpliers (See Appendix D) and a positive control voltage controls the filter functions in a linear manner. The control mechanism is obvious once we consider that the summing junctions (the inverting inputs of the op-amps) are at virtual ground. Current passing through the resistors (R_y) charges the capacitors, since the virtual ground passes no current to ground, but only sits at ground potential. Thus, center frequency is controled by the rate of integration, which is determined by the current through R_y, and this is of course determined only by the voltage output of the multipliers measured above ground.

The circuit becomes a VCO upon application of a negative control voltage. Simply stated, this is because the multipliers become inverters as well as just multipliers, and the Low-Pass output is inverted twice with respect to the summer, hence remains the same, but the Band-Pass output is inverted only once with respect to the summer, and thus furnishes positive feedback controlled by the "Q" control. The voltages on opposite sides of the integrators are 90° out of phase as expected. Use of this circuit as an oscillator is described elsewhere,[10] and a further discussion of the use of two integrators as an oscillator is considered in Appendix E.

PART 3. The Frequency Shifter: Balanced modulators have been used in electronic music for years to produce the so called "Ring Modulator" effect, the word "ring" refers to the ring of four diodes employed in early versions of the device rather than to any characteristic scunds produced. The essential thing about balanced modulation is the production of two sidebands, the sum and difference of the frequencies applied to the inputs. Today, the balanced modulator is being implemented with IC versions of four-quadrant multipliers. These same modules are used as VCA's, etc., as described above, and two of them are used along with the VCF/VCO unit of Part 2 for the frequency shifter. All that remains is an accurate 90° phase shifting network and a couple of summing networks to implement a frequency shifter interconnection.

While the balanced modulators produce a "double sideband" signal, the frequency shifter is a "single-sideband" device, and the present device is capable of separate upper sideband and lower sideband production, i.e., the separated sum and difference frequencies. The frequency to be shifted (i.e., the input signal) is first applied to a phase shifting network, and thus we can think of it as being divided into a sine and a cosine representation, since the phase shifter shifts 90°. Furthermore, the VCF/VCO unit described in Part 2 provides a sine and a cosine representation of the shifting signal. By putting the two sines into one multiplier, and the two cosines into a second multiplier, the normal sum and difference frequencies appear, but there is a difference of sign between either the sum or the difference signals in the output of one of the multipliers relative to the other. Hence, they are easily separated by adding or subtracting the two signals from the multiplier outputs. The actual case is easy to work out from trigonometry.[11,12] A discussion of 90° phase shifting networks is given in Appendix F, and the actual frequency shifter constructed is shown in EN#51, pg 4.

It is fair to say that the device constructed works about 90%. In test setups, oscilloscope traces showed that the sum and differences were clearly separated, but not without some "noise" which was probably the other sideband getting through, etc. This was about 10% of the desired signal. This could probably be improved by construction of better 90° phase-shifters, better balance of the multipliers, and more accurate construction of the summers used to separate the signals. Many 10% components were used, and this is not really good enough. While this 10% error is perhaps not first rate engineering, the unit does work well as a special effects device, cost only about 2% of what a commercially made unit does, and can be broken down into other useful modules when not being used to shift frequencies. The effect is most dramatic with the human voice, where a "Donald Duck" effect is achieved, achieving increasing loss of intelligibility with increasing shift. However, use of downshift is limited, since the voice can only be shifted just so far down before the apparent pitch is so low that it cannot be reproduced by hi-fi amplifiers, or is below the audio range. Crossing the downshift below zero results in an upshift.

Part 4: WAVEFORM SYNTHESIS BY MEANS OF WALSH FUNCTIONS:

a) Introduction: Several methods of digital waveform synthesis have been considered elsewhere. One method is to use a "Bucket Brigade" type of counter, and have a voltage "walk" down the line of n segments, each segment being adjusted to a desired output voltage level by its own pot.[13] A second method is to use commercially available read-only-memories (ROM's) to produce a stored waveform.[14] Thirdly, transversal digital filters have been used for synthesizing sine waves.[15] These three methods are basically segment by segment methods.

A second approach is to use various basis sets of functions and generate a desired waveform by appropriate superposition. Generation of strings of digital harmonics (up to 10 or more) have been suggested by counting down one upper frequency using divide-by-n type circuits.[16,17] This is easy to do with digital IC's, but outputs are square waves or non-symmetric rectangular waves. Assurance of square-wave symmetric outputs can be achieved by doubling the upper frequency and adding divide-by-2 circuits to each of the lower harmonic outputs we want to use. None the less, we really don't want to attempt a Fourier synthesis process using square waves which have substantial harmonic content, even if all the harmonics are available. Efforts to round the square waves into sines (e.g. by using a low-pass filter) have been used, but this works over only a relatively small range of frequencies. Still another approach is offered by the use of shift registers to generate pseudo-noise sequences, i.e., an overall output waveform is derived from the outputs of a shift register and weighted as desired to give a histogram type output.[18] Any and all of the above could be used for useful musical purposes.

The method we are describing here is believed to be general and the simplest from the point of view of hardwear. The complete, orthogonal set of rectangular "Walsh Functions" is employed as a basis set of waveforms from which we can synthesize all other periodic waveforms by a process which exactly parallels Fourier synthesis using sines and cosines. Mathematicians have assured us that the Walsh Functions possess the necessary properties for a corresponding Walsh-Fourier Synthesis. A list of references on Walsh functions is given in Appendix G. Below, a consistant approach is offered to the problem: 1) A general discussion of properties, 2) A computer program, and 3) a TTL hardwear realization all follow the same basic procedure.

b) Discussion of Walsh Functions: Walsh functions as described below are given a single index n, and denoted Wal(n). Properly, they should have a running variable (say t for time) and be denoted Wal(n,t), but this will generally be neglected. Wal(0) is simply a D.C. offset level, and won't enter our calculations of A.C. functions. $Wal(2^i-1)$ is a symmetric square wave, i being a positive integer. This square wave repeats $2^i/2 = 2^{i-1}$ times over the basic interval chosen for the entire set. The rest of the Walsh functions are generated by the recursion relation:

$$Wal(h) \cdot Wal(k) = Wal(h \oplus k)$$

where the $\oplus$ sign indicates "Modulo-2" addition, which means 1+0=1, 0+1=1, 0+0=0, and 1+1=0, and the h and k indices are represented as binary numbers for this addition. Perhaps EXclusive-OR of the two function indices represented as binary numbers is the clearest. A couple of examples will help:

Example 1: Generating a new function:

5 = 0101 in binary notation
15 = 1111 in binary notation
1010 = 10 in decimal notation

Thus: Wal(10) = Wal(5)·Wal(15)

Example 2: Regeneration of Wal(3) square

7 = 0111 in binary notation
4 = 0100 in binary notation
0011 = 3 in decimal notation

Thus: Wal(3) = Wal(7)·Wal(4)

The odd index Walsh functions are also denoted by $Sal(j) = Wal\ (2j-1)$ and the even Walsh functions by $Cal(j) = Wal(2j)$. The motivation behind this notation is the similarity of the Sal and Cal functions with the Sine and Cosine functions, and there are many valid analogs, e.g. the main difference between corresponding Sal and Cal functions is phase. A complete basis of functions could therefore be based on Sal or Cal alone, but we shall stick to the whole set as all are generally required for the complete generation of the Walsh functions.

Another property of the Walsh functions that is an analog of the sine-cosine functions is the terminology of Sequency, analogous to Frequency. Sequency is defined as one-half the total number of zero-crossings per second (ZPS) and since the Walsh functions are irregular within their basic period, the concept of frequency has no meaning.

Some thought will show the following:[19] Square waves generally have + and - values about zero, while logic levels are defined as 1 and 0. A level shift process could be used to convert these logic levels, but is not necessary. Just define the + excursion as logic 1 and the - excursion as 0. Note that this also greatly simplifies the multiplication process indicated in the recursion relation, since it reduces to the exclusive-or operation! The fact that the multiplication reduces to EX-OR is fundamental to what we do next. We have the necessary properties and can now generate the functions, after considering the basic period of the set.

The basic period of the set of Walsh functions is the period of Wal(1), and is made the same as the period of f(t), where f(t) is the periodic waveform we are going to synthesize. We divide this basic interval into 2^m segments, where m is as large as we need for any desired degree of approximation of f(t). We illustrate with m=3, i.e. a basic interval divided into $2^m=8$ segments.

A matrix notation is useful and will be used below. The rows represent the Walsh functions, and the columns represent the segments of the basic interval. First put in the Wal(0) d.c. offset, and square waves in the 2^i-1 positions, i=1,2,3 (m=3)

Wal(0)	11111111
Wal(1)	11110000
Wal(2)	-?-
Wal(3)	11001100
Wal(4)	-?-
Wal(5)	-?-
Wal(6)	-?-
Wal(7)	10101010

Now the rest can be generated with the recursion relation:

Wal(2) = Wal(3)·Wal(1) since $001 \oplus 011 = 010$
Wal(4) = Wal(7)·Wal(3) since $111 \oplus 011 = 100$
Wal(5) = Wal(7)·Wal(2) since $111 \oplus 010 = 101$
Wal(6) = Wal(7)·Wal(1) since $111 \oplus 001 = 110$

The recursion process is probably clear by now without need of the binary index procedure.

