# A DIGITAL WAVEFORM SYNTHESIZER

# USING WALSH FUNCTIONS

# A DIGITAL WAVEFORM SYNTHESIZER USING WALSH FUNCTIONS

By

W. OWEN BROWN, B. ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

November, 1971

MASTER OF ENGINEERING (1971)
(Electrical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: A Digital Waveform Synthesizer Using Walsh Functions

AUTHOR: W. Owen Brown, B.Eng. (McMaster University)

SUPERVISOR: Dr. A.R. Elliott

NUMBER OF PAGES: x, 131

SCOPE AND CONTENTS: This thesis describes the design of a digital waveform synthesizer based on the Walsh series representation of a signal. By designing the unit to operate serially, simplicity and economy have been achieved. Although basically meant to be used as a speech synthesizer to be interfaced to a computer, the unit can operate independently as a low frequency function generator capable of producing essentially any finite waveform having a frequency from zero to 200Hz. The mathematics behind the Walsh series is developed and parameters are adjusted to suit speech synthesis by a short investigation of the properties of speech. Evolution of the hardware design, including detailed analysis of the final circuitry, is also given. Sources of error are investigated and compared to error measurements made from basic waveforms generated by the synthesizer. Finally, a discussion of potential uses of the synthesizer is included.

## ABSTRACT

This thesis describes the design of a digital waveform synthesizer based on signal representation by the Walsh series. The evolution of the machine design is given, along with a short error analysis. The instrument was constructed and preliminary measurements indicate output waveforms well within the bounds given by error analysis.

## ACKNOWLEDGEMENTS

I would like to thank Dr. A.R. Elliott for his constant encouragement and assistance during the course of this work and in the preparation of this thesis.

The financial assistance provided by the Department of Electrical Engineering and the National Research Council of Canada is also gratefully acknowledged.

TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

## LIST OF TABLES

## LIST OF SYMBOLS

| | | |
|---|---|---|
| $t$ | = | time |
| $f(t)$ | = | analog signal represented by a series |
| $f_k$ | = | value of $f(t)$ at the $k^{th}$ sampling instant |
| $T$ | = | period of $f(t)$ |
| $\theta$ | = | position variable for the Walsh functions |
| $n$ | = | Walsh function number |
| $CAL(n,\theta)$ | = | $n^{th}$ even Walsh function |
| $SAL(n,\theta)$ | = | $n^{th}$ odd Walsh function |
| $WAL(n,\theta)$ | = | $n^{th}$ general Walsh function |
| $WAL(j,\theta_k)$ | = | value of $WAL(j,\theta)$ at the $k^{th}$ sampling instant |
| $a,b$ | = | limits on the interval of orthogonality |
| $C_m$ | = | weighting constant |
| $\phi$ | = | basic time interval of the Walsh functions |
| $a_n$ | = | coefficients for the odd orthogonal functions |
| $b_n$ | = | coefficients for the even orthogonal functions |
| $c(n)$ | = | coefficient of $WAL(n,\theta)$ in the Walsh series |
| $C_{k_n}$ | = | $n^{th}$ most dominant coefficient |
| $\Psi$ | = | phase |
| $M$ | = | number of terms used in the series representation of $f(t)$ |
| $N$ | = | number of samples taken of $f(t)$ during the interval $\phi$ |
| $P$ | = | number of computer words used to produce one spoken word |
| $A$ | = | second order Hadamard matrix |

LIST OF SYMBOLS (cont'd)

| | | |
|---|---|---|
| B | = | fourth order Hadamard matrix |
| C | = | eighth order Hadamard matrix |
| D | = | sixteenth order Hadamard matrix |
| $f_{IN}$ | = | rate multiplier input frequency |
| $f_{OUT}$ | = | rate multiplier output frequency |
| $\alpha$ | = | binary number on the control lines of a rate multiplier |
| $\beta$ | = | number of states in the counter used with the rate multiplier |
| h,j,k | = | integers |

# CHAPTER 1

## INTRODUCTION

There are several methods, of varying complexity, commonly used for the synthesis of waveforms. Techniques range from the simplest, allowing the generation of only a small class of basic waveforms, to more sophisticated devices which are programable and can synthesize virtually any arbitrary waveform. Function generators producing sines, pulses and ramps are examples of the simplest methods while a unit such as described by Böβwetter (6)* for synthesizing speech is indicative of the more complex methods.

This thesis describes a unique, all digital device, suitable for operation as a programable waveform synthesizer and ultimately as a speech synthesizer. In the design, such factors as cost, simplicity and maximum flexibility were sought. Also, the prerequisite of minimizing the computer memory needed to produce acceptable speech was an overriding consideration in this work. The result is a waveform synthesizer performing the Inverse Walsh Transform. The detailed design is described in Chapter 3. To more clearly illustrate why this method of synthesizing waveforms was chosen, consider the following possibilities.

The simplest kind of synthesizer that can be interfaced to a com-

---

* Numbers in brackets refer to the list of references given at the end of the thesis.

puter works in the time domain and consists of a word register and a digital-to-analog converter. Binary numbers, representing the amplitude of the desired output signal, are fed into the word register and then converted to the correct analog voltage. Such a method has been successfully used in the generation of electronic music (1). The accuracy of this system is dependent on the number of bits in the word register, which determines how accurately the amplitude is represented, and the rate at which numbers are fed into the word register.

There is a minimum acceptable sampling rate for quantizing analog data which is given by Shannon's sampling theorem. For intelligible speech, a minimum bandwidth of 3kHz is required. This is the standard usually adopted by telephone companies. If the sampled speech is then limited to a 3kHz bandwidth, the sampling theorem requires that the sampling rate be at least 6kHz. Thus, the word register would have to be updated at least every 167 μsec. This technique requires a great deal of computer memory to generate the simplest words. Since the average spoken word has a duration of approximately five to seven tenths of a second (16), the minimum number of computer words needed would be:

$$P = 0.5 \times \frac{10^6}{167}$$

$$= 3000 \text{ words} \qquad (1\text{-}1)$$

Obviously, if the computer is to have a speech vocabulary of any appreciable size, a reduction in the number of computer words needed per spoken word is desirable. One possible method is to break speech down into its basic sounds, or formants (2). The required formants are then generated

sequentially to produce the desired words. This results in a reduction in memory requirements by a factor of up to fifty. On the other hand, there are increased software problems related to ordering the formants and blending them together. The problem of analysis of the speech to determine the correct formants also becomes sizeable.

The simplest of the time domain systems consists of a bank of computer-controlled analog tape recorders on which are recorded the desired speech. The computer is used to locate the required word or phrase on the tapes. Such a system has been used at the London International Airport for several years to handle routine messages on the public address system. It offers the advantage of good fidelity and is quite suitable if a long series of words is often reproduced. However, its lack of flexibility and tendency to break down due to its electromechanical nature make it unsuitable as a general purpose waveform synthesizer.

To reduce the amount of computer memory needed to form a given word, some type of data compression is needed. This is not feasible in the time domain so it is now necessary to consider the frequency domain. By applying the Fourier Transform, all time-varying, periodic waveforms can be converted to the frequency domain. The original signal can be reconstructed by a summation of the components in the frequency domain. This is a straight-forward application of the well-known Fourier series representation of a signal f(t).

$$f(t) = b_o + \sum_{n=1}^{\infty} a_n \sin\left(\frac{2\pi nt}{T}\right) + \sum_{n=1}^{\infty} b_n \cos\left(\frac{2\pi nt}{T}\right) \qquad (1\text{-}2)$$

where f(t) is the original signal (or resynthesized signal)

T is the period of waveform f(t).

More detail concerning the Fourier series is given in Chapter 2. The coefficients $a_n$ and $b_n$ are found by applying the following formula to the original signal f(t).

$$a_n = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi nt}{T}\right) dt \qquad (1\text{-}3)$$

$$b_n = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi nt}{T}\right) dt \qquad (1\text{-}4)$$

For non-periodic waveforms, a similar transform can be made by arbitrarily choosing T and mathematically assuming that the waveform within T repeats outside of T. The transform must then, of course, be updated after every interval T. The analysis is usually carried out using a Fast-Fourier Transform (3). The waveform is sampled at a rate determined by the sampling theorem (approximately 6kHz for speech). To calculate the first N coefficients, N samples are used simultaneously by the Fast-Fourier Transform algorithm. Thus, for speech, a new set of N coefficients is calculated every N/6000 seconds. These coefficients can then be stored in a computer and recalled when desired to generate the original signal f(t) by using equation (1-2). Note that the amount of data stored in the computer is the same as with the time domain systems. The computer must supply N coefficients every N/6000 seconds, or one word every 1/6000 seconds, which is equivalent to the 167μsec rate for the time domain system.

The amount of data stored in the computer can be reduced without appreciable degradation of the output signal simply by saving only the most dominant terms of the summation of equation (1-2). For instance, for speech (11), generally the first sixty-four coefficients are calculated

using the Fast-Fourier Transform (N=64). Thus, the coefficients are updated every 64/6000 seconds, or 10.67 milliseconds. Of these sixty-four usually only eight of the dominant* terms in the Fourier domain are sufficient to reproduce speech so that it may be understood. Previously published work done by Campanella and Robinson (11) demonstrates the feasibility of this approach. Their work has shown that the original signal can be reconstructed with sufficient accuracy to be suitable for satellite transmission of voice using only eight data words every 10.67 milliseconds. This gives a factor of eight saving on the amount of computer space needed to reconstruct the output signal.

However, a system of waveform synthesis using the Fourier series is generally not practical. The computer must not only supply the correct coefficients at the proper time, but it must also control the generation of the sine and cosine waves. Typically, eight out of a possible sixty-four sine and cosine waves must be generated and these must have accurate phase and frequency relationships. Such a system is very difficult to realize in hardware form since the digital computer must control the analog circuitry generating the waveforms, requiring a great deal of digital-to-analog and analog-to-digital conversion. A more simple solution is the generation of the sine and cosine waves by software. However, this ties up a great deal of computer time and essentially eliminates the advantages of

---

* Those coefficients $a_n$ and $b_n$ having the largest absolute value are said to be dominant. The largest absolute valued term is the most dominant, the second largest absolute valued term the second most dominant, and so on.

Fourier Series synthesis. Ideally, a hardware device, easily interfaced to the computer, and capable of producing the required phase and frequency related waveforms, should be developed to carry out the synthesis.

The obvious solution would be to search for waveforms, other than sines and cosines, that can be used to perform transformations, but which are more suitable to computer control. The property of sines and cosines that make them useful in the Fourier series is that they have the mathematical property of orthogonality. In 1923, J. L. Walsh discovered another set of orthogonal functions which became known as Walsh functions (4). These functions are binary in nature, having values of either +1 or -1. Thus, they lend themselves nicely to digital logic by assigning the value +1 to logical 1 and -1 to logical 0. A Walsh function generator that can generate any of the first sixty-four Walsh functions with proper phase and frequency relationships can be designed quite simply. A typical generator is described in Chapter 3. The mathematical representation of the Walsh series is almost exactly analogous to the Fourier series. This relationship is discussed further in Chapter 2. Thus, the desired signal can be derived from the following infinite series which is defined as the Walsh series expansion of f(t).

$$f(t) = \sum_{n=0}^{\infty} c(n)\ \mathrm{WAL}\ (n,\theta) \qquad (1\text{-}5)$$

where $\mathrm{WAL}(n,\theta)$ is the $n^{th}$ Walsh function.