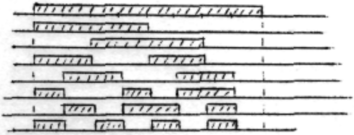
Replacing multiplication by EXclusive-Oring as described above, Wal(2) is generated as indicated:

Wal(1)	11110000
Wal(3)	11001100
Wal(2)	00111100

Likewise, the matrix is filled:

Wal(0)	11111111	D.C.
Wal(1)	11110000	Square Wave
Wal(2)	00111100	EX-OR of 1 and 3
Wal(3)	11001100	Square Wave
Wal(4)	01100110	EX-OR of 7 and 3
Wal(5)	10010110	EX-OR of 7 and 2
Wal(6)	01011010	EX-OR of 7 and 1
Wal(7)	10101010	Square Wave

Or Drawn as Waveforms instead of matrix:



c) Computer Generation of Matrix:

Obviously, filling in the matrix is great fun - for about m=3, but after that, it gets tiresome. Fortunately, a computer can do it, and you could tell a computer how in about 30 second in English, but first we have to develope an algorithm for the process, and convert it into a computer language. An algorithm that we have used is shown below:

STEP 1: Define m, define dimension of matrix as 2^m, Elements W(,)

STEP 2: Generation of Square Waves in positions 2^i-1

> Repeat for k = 1,2,3,.....m
> - Define $L = 2^{m-k}$
>   - Repeat for r = 1,2,3, ... 2^m
>   - Let $p = r/L - 1/2^m$
>   - If (Highest Integer in p)/2 = An Integer
>   - Then $W(2^k-1, r) = 1$
>   - Otherwise $W(2^k-1, r) = 0$
>
> END this Step

STEP 3: Filling in the Matrix

> Repeat for k = 1,2,3,.....m-1
> - Define $L = 2^{k+1} - 2^k - 1$
>   - Repeat for q = 1,2,....L
>   - Repeat for r = 1,2,3,..... 2^m
>   - Let $W(2^{k+1}-1-q, r) = W(2^{k+1}-1, r) \oplus W(q,r)$
>
> END this step

The actual computer program written in PL/C, a Cornell Univ. version of PL/I is shown in Appendix H. PL/C will run on a PL/I complier. You can generate a Walsh Function matrix by writing the above algorithm in your favorite computer language, or use a different algorithm. The computer generated matrix for m= 5 is shown at the right →

Wal(0)	11111111111111111111111111111111
Wal(1)	11111111111111110000000000000000
Wal(2)	00000000111111111111111100000000
Wal(3)	11111111000000001111111100000000
Wal(4)	00001111111100000000111111110000
Wal(5)	11110000000011110000111111110000
Wal(6)	00001111000011111111000011110000
Wal(7)	11110000111100001111000011110000
Wal(8)	00111100001111000011110000111100
Wal(9)	11000011110000110011110000111100
Wal(10)	00111100110000111100001100111100
Wal(11)	11000011001111001100001100111100
Wal(12)	00110011110011000011001111001100
Wal(13)	11001100001100110011001111001100
Wal(14)	00110011001100111100110011001100
Wal(15)	11001100110011001100110011001100
Wal(16)	01100110011001100110011001100110
Wal(17)	10011001100110010110011001100110
Wal(18)	01100110100110011001100101100110
Wal(19)	10011001011001101001100101100110
Wal(20)	01101001100101100110100110010110
Wal(21)	10010110011010010110100110010110
Wal(22)	01101001011010011001011010010110
Wal(23)	10010110100101101001011010010110
Wal(24)	01011010010110100101101001011010
Wal(25)	10100101101001010101101001011010
Wal(26)	01011010101001011010010101011010
Wal(27)	10100101010110101010010101011010
Wal(28)	01010101101010100101010110101010
Wal(29)	10101010010101010101010110101010
Wal(30)	01010101010101011010101010101010
Wal(31)	10101010101010101010101010101010

d) <u>WALSH-Fourier Synthesis</u>: Once the Walsh functions are generated, we ask what we can do with them besides listen to them individually (Which is interesting in itself). Clearly, we want to make use of the complete orthogonal set property and use a "Walsh-Fourier" process completely analogous to the Fourier synthesis process.[20] Denoting the function we want to synthesize by f(t), the <u>Fourier series</u> is:

$$f(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t))$$

$$\text{where } a_n = \frac{2}{T}\int_0^T f(t)\cos(n\omega_0 t)\,dt \quad \text{and} \quad b_n = \frac{2}{T}\int_0^T f(t)\sin(n\omega_0 t)\,dt$$

where T is overall period, t is running variable, and $\omega_0 = \frac{\pi}{T}$

The <u>WALSH-Fourier series</u> is:[21]

$$f(t) = A_0 + \sum_{n=1}^{\infty} (A_n\,\mathrm{cal}(n,\theta) + B_n\,\mathrm{sal}(n,\theta))$$

$$\text{where } A_0 = \frac{1}{T}\int_0^T f(t)\,\mathrm{wal}(0,\theta)\,dt$$

$$A_n = \frac{1}{T}\int_0^T f(t)\,\mathrm{cal}(n,\theta)\,dt \quad \text{and} \quad B_n = \frac{1}{T}\int_0^T f(t)\,\mathrm{sal}(n,\theta)\,dt$$

where T is overall period, t is running variable, and $\theta = t/T$

We will ignore the distinction between sal and cal, set T=1, and simplify the series to:

$$F(x) = \sum_{n=0}^{\infty} C_n\,\mathrm{Wal}(n,x) \qquad \text{where } C_n = \int_0^1 F(x)\,\mathrm{wal}(n,x)\,dx$$

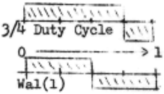
Note that mathematically this process is much simpler than standard Fourier series since Wal(n,x) takes on only the values +1 and -1 (or +1 and 0) and thus this only breaks up the interval of integration. Thus we only have to integrate F(x), and not the function times sine or cosine. Therefore, we can calculate the C_n for any F(x) that we know how to integrate.

Let's examine some of the waveforms we can synthesize from the Walsh functions We note that to an extent, when calculating Walsh-Fourier coefficients (C_n), we can ignore level shifts, inversions of Walsh-Functions, and the absolute value of the coefficients. As long as we keep in mind what we are doing, we can take steps to keep the math as simple as possible, and end up with the signs and ratios of the various coefficients. We shall always give the coefficients C_n consistant with the TTL realization we have in mind, although we may at times work with coefficients c_n as dummy coefficients.

First of all, we have the Walsh functions themselves available, and this includes the usual square waves. A second group of waveforms we can easily get are pulses of various duty cycle. In such cases, the integration is simply the summation of signed areas. We illustrate first for the pulse of 3/4 duty cycle, calculated for equal excursions about zero:

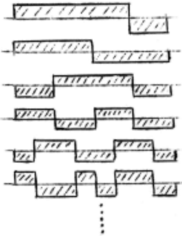
$$c_1 = \int_0^1 F(x)\mathrm{Wal}(1,x)dx = \int_0^{\frac{1}{2}} 1\cdot 1\,dx + \int_{\frac{1}{2}}^{3/4} 1\cdot(-1)\,dx + \int_{3/4}^{1} (-1)\cdot(-1)\,dx$$

$$= x\Big]_0^{\frac{1}{2}} - x\Big]_{\frac{1}{2}}^{3/4} + x\Big]_{3/4}^{1} = \tfrac{1}{2}-0-3/4+\tfrac{1}{2}+1-3/4 = \tfrac{1}{2}$$



this is done by straight integration, but can be done by signed areas (see next page)

For signed areas, multiply F(x) times Wal(n) and multiply by 1/8 for each of the 8 segments. We do the calculation for the 3/4 duty cycle pulse completely:



Wal(1) $C_1 = 1/8 + 1/8 + 1/8 + 1/8 - 1/8 - 1/8 + 1/8 + 1/8 = \frac{1}{2}$

Wal(2) $C_2 = -1/8 - 1/8 + 1/8 + 1/8 + 1/8 + 1/8 + 1/8 + 1/8 = \frac{1}{2}$

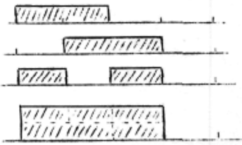
Wal(3) $C_3 = 1/8 + 1/8 - 1/8 - 1/8 + 1/8 + 1/8 + 1/8 + 1/8 = \frac{1}{2}$

Wal(4) $C_4 = -1/8 + 1/8 + 1/8 - 1/8 - 1/8 + 1/8 - 1/8 + 1/8 = 0$

Wal(5) $C_5 = 1/8 - 1/8 - 1/8 + 1/8 - 1/8 + 1/8 - 1/8 + 1/8 = 0$

$C_6, C_7, \ldots\ldots = 0$

We can now reconstruct this for the o and 1 digital levels:



$\frac{1}{2}$ Wal(1)

$\frac{1}{2}$ Wal(2)

$\frac{1}{2}$ Wal(3)

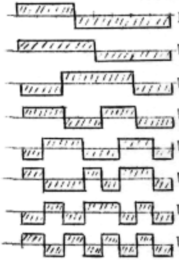
Sum of above

<u>3/4 Duty Cycle Pulse</u>

Note that we only <u>had</u> to go to to m=2 in this case. In general, we would not expect to require division of the basic interval into a greater number so segments than the denomenator of the duty cycle

Repeating this for the <u>$\frac{1}{2}$ duty cycle pulse</u> gives: $C_1 = \frac{1}{2}$, $C_2 = -\frac{1}{2}$, and $C_3 = \frac{1}{2}$

For a <u>3/8 duty cycle pulse</u>



Wal(1) $C_1 = 1/8 + 1/8 + 1/8 - 1/8 + 1/8 + 1/8 + 1/8 + 1/8 = 3/4$

Wal(2) $C_2 = -1/8 - 1/8 + 1/8 - 1/8 - 1/8 - 1/8 + 1/8 + 1/8 = -\frac{1}{4}$

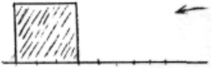
Wal(3) $C_3 = 1/8 + 1/8 - 1/8 + 1/8 - 1/8 - 1/8 + 1/8 + 1/8 = \frac{1}{4}$

Wal(4) $C_4 = -1/8 + 1/8 + 1/8 + 1/8 + 1/8 - 1/8 - 1/8 + 1/8 = \frac{1}{4}$

Wal(5) $C_5 = 1/8 - 1/8 - 1/8 - 1/8 + 1/8 - 1/8 - 1/8 + 1/8 = -\frac{1}{4}$

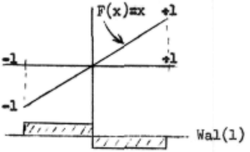
Wal(6) $C_6 = -1/8 - 1/8 - 1/8 - 1/8 - 1/8 - 1/8 - 1/8 - 1/8 = -\frac{1}{4}$

Wal(7) $C_7 = 1/8 - 1/8 + 1/8 + 1/8 - 1/8 + 1/8 - 1/8 + 1/8 = \frac{1}{4}$



Reconstruction at digital 1 and 0 levels. Other pulses are calculated in the same way. However, something like a 1/3 duty cycle presents a problem, we have to settle for $\frac{1}{4}$ or 3/8, or use more Walsh functions to get 5/16, 11/32, etc. This is very similar to the problem we run into when trying to generate a square wave with a finite number of sine waves.

Another group of functions of interest are the linear ones, in particular the sawtooth and the triangle wave. The sawtooth is considered first, and for mathematical convenience, we switch the basic interval from 0 to 1 to the interval -1 to 1.

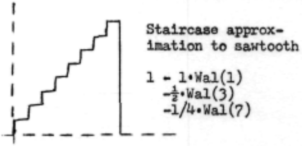


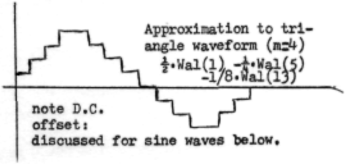
$$c_1 = \int_{-1}^{1} F(x)\cdot \mathrm{Wal}(1,x)\,dx = \int_{-1}^{1} x\cdot \mathrm{Wal}(1,x)\,dx$$

$$= \int_{-1}^{0} x\,dx - \int_{0}^{1} x\,dx = \frac{x^2}{2}\Big]_{-1}^{0} - \frac{x^2}{2}\Big]_{0}^{1}$$

$$= 0 - \tfrac{1}{2} - \tfrac{1}{2} + 0 = -1$$

Likewise, one finds that $c_2 = 0$, $c_3 = -\frac{1}{2}$, $c_4 = 0$, $c_5 = 0$, $c_6 = 0$, $c_7 = -1/4$ etc. i.e., we use the square waves, and each time we double the frequency (square waves still have frequency), we halve the amplitude of the coefficient. This is a fairly well known method of generating a "Staircase Wave", but not as a consequence of Walsh Fourier synthesis. Converting back to the basic interval 0 to 1, we have to cut the c_n in half to get the C_n, and in this case, we will level shift by +1 to get a conventional looking staircase, approximation the sawtooth. Similar calculations for the triangle wave give $C_1 = \frac{1}{2}$, $C_2 = 0$, $C_3 = 0$, $C_4 = 0$, $C_5 = -\frac{1}{4}$, C_6 to $C_{12} = 0$ $C_{13} = -1/8$ and in general, $C_{2n-3} = 1/(2^{n-2})$. Note that for these functions, unlike the first pulses we considered the synthesized waveforms are evidently approximations to the desired waveform, and it is clear that we can make this approximation better by adding more Walsh functions, i.e., by dividing the basic interval into more segments by increasing m.





One waveform of fundamental interest is of course the sine wave. Here we must alter the basic interval to represent 0-2π but if we consider the basic interval to be in units of pi, we can still use 0-1. For example, the first coefficient is calculated as: (for a peak to peak amplitude of 1 unit)

$$c_1 = \int_0^1 \mathrm{Wal}(1,x)f(x)\,dx = \int_0^{\frac{1}{2}} (\tfrac{1}{2})\sin(2\pi x)\,dx - \int_{\frac{1}{2}}^{1} (\tfrac{1}{2})\sin(2\pi x)\,dx$$

$$= \frac{1}{4\pi}\int_0^{\frac{1}{2}} \sin(2\pi x)\,dx - \frac{1}{4\pi}\int_{\frac{1}{2}}^{1} \sin(2\pi x)\,dx$$

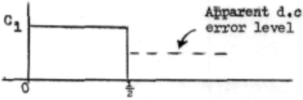
$$= \frac{1}{4\pi}\Big[-\cos(2\pi x)\Big]_0^{\frac{1}{2}} - \frac{1}{4\pi}\Big[-\cos(2\pi x)\Big]_{\frac{1}{2}}^{1} = 1/\pi = 0.318$$

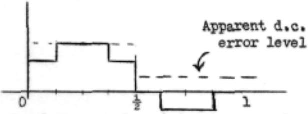
In the same way, we can break up the interval for every zero crossing of the Walsh functions. The integral of $\sin(2\pi x)$ is of course $-\cos(2\pi x)$ in all cases. One finds for the first 8 Walsh functions (Wal(0) to Wal(7)) that all c_n are zero except c_1 above and $c_5 = -0.138$. These first two are plotted on the next page.

Sinewave approximation for m=3
0.318 Wal(1) - 0.138 Wal(5)
plotted for equal excursions of
the Walsh functions about zero.

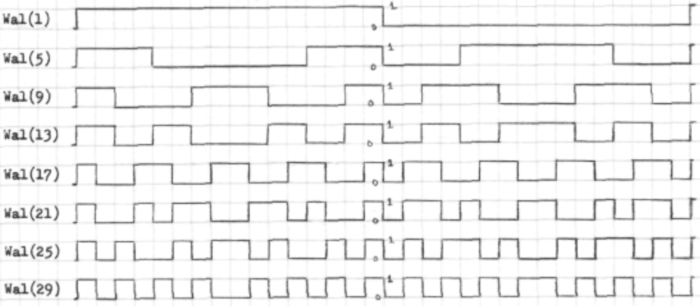
Here, as in the case of the sawtooth and triangle, it becomes apparent that we must subdivide the basic interval into more segments to get a good enough approximation to the waveform we want. The math again gets too tedious, and we must resort to the computer. The process the computer must follow is exactly the same as for the hand calculations on the -cos values. However, generally it is necessary to carry out the integration over all 32 intervals (for m=5) rather than just at the zero crossings. Of course, the limits of integration often cancel in such a process, and a small accumulated error may build up, but this generally is not a problem. The computer program used is in appendix H, and the results are plotted on the next page. Note that in the computer program, the one and zero levels were used, i.e., the matrix elements calculated before. Thus, the amplitude of the sine wave was made 1 instead of $\frac{1}{2}$ as in the hand calculations. This gives results for 1 unit peak to peak, when generated with equal excursion Walsh functions. With the 0 and 1 digital levels, we must again double the coefficienst as indicated on the next page. In any event, it is the ratios that are important