Again, in a practical situation, the signal can be reconstructed using only the most dominant terms, allowing a reduction in the amount of computer memory needed to derive the speech output signal. Computer

interfacing is much simpler than it would be for a Fourier series synthesis since the Walsh function generator, and indeed all of the Walsh series circuitry, can be made out of digital logic. There is one other advantage to using Walsh functions instead of sine and cosine waves. From a Fourier analysis of the Walsh functions, it is obvious that all of them contain a large number of high frequency components (5). A great many speech sounds also contain a large number of high frequency components. Thus, a more accurate reproduction of speech can be obtained from the first sixty-four Walsh functions, for example, than the first sixty-four sine and cosine functions. For these reasons, it was decided that the design of the speech synthesizer should be based on the storage of data in the Walsh domain and resynthesis of the speech by performing the mathematics of equation (1-5).

In the last few years, research in speech synthesis by Walsh functions has been actively pursued by Bößwetter (6). His preliminary results, choosing dominant terms from among the first sixteen Walsh functions, and using a constant update time of 4 milliseconds, have shown that speech synthesis using one or two dominant terms per interval produces good sounding speech which is highly intelligible.

This previous work demonstrates the feasibility of synthesizing speech using the Walsh Transform. However, this device and others like it do not fully take advantage of the binary nature of Walsh functions in that the circuitry used is, to some degree, analog. The device to be described in this thesis is completely digital except for the digital-to-

analog converter on the output. Great efficiency and simplicity have been achieved by this method of design.

In the following chapters, the evolution of the design of the device is given, along with the results obtained. Chapter 2 deals with the mathematical fundamentals of the Walsh series and includes an algorithm for the generation of Walsh functions. Some mention is made of the properties of speech and how they influenced the design of the apparatus. Chapter 3 gives the system layout and the detailed design. The results obtained and a discussion of errors can be found in Chapter 4 while Chapter 5 discusses the potential uses of the device and what work is left to be done.

## CHAPTER 2

## A THEORETICAL BACKGROUND FOR ORTHOGONAL TRANSFORMATIONS

The previous chapter contained a brief outline of the mathematics involved in the design of a speech synthesizer based on the Walsh series. In the following pages, a more thorough examination of the mathematics is given, starting off with general theory of orthogonal functions followed by the specific cases of the sine and cosine functions, Walsh functions and Hadamard functions.* Finally, there is some discussion concerning those properties of speech that must be considered in adapting the Walsh series into a practical speech synthesizer.

### 2.1 Orthogonal Functions

A system of functions $\{f(n,x)\}$ is defined as being orthogonal in the interval $a \leq x \leq b$ if the following is true:

$$\int_a^b f(m,x)f(n,x)dx = C_m \delta_{m,n} \qquad (2\text{-}1)$$

where $\delta_{m,n} = 1$ if $m=n$

$0$ if $m \neq n$

$C_m$ is a constant.

If the constant $C_m$ is equal to unity for all m, then the system of functions is said to be orthonormal. A thorough discourse on orthogonal

---

* Hadamard and Walsh functions are related in a manner to be discussed later.

functions is given in a recent book by H.F. Harmuth (7).

An important property of orthogonal functions is that they are linearly independent. It is this property that leads to their usefulness in the series representation of a function. For instance, the function F(x) can be represented by a series of orthogonal functions as follows:

$$F(x) = \sum_{n=0}^{\infty} c(n)f(n,x) \tag{2-2}$$

This series representation is useful only if the coefficients c(n) can be evaluated. To determine a particular coefficient c(k), both sides of (2-2) are multiplied by f(k,x) and both sides integrated over the interval of orthogonality.

$$\int_a^b F(x)f(k,x)dx = \int_a^b \sum_{n=0}^{\infty} c(n)f(n,x)f(k,x)dx \tag{2-3}$$

$$= \sum_{n=0}^{\infty} \int_a^b c(n)f(n,x)f(k,x)dx$$

Due to the orthogonal nature of the functions, f(j,x), all terms on the right side of the equation will vanish except for the $k^{th}$ term. Therefore,

$$\int_a^b F(x)f(k,x)dx = c(k) \int_a^b f(k,x)f(k,x)dx \tag{2-4}$$

Furthermore, if the functions are orthonormal, the integral on the right

side will be unity. Thus, the value of the coefficient is:

$$c(k) = \int_a^b F(x)f(k,x)dx \tag{2-5}$$

The other coefficients can be evaluated in the same way. Once all the coefficients have been evaluated, the original function F(x) can be reconstructed using equation (2-2). Specific examples of systems of orthogonal functions will now be considered.

## 2.2 The Fourier Series

The Fourier series is based on the fact that the sine and cosine functions are orthogonal over the interval 0 to $2\pi$. This is evident from the following trigonometric relations:

$$\int_0^{2\pi} \sin(mt)\sin(nt)dt = \pi\delta_{m,n} \qquad n,m\neq 0 \tag{2-6}$$

$$= 0 \qquad n \text{ or } m = 0$$

$$\int_0^{2\pi} \cos(mt)\cos(nt)dt = \pi\delta_{m,n} \qquad n,m\neq 0 \tag{2-7}$$

$$= 2\pi \qquad m \text{ and } n = 0$$

$$\int_0^{2\pi} \cos(mt)\sin(nt)dt = 0 \tag{2-8}$$

Thus, the set of functions $\{\sin(mt), \cos(mt)\}$ are orthogonal over the

interval 0 to $2\pi$. Similarly, the set of functions

$$\left\{ \sin\left(\frac{2\pi mt}{T}\right) , \cos\left(\frac{2\pi mt}{T}\right) \right\}$$

are orthogonal over the interval 0 to T. This second set of functions is useful in a Fourier series for the reconstruction of a waveform having a basic period of T. The waveform may then be evaluated by using the following Fourier series:

$$f(t) = \sum_{n=0}^{\infty} a_n \sin\left(\frac{2\pi nt}{T}\right) + \sum_{n=0}^{\infty} b_n \cos\left(\frac{2\pi nt}{T}\right) \qquad (2\text{-}9)$$

Note that, for n=0, the sine function is zero and the cosine function is unity so equation (2-9) may be rewritten:

$$f(t) = b_o + \sum_{n=1}^{\infty} \left[ a_n \sin\left(\frac{2\pi nt}{T}\right) + b_n \cos\left(\frac{2\pi nt}{T}\right) \right] \qquad (2\text{-}10)$$

The term $b_o$ represents the DC offset to the signal f(t) since it is time independent.

It is now necessary to evaluate the coefficients $a_j$ and $b_j$. To determine $a_j$, both sides of equation (2-10) are multiplied by $\sin\left(\frac{2\pi jt}{T}\right)$ and integrated over the interval of orthogonality. Thus, the coefficient $a_j$ is given by:

$$a_j = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi jt}{T}\right) dt \qquad (2\text{-}11)$$

Similarly, the coefficient $b_j$ is given by:

$$b_j = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi jt}{T}\right) dt \qquad (2\text{-}12)$$

Finally, the coefficient $b_o$ must be evaluated. This is done by simply integrating both sides of equation (2-10) over the interval of orthogonality giving:

$$b_0 = \frac{1}{T} \int_0^T f(t)dt \qquad (2\text{-}13)$$

This represents the average value, or DC offset, of the signal f(t).

In summary, then, all periodic functions of time can be represented by a Fourier series. This series is a summation of sine and cosine waves of varying amplitudes having fixed phase and frequency relationships. The amplitude of each of the terms is given by their coefficients which are determined by equations (2-11), (2-12) and (2-13). Note that the summation is made up of two classes of functions, sines and cosines, which are odd and even respectively. It is evident that by introducing a phase term the summation can consist of only one of these classes:

$$f(t) = \sum_{n=0}^{\infty} a_n \sin\left(\frac{2\pi nt}{T} + \Psi\right) \qquad (2\text{-}14)$$

However, it is often convenient to keep the functions separated as in equation (2-10). In dealing with Walsh functions, it is also possible to keep the odd and even functions separated, as will be discussed in the next section. Other considerations make it practical not to do so,

however, and the Walsh series finally used will consist of only one class of functions.

## 2.3 Walsh Functions

As was mentioned in Chapter 1, there are several problems in the hardware implementation of the Fourier series. Thus, it is desirable to use another set of orthogonal functions more adaptable to a practical system. Such a set of functions are the Walsh functions.

Figure 2-1 illustrates the first sixteen Walsh functions. The functions alternate between +1 and -1 in value. The functions repeat after the basic time interval. Notice that they have been subdivided into two classes, SAL(n,θ) and CAL(n,θ) representing the odd and even functions respectively in accordance with Harmuth's notation (7). The functions can also be considered as being of one class of functions WAL(n,θ). The relationship between the two notations is:

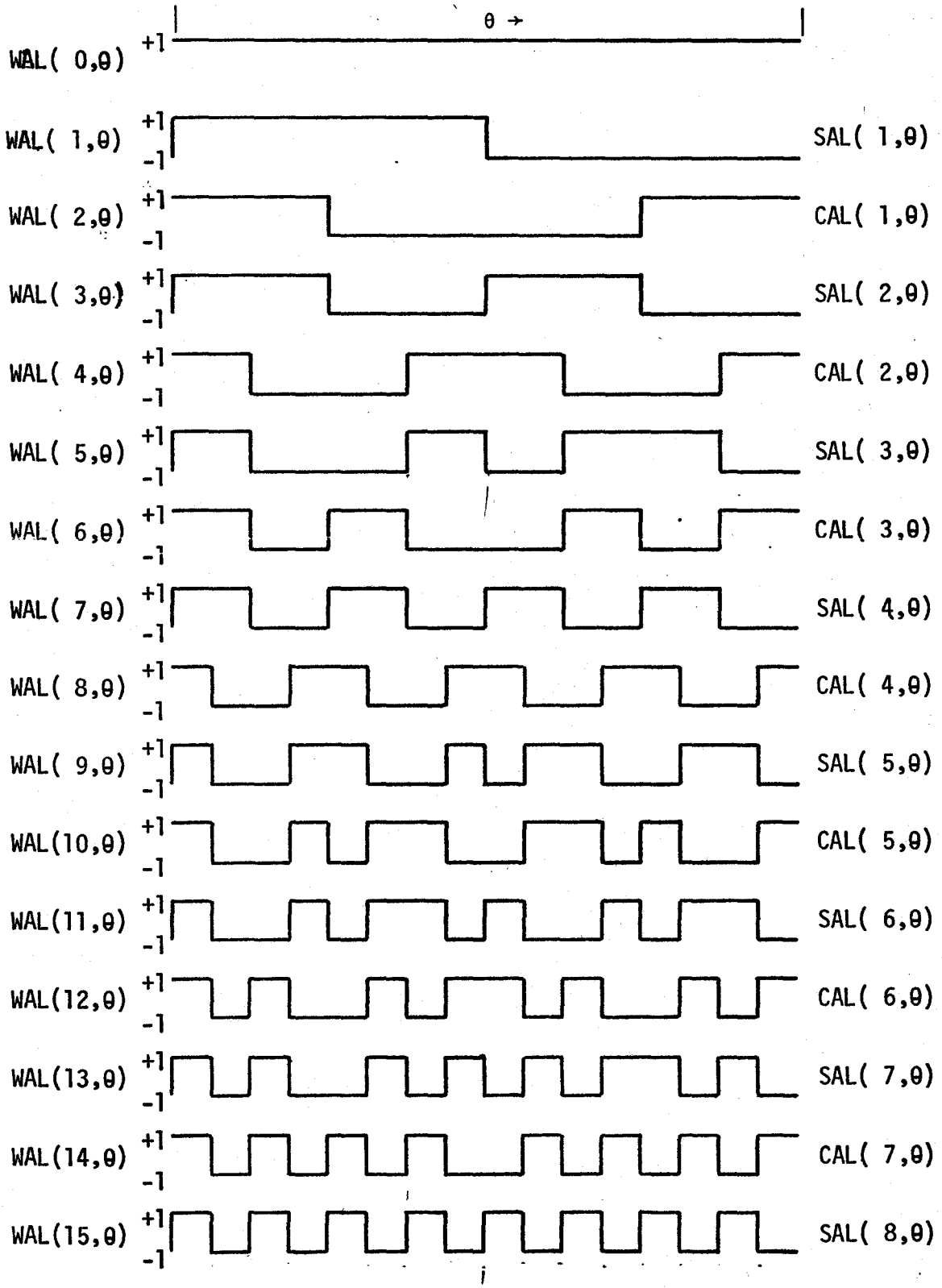
$$\mathrm{WAL}(2n,\theta) = \mathrm{CAL}(n,\theta) \qquad (2\text{-}15)$$