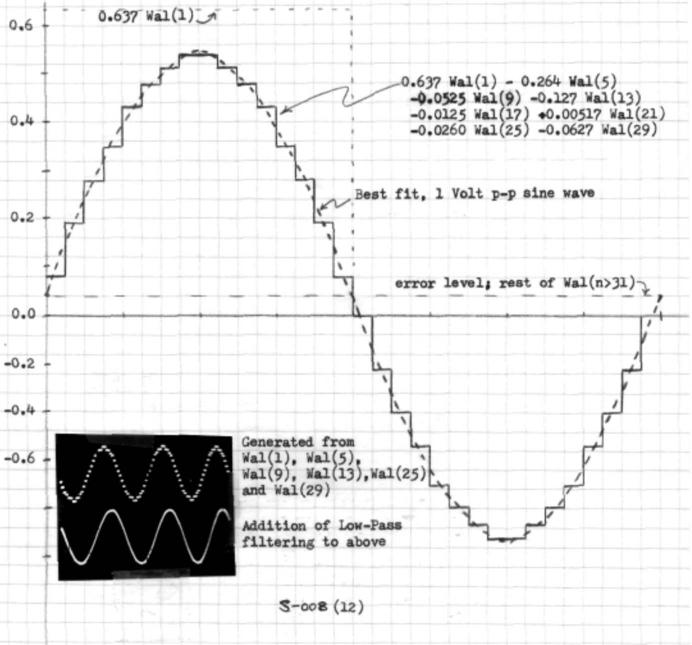
The actual formation of the waveform with the addition of subsequent Walsh components is interesting. Several features should be mentioned. (1) In Fourier sine wave synthesis, each additional component seems to make the waveform better, but some Walsh-Fourier components seem to make it worse. However, appearance of the next component show that the hole dug out by the former was to make allowance for a hill on the latter. Note this is just a comment on appearances, and does not necessairily say anything about mathematical convergence. (2) For a balanced waveform, such as the sinewave, the coefficient of Wal(0) <u>is</u> 0, as it should be, since there is no D.C. offset. However, we see that the first component of the sine wave is positive, and builds up a large "block" on the left side (0 to $\frac{1}{2}$). Thus this has an apparent D.C. error of $\frac{1}{2}C_1$. It remains for the rest of the Walsh components to chip away at the corners of the block, and dig a "pit" on the right side, and this shifts the apparent D.C. error down as indicated below:





A glance at the next page shows that the error level is very low, but not zero yet, and we would have to use an infinite number of Walsh functions to make it zero. Thus, truncating the series results in an apparent D.C. error level, which we can easily correct by adding some Wal(0) to the waveform, but this is not the same as an actual Walsh component. The sine wave initially refuses any part of Wal(0) but then when it sees it is not going to get all the others, it goes back and asks for a D.C. component which can be <u>supplied</u> by Wal(0). The distinction is mainly academic, as as we said, level shifts tend to be somewhat arbitrary anyway. Also, this level shift problem generalizes somewhat for the generation of higher harmonics, and one must be careful to remember that these error levels do come in when the series is truncated.





Generated from Wal(1), Wal(5), Wal(9), Wal(13),Wal(25) and Wal(29)

Addition of Low-Pass filtering to above

e) Fourier To Walsh Fourier--Higher Sinewave Harmonics:

Finally, we are also interested in the generation of higher sinewave harmonics, not so much for the purpose of generating them. but for use in a Fourier to Walsh-Fourier transition. The 2nd, 4th, and 8th sinewave harmonics, etc. are easy to generate using the results for the sinewave. We can see this by observing that each of the Walsh functions eventually occurs at twice the sequency (ZPS) and thus we have only to look for the same waveform at double the sequency, assign it the appropriate sinewave Walsh coefficient, and this will result in a sinewave of twice the frequency. The process becomes simple when we observe that the first Walsh function appearing in each sine wave is just the square wave of the same frequency. None the less, the "density" of Walsh functions increases as the square, so the relative spacing of required Walsh functions must increase. Recall the sine itself required every fourth harmonic, starting with one. The 2nd harmonic requires every eighth harmonic, starting with three. The fourth sinewave harmonic requires every 16th Walsh harmonic, starting with Wal(7), etc.

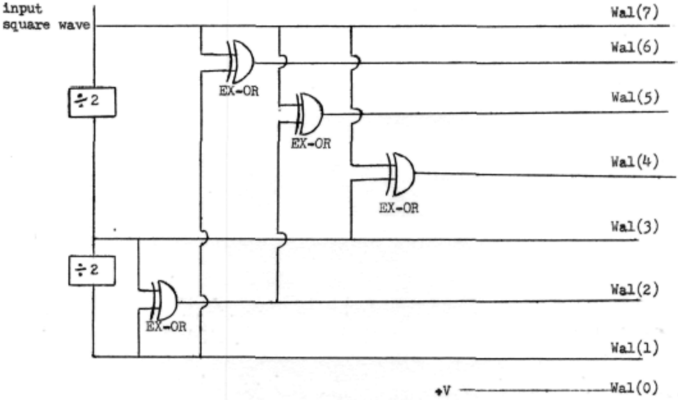
The odd numbered harmonics such as the third sinewave present more of a problem since basically we are trying to generate six bumps using 4 or 8, etc. They are however, like the first sinewave, antisymmetric about the midpoint, and can be built up in the same way using the sal(n) Walsh functions. The results for the first 8 sinewave harmonics, as generated by the first 32 Walsh functions are shown below. The coefficients of the even index Walsh Functions are all zero, and are not included.

SINEWAVE HARMONICS

		First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth
Wal(1)	C_1	0.637	0.0	0.212	0.0	0.126	0.0	0.090	0.0
Wal(3)	C_3	0.0	0.637	0.0	0.0	0.0	0.212	0.0	0.0
Wal(5)	C_5	-0.264	0.0	0.512	0.0	0.307	0.0	-0.036	0.0
Wal(7)	C_7	0.0	0.0	0.0	0.637	0.0	0.0	0.0	0.0
Wal(9)	C_9	-0.0525	0.0	-0.342	0.0	0.46	0.0	0.188	0.0
Wal(11)	C_{11}	0.0	-0.264	0.0	0.0	0.0	0.512	0.0	0.0
Wal(13)	C_{13}	-0.127	0.0	0.14	0.0	-0.190	0.0	0.456	0.0
Wal(15)	C_{15}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.637
Wal(17)	C_{17}	-0.0125	0.0	-0.042	0.0	-0.100	0.0	-0.375	0.0
Wal(19)	C_{19}	0.0	-0.0525	0.0	0.0	0.0	-0.342	0.0	0.0
Wal(21)	C_{21}	+0.00517	0.0	-0.102	0.0	-0.246	0.0	0.154	0.0
Wal(23)	C_{23}	0.0	0.0	0.0	-0.264	0.0	0.0	0.0	0.0
Wal(25)	C_{25}	-0.0260	0.0	-0.154	0.0	0.164	0.0	0.030	0.0
Wal(27)	C_{27}	0.0	-0.127	0.0	0.0	0.0	0.140	0.0	0.0
Wal(29)	C_{29}	-0.0627	0.0	0.064	0.0	-0.068	0.0	0.074	0.0
Wal(31)	C_{31}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The above arrangement of numbers can be considered as a conversion matrix if you like to work with such things, the Walsh coefficients being a row vector, and the Fourier coefficients being a coulmn vector in such a case. In practice, suppose you have read in a paper on musical acoustics that the overtone series of instrument X is one times the fundamental and $\frac{1}{2}$ times the second harmonic. You would multiply the first column of coefficients by one, the second by $\frac{1}{2}$, and add all the results for a given C_n to get an overall C_n. This additional conversion step is of course a bother, but equipment advantages may make it more desirable to use Walsh harmonics. Except for the vast amount of published data on Fourier sinewave overtones, there is no reason to object to Walsh harmonics from an academic point of view, particularly for persons on low budgets. From the musicians point of view, in an actual musical instrument, it may be impossible to tell from the exterior if Fourier or Walsh-Fourier methods or something else is used. Also, whenever it gets down to knob twisting, as it often does, the two are about equivalent. Bear in mind that these results in the table apply to equal excursion Walsh functions. See remarks on D/A methods in 4g. See also Appendix I.

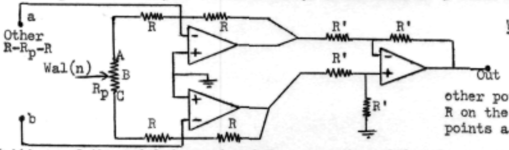
f) <u>Hardwear Generation of Walsh Functions</u>:[22] Hardwear realization is quite simple; we just have to follow the discussion above. The square waves are generated by simple divide-by-2 flip-flops, properly phased. EXclusive-ORing is done by EX-OR gates. The connection of the logic is shown below for the generation of the first 8 Walsh Functions. The circuit can be extended for more Walsh Functions by adding more flip-flops on the top side, and exclsuive-oring this flip-flop's input with the Walsh functions already generated. This should be obvious from the discussion above.



The actual TTL circuit we used is shown in Appendix J. While this circuit does work, and is useful, this is not intended as a model from the point of view of its particular features, but only as a general method and as an illustration of the proper interconnections of the IC's used.

g) <u>Digital to Analog conversion</u>: It is of course necessary to combine the various Walsh functions in proportion and in the proper sense to get the proper waveform for the calculated Walsh coefficients. Four possible methods are shown below:

(i) Strict Digital Coefficients: One pot of resistance R_p and two resistors R are required for each Walsh function we want to handle. The table gives the output for the extreme values of the pots (A=top, B=center, and C=bottom) for both the 0 and the 1 logic level outputs of the IC's (1 level taken as 5 volts here for illustration)

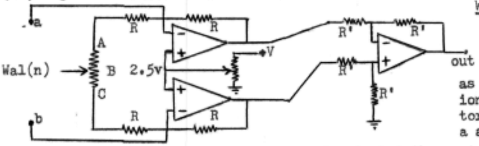


Wal(n)	A	B	C
+5	5	0	-5
0	0	0	0

other pots with resistors R on the ends connect to points a and b.

Settings of the pot between A and B represent positive C_n, settings of the pot between B and C represent negative C_n. The pots should be linear, and their value should be at least 50 times the value of R used.