$$\mathrm{WAL}(2n-1,\theta) = \mathrm{SAL}(n,\theta)$$

$$n = 1, 2, 3, \ldots\ldots$$

It is obvious that, in general, the locations of the zero crossings of the Walsh functions are not evenly spaced. For this reason, the standard concept of period and frequency used for sine waves is not valid for Walsh functions. Instead, a new parameter, called sequency, is defined. This number, n, is defined as one-half the average number of zero crossings per second. The Walsh functions in Figure 1-1 are ordered by their sequency, from zero to eight. Note that the concept of sequency is difficult to apply to the WAL(n,θ) notation since adjacent pairs of the

Figure 2-1. First Sixteen Walsh Functions



functions WAL(n,θ) have the same sequency. In any event, for the CAL(n,θ) and the SAL(n,θ) notation, the first number listed is the sequency and the second number is the position variable, ranging from zero to one. For functions of time, θ can be defined as

$$\theta = \frac{t}{T}$$

where T is the interval of orthogonality, or the period of the Walsh functions.

Walsh functions are usually formed from the product of Rademacher functions (9). The $n^{th}$ Rademacher function is WAL($2^n$-1,θ). These functions are easily generated since they are simply square waves having $2^n$ zero crossings per interval. Using the following relation, one can obtain other Walsh functions from the Rademacher functions:

$$WAL(h,\theta) \times WAL(k,\theta) = WAL(h \oplus k,\theta) \qquad (2\text{-}16)$$

where ⊕ represents conversion to binary and a bit-by-bit modulo-2 addition and a conversion back.

The sole exception is WAL(0,θ) which is defined to be +1 throughout the interval. For example, consider the product of the second and third Rademacher functions, WAL(3,θ) and WAL(7,θ).

$$WAL(3,\theta) \times WAL(7,\theta) = WAL(3 \oplus 7,\theta) \qquad (2\text{-}17)$$

The ⊕ operation is carried out as follows:

$$3_{10} = 011_2$$

$$7_{10} = 111_2$$

$$3_{10} \oplus 7_{10} = 100_2 = 4_{10}$$

Therefore, $WAL(4,\theta) = WAL(3,\theta) \times WAL(7,\theta)$ (2-18)

Similarly, other Walsh functions can be generated. This technique can easily be adapted to a hardware realization (8). Figure 2-2 shows a typical circuit. The Rademacher functions are generated by a binary counter and the multiplication is carried out by exclusive-or gates.

The above technique becomes quite complicated when high order Walsh functions are to be generated. Thus, another approach must be developed. To this end, Siemens and Kitai (9) produced an algorithm for the generation of Walsh functions using a reverse-order Gray code to determine which Rademacher functions should be combined to produce the desired Walsh functions. For its implementation, the time interval T must be divided up into equal sub-intervals during which all Walsh functions under consideration will maintain a constant value. For instance, if the first eight Walsh functions are to be generated, the interval T will be divided up into eight sub-intervals labelled zero to seven. These are called the position numbers. Refer to Figure 2-3. The value of the $j^{th}$ Walsh function at position number k is found by manipulating the binary codings of the numbers j and k in the following manner:

1. Convert the Walsh number j to binary form.
2. Convert the binary coding of j to Gray code.
3. Reverse the order of the bits of this number.
4. Convert the position number k to binary form.
5. Form a bit-by-bit logical product of the numbers formed in steps three and four.