(ii) Equal Excursions about zero (method 1):



Wal(n)	A	B	C
+5	2.5	0	-2.5
0	-2.5	0	2.5

as in (i) above, additional pots with resistors connect to points a and b

This method avoids the problem of D.C. error level, but the center reference voltage to the two op-amps must be well defined with respect to the ends.

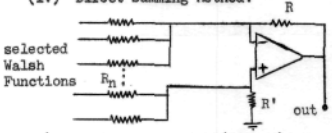
(iii) Equal Excursion about Zero (Method 2):

Wal(n)	A	B	C
+5	2.5	0	-2.5
0	-2.5	0	2.5

additional pots with summing resistors R connect to point a exactly as the one shown

This method is about the same as the method above, except for different parts required This one uses one inverter for each Walsh function, one less resistor per function, and two less op-amps total.

(iv) Direct Summing Method:



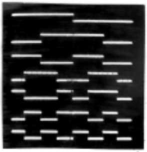
This is a one-waveform fixed circuit. You could have a good number of these in parallel to give a standard set of waveforms. Standard op-amp summing methods are used along with the appropriate C_n values to determine the R_n, R and R'.

h) <u>WAVEFORMS GENERATED</u> (<u>PHOTOS</u>) The photos on the next page show a group of waveforms as actually generated by the equipment in Appendix J. Captions describe the waveforms and the location of corresponding discussion of them in the text.

i) <u>Applications</u>: *Application possibilities are somewhat unlimited at the moment* We have discussed the generation of basic waveforms above, and use of the Walsh Functions for envelope shaping is found in Appendix C. The applications to filtering seem quite attractive, but we are not prepared for definite suggestions at the moment. A few other possibilities are considered below:

(i) The study of conventional instrument timbres by Synthesis: In such cases, the conversion from Fourier to Walsh Fourier coefficients would seem to be indicated, as sinewave overtones are considered in the available literature.

(ii) Live Performance: Frequency information and amplitude information could be recovered from a live instrument by some sort of a pickup, and this would then be used to generate a new instrument sound. The live instrument might well be muted in such a case, to give only the new instrument. This process would very likely require some sort of frequency multiplication technique to give a much higher pitch to drive the Walsh generator. A possible phase locked frequency multiplier scheme is indicated on page 17. While it might be difficult for a phased-locked-loop to to have a wide enough capture range, this might be useful for written music where the performer has plenty of time to program the Walsh generator, and set the center frequency of the loop in preparation for a passage to come.



Wal(1)
Wal(2)
Wal(3)
Wal(4)
Wal(5)
Wal(6)
Wal(7)

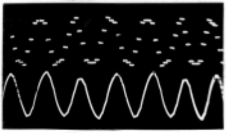
Walsh Functions 1 to 7
(See page 6)

1/4 Duty Cycle Pulse Generated as on Page 9
(note this is more easily obtained as the AND of Wal(1) and Wal(3))

3/4 Duty Cycle Pulse Generated as on Page 9
(note this is more easily obtained as the OR of Wal(1) and Wal(3))

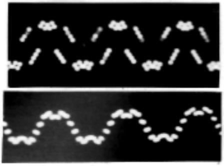
Sawtooth Wave - Staircase Approximation
Generated as on Page 10

Triangle Wave Generated as on Page 10

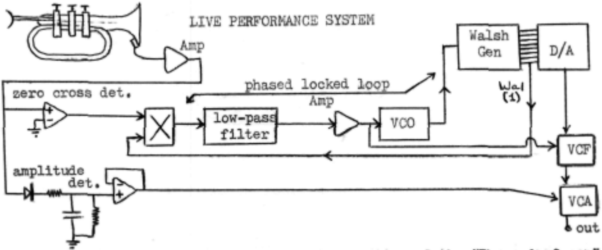


Third Harmonic Sine Wave Generated from Wal(1), Wal(5), Wal(9), Wal(13), Wal(21) & Wal(25) only. Coefficients from Page 13. See also Appendix I.

Waveform Above with Low-Pass Filtering



Misc. Periodic Waveforms. While these waveforms are very interesting to look at, they are no more interesting to listen to than many of the smoother ones. In general it is true that the ear doesen't listen very long to any periodic waveform of this type, and it is with amplitude shaping and time dependent harmonic percentages, not fancy waveforms, that an interesting musical sound is achieved.



(iii) Scientific studies of phasing: The question of the "Phase deafness" of the ear is an unanswered question[23]. Ohm's Law of acoustics states that the ear is unable to hear the phase relationship of the overtones in a periodic function. Others contend otherwise. Since it is so easy to change the phase of Walsh harmonics by simply substituting a cal for a sal, it would seem that Walsh functions could be a useful tool.

(iv) Control Voltage Sequences: Since all waveforms generated by Walsh Functions necessairily involve discrete steps, when steped relatively slowly, a sequence of control voltages could be made available for various purposes when used with voltage controlled systems.

REFERENCES:

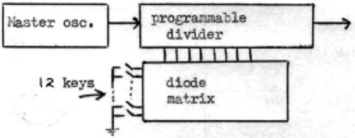
1. R.A. Moog "Voltage Controlled Electronic Music Modules" JAES 13 #3, (1965)
2. F. Maynard "Digisyntone" Radio-Electronics Sept. 1970
3. R.W. Burhans "Digital Tone Synthesis" JAES 19 #8, Sept. 1971, and "TTL Circuits for Monophonic Devices" AES Preprint 886 (N-1), 43rd Convention, Sept. 72
4. J. Marshall "A TTL Digisyntone" Electronotes Newsletter #13 Oct. 10, 1972
5. R. Schrecongost "LSI in Organ Design" JAES 20 #4 May 1972
6. ______________ "Pipe Organ Goes Digital" Electronics May 24, 1971
7. D. Lancaster "Psych-Tone" Popular-Electronics Feb. 1971
8. B. Hutchins "Digital Synthesizer" in EN#1 to EN#3 Jan-Mar. 1972
9. B. Hutchins "Frequency Modulation" EN#13, Oct. 10, 1972 (see pg 4)
10. Girling & Good "Active Filters 8: The Two Integrator Loop, Continued" Wireless World March 1970
11. H. Bode and R.A. Moog "A High Accuracy Frequency for Professional Audio Applications" AES Preprint No. 865 (D-6) 1972
12. B. Hutchins "Frequency Shifting 2" EN13 page 11, Oct. 10, 1972
13. E.T. Powner "Digital Waveform Synthesis" Electronic Engineering Aug 69, pg 50
14. K. Huehne "Programmable ROM's Offer a Digital Approach to Waveform Synthesis" EDN Aug. 1, 1972
15. L.J. Mandell "Sine-Wave Synthesizer Has Low Harmonic Distortion" EDN Aug. 15 1972. Corrections, Nov. 1, 1972
16. B. Hutchins "Digital Harmonics 1" EN#12, pg 5, Sept. 20, 1972
17. B. Hutchins, et al "Digital Harmonics 2" EN#13, pg. 12, Oct. 10, 1972
18. R.W. Burhans "Pseudo-Noise Timbre Generators" JAES 20 #3, April 1972. See also additional information, JAES 20 pg 210-211
19. H.F. Harmuth Transmission of Information by Orthogonal Functions, 2nd Ed. pg. 91 Springer-Verlag, 1972
20. B. Hutchins "Fourier Analysis 2" To appear in EN#16
21. K. Siemens & R. Kitai "Digital Walsh-Fourier Analysis of Periodic Waveforms" IEEE Trans. on Instru and Measur. IM-18 #4 pg. 316 Dec. 69
22. Reference 19, pages 90-91
23. V. Lozhkin "Monaural Phase Effects" Soviet Phys. Acoustics 17 #1, July-Sep. 71

CONCLUSIONS: We wish to emphasize that while the systems and modules described in this report are believed or known to work properly (except untested ideas as noted), they have not received a full technical or musical evaluation as yet. Therefore, this report should be used more as a basis for design ideas, and not as a basis for construction. Furthermore, this report will appear as a supplement to Electronotes Newsletter #16, and future issues of the newsletter may expand on the material given here. More exacting design and construction details may appear as well.

ACKNOWLEDGMENT: We are greatly indebted to Dr. Carl Frederick of the Center for Radiophysics and Space Research at Cornell Univ. for suggesting the use of Walsh functions for music synthesis, and for many valuable discussions. We also acknowledge the support of Professor Walter Ku of the Dept. of Electrical Engineering at Cornell, and the advice of Lou Phillips with regard to the formulation of the computer program.