Figure 2-2. Walsh Generator Based on Products of Rademacher Functions

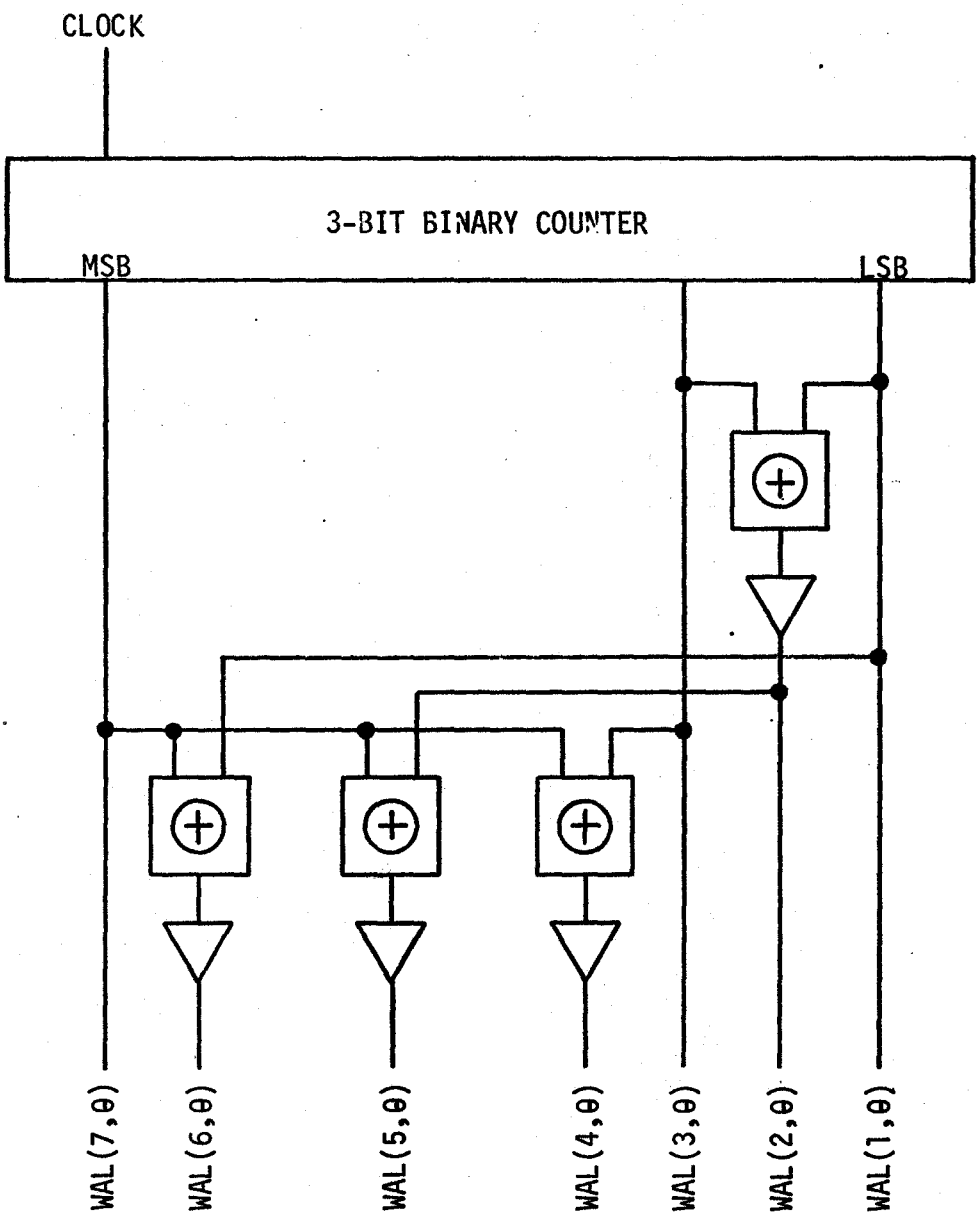


Figure 2-3. Illustration of Position Numbers for WAL(5,θ)

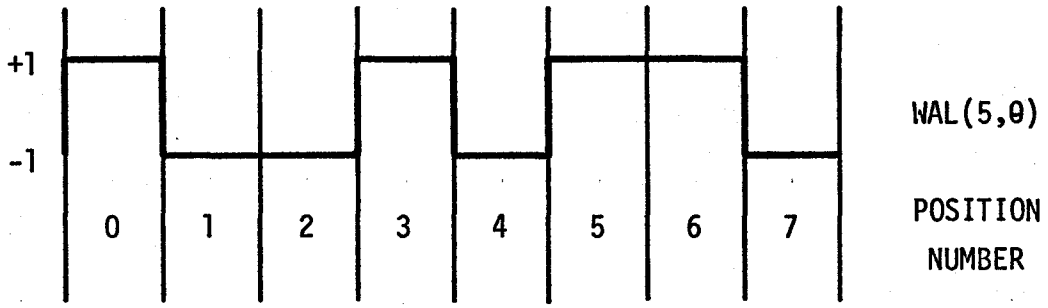



Figure 2-4. Binary to Gray Code Conversion

BINARY CODED NUMBER $X = a\ b\ c\ d \cdots$

GRAY CODED NUMBER $Y = e\ f\ g\ h \cdots$

CONVERSION:

$$e = a$$
$$f = a \oplus b$$
$$g = b \oplus c$$
$$h = c \oplus d$$
$$\vdots$$

where $\oplus$ represents modulo-2 addition

6. Calculate the parity of the number formed in step five.

7. If the parity is even, the value of the Walsh function is +1. If the parity is odd, the value is -1.

For a numerical example, the following is a calculation of the value of the fifth Walsh function at position number 3.

1. $5_{10}$ = 101 Binary Code - Walsh number
2. 101 = 111 Gray Code
3. 111 Bit order reversed
4. $3_{10}$ = 011 Binary Code - Position number
5. 011 Bit-by-bit logical product
6. Parity is even (there are an even number of ones)
7. The fifth Walsh function has value +1 at position number 3.

The binary to Gray code conversion is carried out by modulo-2 addition of the adjacent bits of the binary coded number. Figure 2-4 details this concept.

This algorithm adapts very well to both software and hardware implementation. A very simple Walsh function generator based on this algorithm is described in the next chapter.

## 2.4 The Walsh Series

The usefulness of Walsh functions in the series representation of a signal is based on the fact that they are orthonormal over the interval $0 \leq \theta < 1$. The orthonormality of the Walsh functions can be proven quite easily.