* * * * * * * * *

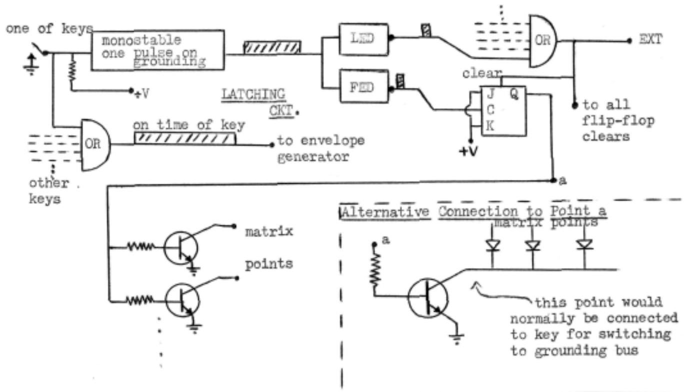
APPENDIX A: Digital Methods of Scale Tone Generation: The basic system indicated below has been used for the accurate generation of musical scale tones:



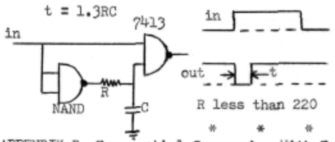
Diodes are placed in appropriate positions in the matrix to program the divider to divide by various numbers (from 2 to 255 in the case of an 8-bit counter) and these numbers are picked for the ratio $1:2^{1/12}$ (1:1.059463....) in the case of a well-tempered scale.

Several instruments of this type have been designed and constructed.[2,3,4] The application of this system to other than the well-tempered scale has been discussed (D.Gossel "Generation of Musical Intervals by a Digital Method", Phillips Technical Review, 26 pg 170 (1965), and N. Franssen & C. van der Peet, "Digital Tone Generation for a Transposing Keyboard Instrument", PTR, 31 (1970) pg 354) and the general case could of course be approached with a matrix switching process for each point on the diode programmer. This sort of instrument might be of great interest to musicologists who are studying music from a period when other methods of scale tuning were used. The generation of octaves other than the one produced by direct counting can be accomplished by simply adding flip-flop divide-by-two circuits, switched by a separate bus on the keyboard, or through use of the single bus technique devised by R. Burhans ("Single Bus Control for Digital Musical Instruments" JAES 19 #10, pg. 865, Nov. 71, and AES Preprint #886) Actual numbers commonly used for an 8-bit divider are: 123, 130, 138, 146, 155, 164, 174, 184, 195, 207, 219, 232. Stapelfeldt ("Approximating the Frequencies of the Musical Scale with Digital Counter Circuits" JASA 46 #2 (Part 2), pg 478 (1969)) has determined the proper integers for minimum error for counters of total capacity from 6 to 12 bits. Cotton ("Tempered Scale Generation from a Single Frequency Source", JAES 20 #5, June 1972, pg 377) has discussed two other methods of digital scale generation; one involves addition of pulses, and the other involves repeated dividion by 196/185 to approximate the twelfth root of 2.

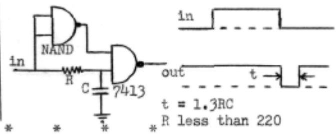
It is of course desirable to have a memory for frequency during decay (i.e., the equivalent of a sample-and-hold with a VCO). A possible, but undeveloped system for holding one pitch activated by a keyboard until another key is pressed is indicated on the next page. Also shown is a possible TTL realization of a leading-edge-detector (LED) and a following-edge-detector (FED) based on H.A. Cole ("TTL Trigger Circuits" Wireless World, Jan 1972, pg 31, with corrections in March 1972, pg 116).



Leading Edge Detector (LED)



Following Edge Detector (FED)

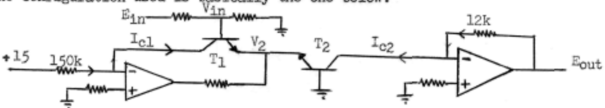


* * * * * * *

APPENDIX B: Exponential Conversion With Transistors: Shockley's theory of the PN junction gives: $I = I_o(\exp(qv/k_BT) - 1)$, and this results in a collector current that is an exponential function of the base-emitter voltage as defiend below:

$I_c = aI_{es}\, e^{-qV_{BE}/k_BT} = A\, e^{-BV_{BE}}$
for currents I_c from 1 pa to 1 ma

The configuration used is basically the one below:



$I_{c1} = 15/150k = 0.1$ ma. We can write equations for both transistors:

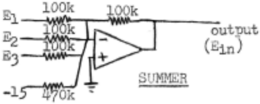
$I_{c1} = A\, e^{-BV_{BE1}} = A\, e^{B(V_{in}-V_2)}$ and $I_{c2} = A\, e^{-BV_{BE2}} = A\, e^{-BV_2}$

now: $I_{c1} = 0.1\text{ma} = A\, e^{-BV_{in}}\, e^{-BV_2} = \cancel{A}\, e^{BV_{in}} . I_{c2}/\cancel{A}$

so: $I_{c2} = 0.1\text{ma}\, e^{-BV_{in}}$ and therefore $E_{out} = 12k \cdot I_{c2} = 1.2\, e^{-BV_{in}}$

for exact circuit used, including voltage divider on input: $E_{out} = 10^{-E_{in}}$

The op-amp summer is used to sum control voltages, and to put the exponential converter in the proper range:



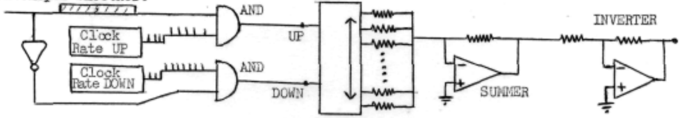
The summer output (E_{in} of the exponential converter is offset by a constant voltage setting the output at +3 volts when $E_1 + E_2 + E_3 = 0$. Thus the output voltage is $10^{-3} = 0.001$ volts. When $E_1 + E_2 + E_3 = 4$ for example, $E_{in} = -4 + 3 = -1$, and $E_{out} = 10^1 = 10$.

References for appendix B: (1) R.C. Dobkin "Logarithmic Converters" National Semiconductor Applications Note AN-30 (1969) (2) K. Huehne "Transistor Logarithmic Conversion Using an Integrated Operational Amplifier" Motorola Semiconductor Products Application Note AN-261A (1971) (3) W.L. Paterson "Multiplication and Logarithmic Conversion by Operational Amplifier-Transistor Circuits" Review of Scientific Instruments 34 #12 Dec. 1963

* * * * * * *

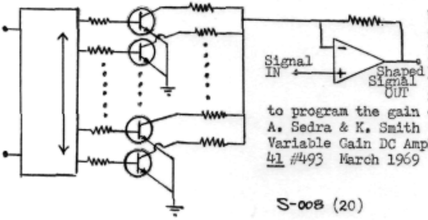
APPENDIX C: Digital Methods of Envelope Generation: In regard to digital methods of obtaining control envelopes, several methods come to mind. We require these control envelopes for both amplitude and harmonic shaping. We assume that some signal is present, either as an analog trigger derived directly from a comparator signifying the down position of a key, or as indicated in Appendix A. The methods below have not been implemented or tested.

(1) Suppose we have a counter of the bucket brigade type, which when periodically triggered will count up to the top and stop, and upon triggering in a different mode will count down to the bottom and stop. Presumably, it will also reverse direction in midcount as well. A simple envelope generator could be made that operates much like the capacitor charge-discharge type of V.C. systems. A simple arrangement could be as follows: We assume here and elsewhere in this appendix that some means of smoothening the sharp corners (e.g. with a low-pass filter) is available if needed to make the envelopes smoother:



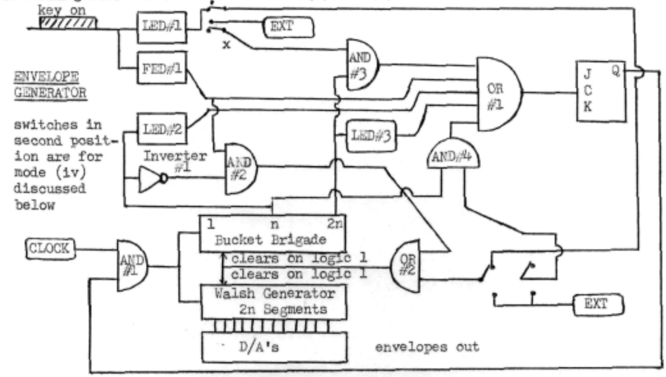
Resistance values would be selected for any desired envelope waveform, and any number of parallel envelopes could be generated.

(2) A generator of the above type could be used for direct gain control, instead of applying the envelope to a VCA, in the following manner:

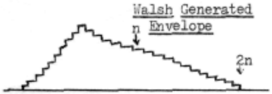


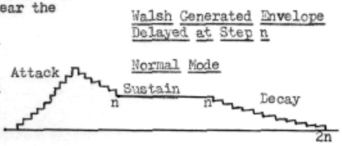
Where the resistors to the bases of the transistors are selected to saturate the transistors on logic 1, and the resistors from the collectors to the inverting input of the op-amp are selected to program the gain of the op-amp stage. Reference: A. Sedra & K. Smith "Simple Digitally-Controlled Variable Gain DC Amplifier" Electronic Engineering 41 #493 March 1969

(3) In a very general type system, where a very large number of envelopes are to be generated (for eaxmple in a complete additive synthesis system), generation of envelopes by Walsh functions (Part 4) might be considered. Far fewer summing resistors would be required in many cases. This system as outlined below also has a re-triggerable feature that might be musically advantageous (see D. Rossum, EN#13, pg. 16, sect. 8e). Note that the Walsh function generator operates parallel to the bucket brigade, and is not essential, but a means of assuring general envelopes for the least number of summing resistors. The methods of (1) and (2) above could be used directly.