Consider equation (2-16). By integrating this equation over the

interval $0 \leq \theta < 1$, the following result is obtained:

$$\int_0^1 \mathrm{WAL}(h,\theta) \times \mathrm{WAL}(j,\theta)d\theta = \int_0^1 \mathrm{WAL}(h \oplus j,\theta)d\theta \qquad (2\text{-}19)$$

The right side of this equation represents the average value or DC offset of the function $\mathrm{WAL}(h \oplus j,\theta)$. Since all Walsh functions (except $\mathrm{WAL}(0,\theta)$) are made up of the product of Rademacher functions having an average value of zero, the average value of any Walsh function is zero except for $\mathrm{WAL}(0,\theta)$. Thus, the right side of equation (2-19) is zero except when $h \oplus j = 0$. If $h \oplus j = 0$, then $h = j$. Thus:

$$\int_0^1 \mathrm{WAL}(h,\theta) \times \mathrm{WAL}(j,\theta)d\theta = 0 \qquad h \neq j \qquad (2\text{-}20)$$

$$\int_0^1 \mathrm{WAL}(j,\theta)\mathrm{WAL}(j,\theta)d\theta = \int_0^1 \mathrm{WAL}(j \oplus j,\theta)d\theta \qquad (2\text{-}21)$$

$$= \int_0^1 \mathrm{WAL}(0,\theta)$$

$$= 1$$

The above is true since $\mathrm{WAL}(0,\theta)$ is equal to one throughout the interval zero to one. It is the only Walsh function to have a DC offset value. Thus, the Walsh functions are proven to be orthonormal over the interval $0 \leq \theta < 1$.

Being orthonormal, the Walsh functions can be used in a series expansion of a signal. This series is sometimes referred to as the

Walsh-Fourier series but in this thesis it will be designated the Walsh series and is defined as:

$$f(t) = \sum_{n=0}^{\infty} c(n)\,WAL(n,\theta) \tag{2-22}$$

Alternatively, the series can be written in terms of the odd and even functions, analogous to equation (2-10) for the Fourier series:

$$f(t) = a_0\,WAL(0,\theta) + \sum_{n=1}^{\infty}\left[a_n\,SAL(n,\theta) + b_n\,CAL(n,\theta)\right] \tag{2-23}$$

For simplicity, the form of equation (2-22) will be used. The coefficients c(n) of equation (2-22) can be derived by minimizing the mean squared error ($\overline{\varepsilon^2}$) between the function f(t) and its series representation.

$$\overline{\varepsilon^2} = \int_0^1 \left[f(t) - \sum_{n=0}^{\infty} c(n)\,WAL(n,\theta)\right]^2 d\theta \tag{2-24}$$

$$= \int_0^1 f^2(t)\,d\theta - 2\sum_{n=0}^{\infty} c(n) \int_0^1 f(t)\,WAL(n,\theta)\,d\theta$$

$$+ \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} c(n)c(m) \int_0^1 WAL(n,\theta)\,WAL(m,\theta)\,d\theta$$

From equation (2-20) and (2-21),

$$\int_0^1 WAL(n,\theta)\,WAL(m,\theta)\,d\theta = \delta_{n,m} \tag{2-25}$$

Therefore,

$$\overline{\varepsilon^2} = \int_0^1 f^2(t)d\theta - 2 \sum_{n=0}^{\infty} c(n) \int_0^1 f(t)WAL(n,\theta)d\theta + \sum_{n=0}^{\infty} c^2(n) \qquad (2\text{-}26)$$

To determine a specific coefficient c(j) minimize the mean square error with respect to c(j).

$$\frac{\partial \overline{\varepsilon^2}}{\partial c(j)} = -2 \int_0^1 f(t)WAL(j,\theta)d\theta + 2c(j) = 0$$

Therefore,

$$c(j) = \int_0^1 f(t)WAL(j,\theta)d\theta \qquad (2\text{-}27)$$

Thus, the coefficients can be evaluated by equation (2-27) and the original waveform f(t) reconstructed by applying equation (2-22).

The integral on the right side of equation (2-27) can be approximated by a finite series by sampling the signal f(t) a finite number of times. If the signal is sampled N times during the interval $0 \leq \theta < 1$ then the coefficient can be approximated by the following:

$$c(j) \simeq \frac{1}{N} \sum_{k=0}^{N} f_k WAL(j,\theta_k) \qquad (2\text{-}28)$$

where $f_k$ is the value of f(t) at the $k^{th}$ sampling instant and $WAL(j,\theta_k)$ is the value of $WAL(j,\theta)$ at the $k^{th}$ sampling instant.

This technique was applied to calculate, by computer, the coefficients of the first sixty-four Walsh functions for common waveforms

such as sines, cosines, triangle waves, ramps and rectangular pulses. The algorithm developed by Siemens and Kitai described earlier was used to software generate the Walsh functions required in equation (2-28). The results of the calculations are given in Appendix A.

## 2.5 Hadamard Functions

The question should be posed as to whether Walsh functions are the optimum set of orthogonal functions to be used in the series representation of a signal. The Siemens-Kitai algorithm for Walsh function generation mentioned earlier is seen to contain a great deal of code conversion. For instance, in step two of the algorithm, the Walsh function number is converted from binary code to Gray code. In the following step, there is a reversal of the order of the bits which is another code conversion. Since both these steps are really a one-to-one mapping from one set of numbers to another, it is obvious that, if functions were generated by the algorithm omitting steps two and three, these new functions would also be orthonormal. In fact, they would be Walsh functions that have been reordered. The second Walsh function in one system might become the third in another system, for instance. The code conversion was originally included in the algorithm so that the Walsh functions would be ordered in terms of their sequency. Although this is a useful property from a mathematical sense, its elimination can greatly simplify the generation of the orthonormal functions.