The Normal Mode of the system is as follows: Bucket brigade is in the 2n "rest" state. Clearing puts the bucket brigade in the 1 state. Suppose we have an envelope prepared to be generated by Walsh functions as shown at the right. When the leading edge of the signal signifying that a key is down is detected by leading edge detector #1 (LED#1), the output of LED#1 clears the counter through OR Gate #2 and also fires AND Gate #3 and OR Gate #1, turning on flip-flop #1 to Q=1, thus turning on AND Gate #1, starting up the bucket brigade counter and the parallel Walsh function generator. The timing on these events should be correct. The bucket brigade counts up to n where LED#2 turns the flip-flop off again. Then, upon release of the key, Following edge detector #1 (FED#1) turns the flip-flop back on again (but does not clear the counter, prevented by inverter #1) and the counter counts down to 2n, where LED#3 again shuts it off again (rest state) See right:

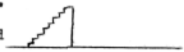




Note that the system makes no provisions for different clock rates during attack and decay. Although this could probably be done fairly easily, theoretically it is not necessary. A change in the apparent attack or decay rates could be made by changing the slope (as a function of n) of the Walsh-generated envelope.

However, the fact is that musicians don't always play in such nice normal modes as outlined above, hence, we must examine how the system responds in other modes.

(i) Key is lifted during count from 1 to n: Inverter 1 is in state 1, so AND Gate #2 will deliver the pulse from FED#1 to clear the counter. Thus, the envelope will drop rapidly to the cleared state of zero voltage, and presumably, this sudden drop could be delayed slightly by a capacitor if it is unpleasently sharp.



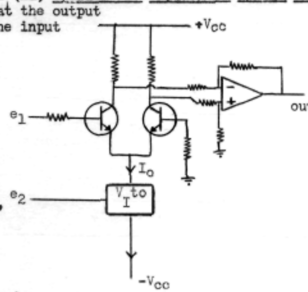
(ii) Key is lifted, starting decay from n to 2n, but another key is pressed during the count from n to 2n: LED#1 signal will clear the counter, but not turn off the flip-flop, prevented by AND Gate #3, thus the system finds itself in the 1 to n count of the normal mode, and thus counts up to n and stops, or takes mode (i) as outlined above.

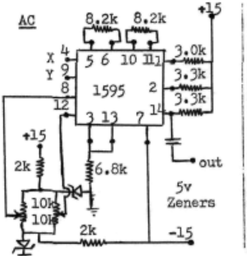
(iii) The problem of simultaneous events: (Keyboard switching transitions at same time as counter switches in or out of the n or 2n states: This is a problem for the designer working with a specific logic family to answer. Very likely, refinements may be needed, or the case might be insignificantly rare. Requires further study.

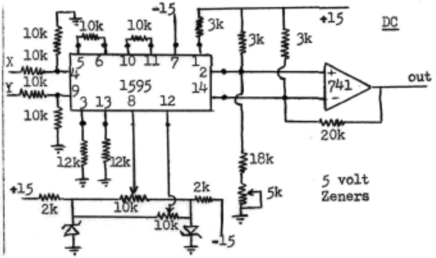
(iv) Case where Pitch is changed by sliding from one key to the other without actually lifting the first key before the second makes contact: Here, we refer back to Appendix A, and note that the point marked EXT in the second figure delivers a pulse whenever a new key is pressed, regardless of whether another key is still down or not. If we are using this digital generation of scale tones, we can use the EXT output as a third input to OR Gate #2, to clear the counter and start a new envelope. Note that in general, the counter would probably be in the n state when the pitch change arrived, so it is necessary for the counter to be restarted through AND Gate #4. At this point. LED#1 becomes redundant, and we would probably disconnect it and connect the EXT pulse to the point marked x. This would of course make OR Gate #2 a two input gate again. Further, it is clear that the EXT input is essential if we are to permit mode (iv), so even if a V.C. system is used, it would pay to use a differentiator on the control voltage to provide this EXT input. Thus, the signals we want to have are one representing a change of pitch (different key or change of input control voltage) and a second representing the point at which all keys are lifted.

* * * * * * * *

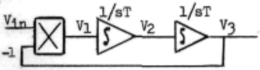
APPENDIX D: Multipliers: The multiplier we used in the various voltage controlled modules is of the monolithic, variable transconductance type. In its most basic form, this is a differential amplifier stage with a voltage to current converter. It is easy to show (see for example J. Graeme et al (Ed) Operational Amplifiers Design and Applications, Mc-Graw Hill (1971) pg 276, that the output of this combination is proportional to e_1, the input to the differential amplifier, second input grounded, and to the current I_o, which in this case is made proportional to e_2. Hence the output is proportional to the product of the input voltages. In the monolithic IC (type 1595) this is made into a four-quadrant device. An excellent discussion of the device is found in Ch. 12 "The Linear Four-Quadrant Multiplier" by E.L. Renschler, in J. Eimbinder (Ed) Applications Considerations for Linear Integrated Circuits, Wiley, (1970). AC and DC versions for multiplier modules are easily set up, and shown on the next page. An improved version of the 1595, the 1594, is also available.

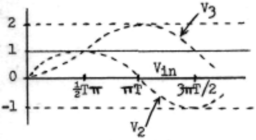






APPENDIX E, The Two Phase, Two Integrator, Quadrature Oscillator: To understand the two integrator loop as an oscillator, we have only to recall that it presents a 90° phase shift over all frequencies, and the roll-off is -20 db/ decade and we can represent this as $1/sT$, where s is the complex frequency, and T is the time constant of the integrator. Two integrators in series thus represent -40 db/decade, and a -180° (+180°) phase shift. Putting in a loop with one inversion, at low frequency, the high forward gain gives unity loop gain, and at the frequency where the forward gain drops to unity, the signal fed back is of course the same as the input to the first integrator, and the phase shift is of course 180° + 180°, hence it oscillates.





Mathematically, this follows from the fact that $V_2 = sTV_3$ and therefore $V_1 = s^2T^2V_3$. Hence when $V_{in} = 0$, $-s^2T^2V_3 = V_3$, which is the equation of simple harmonic motion at frequency $f = 1/(2\pi T)$, where $s = j \cdot 2\pi f$. Note that for unequal time constants of the integrators, the equation for the freq. of oscillation is the same except $T = (T_1T_2)^{\frac{1}{2}}$. (Ref: E.Good, Electronic Engineering, 29 April 1957)

* * * * * * * *

APPENDIX F, 90° Phase Shift Networks: To realize the 90° phase shift networks, one uses a pair of all-pass networks. Each network has a zero in the right half plane for each pole in the left-half-plane to give flat gain characteristics and non-minimal monotonically decreasing phase shift. The phase difference between the two is made to ripple about 90° over a specified frequency range. The ratio of the transfer functions contains both poles and zeros in the R.H.P. With enough poles, it is theoretically possible to approximate 90° over any frequency range with any phase tolerance desired. Pole location is determined by making positive and negative phase deviations about 90° equal and minimal. This can be done by use of the Chuer functions in a manner similar to the use of Chebyshev functions for equal amplitude ripples in the passband.

References: W. Albersheim & F. Shirley "Computation Methods for Broad-Band 90° Phase-Difference Networks" IEEE Trans on Ckt. Theory May 1969, pg 189. This presents the necessary theory and the element values of a phase shifter (6-poles, 25-6000 Hz, 3° phase tolerance, 4 op-amps, 16 precision passive components). F. Shirley "Shift Phase Independent of Frequency" Electronic Design 18, Sept. 1, 1970 gives a phase shifter (4-pole, 70-2200 Hz, 3.8° phase tolerance, 2 op-amps, 12 precision passive components). R. Orban gives a very interesting phase shifter (90°, ±1.96° from 20-20,000 Hz using discrete components (JAES 18, #4, pg. 442, August 1970). Finally, reference 11 shows the schematic of the "dome filter" used by Bode and Moog, but component values are not given. Note that the networks given in the second reference were used in the frequency shifter we constructed.

APPENDIX G: Some References on the Walsh Functions:

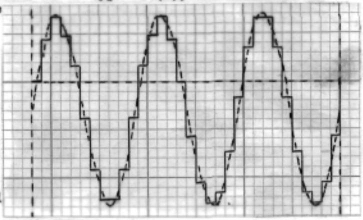
1. J.E. Walsh "A Closed Set of Normal Orthogonal Functions" American Journal of Mathematics 45 pg 5 (1923)
2. N.J. Fine "On the Walsh Functions" Amer. Mathematical Society Transactions, 65 pg 372 (1949)
3. N.J. Fine "The Generalized Walsh Functions" Trans.Amer. Math. Soc. 69 (1950),66
4. G. Morgenthaler "On the Walsh-Fourier Series" Trans Amer Math Soc 84 (1957),472
5. H.F. Harmuth, Transmission of Information by Orthogonal Functions, Springer-Verlag N.Y., Heidelberg, Berlin. (1972)(2nd Ed)
6. H.F. Harmuth "A Generalized Concept of Frequency and Some Applications" IEEE Trans on Information Theory, IT-14 #3 May 68, pg 375-382
7. ______ 1970 Walsh Function Symposium (AD 707-431) 40 papers, 274 pages, $3.00 from National Technical Information Service, Operations Division, Springfield, Va. 22151. order number AD 707-431
8. ______ Applications of Walsh Functions, 1971 Proceedings, 13-15 April, Washington, D.C. 38 papers order number AD 727-000
9. K. Siemens & R. Kitai "Digital Walsh-Fourier Analysis of Periodic Waveforms" IEEE Trans on Instrumentation and Measurment IM-18 #4 pg 316 Dec. 1969
10. N. Alexandrisdis "Relation Among Sequency, Axis Symmetry, and Period of Walsh Functions" IEEE Trans on Information Theory IT-17 #4 July 1971
11. H. Andrews & K. Caspari "A Generalized Technique for Spectral Analysis" IEEE Trans on Computers C-19 #1 Jan 1970, pg 16-25
12. J. Gibbs & H. Gebbie "The Application of Walsh Functions to Transform-Spectroscopy" Nature 224 (1969) pg 1012-1013
13. K. Henderson "Some Notes on the Walsh Functions" IEEE Trans Electronic Computers (Correspondence) EC-13 pg 50-52, Feb 64
14. S. Kak "Sampling Theorem in Walsh-Fourier Analysis", Electronic Letters, 6, pg 447, July 1970
15. F. Pichler "Walsh-Fourier-Synthese Optimaler Filter" AEU 24 (1970) pg 350
16. J. Shanks "Computation of the Fast Walsh-Fourier Transform", IEEE Trans on Computers, C-18 pg 457 May 1969
17. F. Pichler "Some Aspects of a Theory of Correlation with Respect ot Walsh Harmonic Analysis" U. of Md. Report R-70-11, College Park, Md. (1970)
18. R. Lackey "So What's A Walsh Function" Proc. IEEE Fall Electronics Conference, pg 368, (1971)
19. R. Adams Properties and Applications of Walsh Functions, Thesis, U. of Pa. Moore School of Elect. Eng., Philadelphia (1962)
20. E. Gibbs "Walsh Spectroscopy, A form of Analysis Well Suited to Binary Digital Computation" (1967) Unpublished
21. S. Manoli "Walsh Function Generator" Proc IEEE 59 (1971) pg 93
22. H.F. Harmuth, "Application of Walsh Functions in Communications" IEEE Spectrum, Nov. 1969, pg 82
23. J. Hammond & R. Johnson "A Review of Orthogonal Square-Wave Functions" Journal of the Franklin Instit., 273 (1962) pg 211
24. G.R. Redinbo "An Implementation Technique for Walsh Functions" IEEE Trans on Computers C-20 June 1971 pg 706
25. N.M. Blackman "Spectral Analysis With Sinusoidal and Walsh Functions" IEEE Trans on Aerospace and Elect. Systems AES-7 (1971) pg 900
26. D.A. Swich "Walsh Function Generation" IEEE Trans on Info Theory IT-15 pg 167 Jan 1969
27. W.G. Szok Waveform Characterization in Terms of Walsh Functions Master's Thesis, Syracuse U. June 1968

NOTE: Not all of the above papers have been reviewed as yet, and numerous others exist that have not been listed. Consequently, we are not certain that some of the material presented in this report has not been done elsewhere.

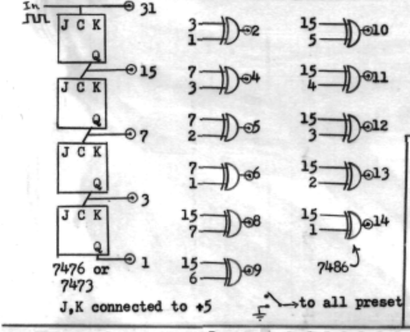
APPENDIX H: Computer Program For Walsh Matrix and Sinewave Harmonic Coefficients:

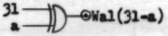
```
STMT LEVEL NEST BLOCK     SOURCE STATEMENT

  1                        WALSH:  PROCEDURE OPTIONS(MAIN)  ;
  2     1           1              DECLARE(N,M,R) FIXED ;
  3     1           1              GET LIST(M)  ;
  4     1           1              R = CEIL(2**M) ;
  5     1           1              BEGIN ;
  6     2           2                   DECLARE W(R-1,R) BIT(1),(I,J,K,L,N,D) FLOAT ,
                                                COEF(R-1) FLOAT,(X,PI,A,F) FLOAT,
                                                 Y FLOAT, SUM FLOAT INIT(0) ,
                                                 (C,P) FLOAT, Q FIXED ;
  7     2           2              L=R ;
  8     2           2              DO K=1 TO M ;
  9     2     1     2                 L=L/2 ;
 10     2     1     2                 J= CEIL(2**K-1) ;
 11     2     1     2                 DO I=1 TO R ;
 12     2     2     2                    P= I/L-1/R ;
 13     2     2     2                    C= FLOOR(P)/2 ;
 14     2     2     2                    W(J,I) = (FLOOR(C)-C=0) ;
 15     2     2     2                 END ;
 16     2     1     2              END ;
 17     2           2              DO K=1 TO M-1 ;
 18     2     1     2                 L= CEIL(2**K-1) ;
 19     2     1     2                 N= CEIL(2**(K+1)-1) ;
 20     2     1     2                 DO Q=1 TO L ;
 21     2     2     2                    DO I=1 TO R ;
 22     2     3     2                       W(N-Q,I) = (W(N,I) ¬=W(Q,I)) ;
 23     2     3     2                    END ;
 24     2     2     2                 END ;
 25     2     1     2              END;
 26     2           2              PUT SKIP ;
 27     2           2              DO I=1 TO R-1 ;
 28     2     1     2                 DO J=1 TO R ;
 29     2     2     2                    PUT EDIT(W(I,J))(B(1)) ;
 30     2     2     2                 END ;
 31     2     1     2                 PUT SKIP ;
 32     2     1     2              END ;
 33     2           2              PI= 3.1416 ;        A=1/R ;
 35     2           2      START:  GET LIST (D) ;
 36     2           2              F= 2*PI*D ;
 37     2           2              DO I=1 TO R-1 ;
 38     2     1     2                 X=0;          COEF(I) = 0 ;
 40     2     1     2                 DO J=1 TO R ;
 41     2     2     2                    IF W(I,J) ='1'B THEN Y= 1 ;
 43     2     2     2                    ELSE Y = 0 ;
 44     2     2     2                    SUM = (COS(F*X)-COS(F*(X+A)))*Y/F ;
 45     2     2     2                    COEF(I) = SUM+COEF(I) ;
 46     2     2     2                    X=X+A ;
 47     2     2     2                 END ;
 48     2     1     2              END ;
 49     2           2              PUT PAGE ;
 50     2           2              PUT DATA(COEF) ;
 51     2           2              GO TO START ;
 52     2           2              END ;
 53     1           1              END WALSH ;
```

APPENDIX I, Comments on the Level Shift Problem: The D.C. level shift problem due to truncation of the Walsh-Fourier series was mentioned on page 11. When generating higher sinewave harmonics, a similar problem arises for the following reason: By defining a given sinewave frequency as the fundamental, we necessairily define a corresponding sequency for Wal(1). When synthesizing the harmonics of order 2^i, where i is an integer, the second for example, we have only to start with the Walsh function of twice the sequency. However, when synthesizing the third sinewave harmonic, the Walsh function of three times the sequency is not in the original set, and thus the synthesis is not the most natural one. As a result, we see a unbalanced effect on the synthesized waveform when the Walsh-Fourier series is truncated. This error is of type Wal(1), and therefore is not corrected by A.C. coupling. Note from the drawing at right that the unbalance is very small after the first 31 Walsh functions have been considered. Also from the photo on page 16 note that every third cycle shows a displacement. The drawing considers all the Walsh harmonics listed on page 13 for the third sinewave harmonic, while the photo omits C_{17} and C_{29} due to equipment limitations. This may well be more important for higher sinewave harmonics. Note also that the D.C. error is still with us, but is not a problem. In practice, we might simply reduce the value of C_1 to give a more balanced waveform, and this would result in a slightly revised version of the table on page 13 corresponding to a given truncation of the Walsh-Fourier series. Further study is indicated.



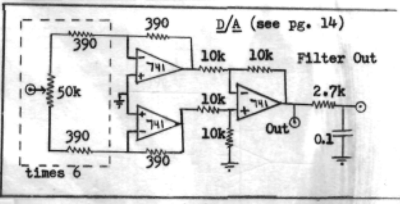
APPENDIX J, Hardwear Setup Used: Numbers indexing connection points represent the order of the Walsh Function at that point.

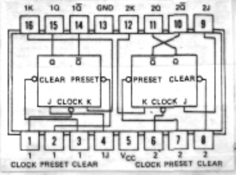


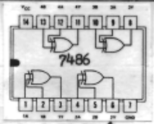


Three of these units are available
a is connected by selector switch to any of Wal(1) to Wal(15) to give selectively Wal(16) to Wal(31)

WALSH FUNCTION GENERATOR







Quad EX-OR

Dual JK Flip-Flop (7476)

Wal(1) through Wal(15) are available at their output jacks. Three of Wal(16) through Wal(31) may be selected by the switches. Up to six Walsh functions may be processed through the D/A, which sets the correct values of C_n