The new set of functions that are not ordered in terms of their sequency are called Hadamard functions. The first sixteen of these are

shown in Figure 2-5. If the Hadamard functions are represented in matrix form, with the rows representing the function numbers and the columns representing the position numbers, certain symmetries become evident, making the matrix easily generated by software techniques. Figure 2-6 shows a sixteenth order Hadamard matrix.

The Hadamard functions can also be generated using an algorithm similar to that used for Walsh functions:

1. Convert the Hadamard number j to binary form.
2. Convert the position number k to binary form.
3. Form a bit-by-bit logical product of these numbers.
4. Calculate the parity of the number formed in step three.
5. If the parity is even, the $j^{th}$ Hadamard function at position k is +1. Otherwise, it is -1.

Due to the similarity of the algorithm for the generation of Walsh and Hadamard functions, a single hardware generator, with suitable gating, can be used to generate both. Such a generator is described in the following chapter.

In summary, then, Hadamard functions are useful in the series representation of a signal due to the ease in which they can be generated, both in software and hardware. Walsh functions are also used since they are sequency ordered and thus more mathematically attractive.

## 2.6 Application of the Walsh Series to Speech Synthesis

The major problem in using the Walsh series in the generation of speech is the determination of the coefficients c(n) in equation (2-22).

Figure 2-5. First Sixteen Hadamard Functions

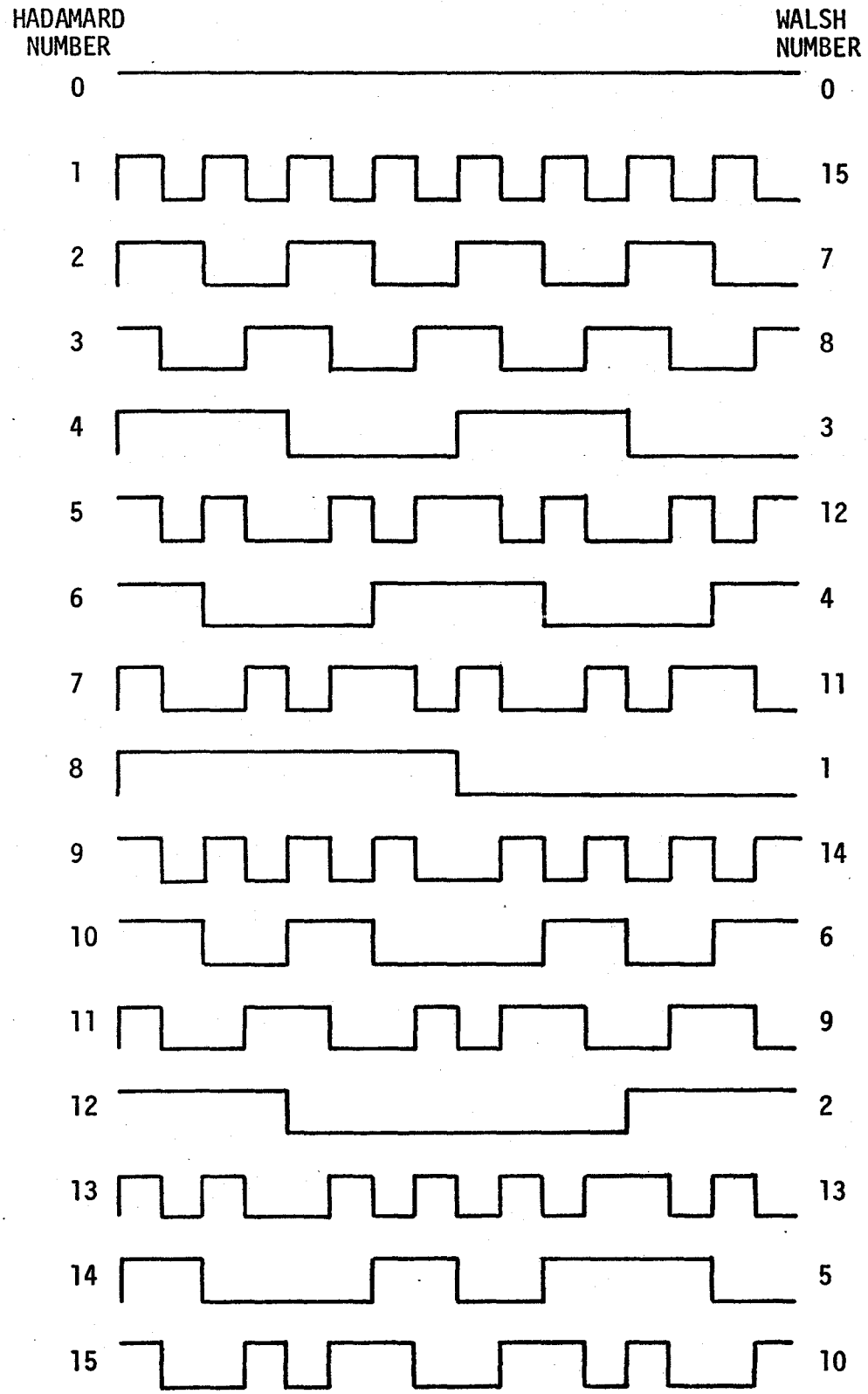


Figure 2-6. Sixteenth Order Hadamard Matrix

$$D = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} A & A \\ A & -A \end{bmatrix} \qquad C = \begin{bmatrix} B & B \\ B & -B \end{bmatrix} \qquad D = \begin{bmatrix} C & C \\ C & -C \end{bmatrix}$$

For basic functions that can be described mathematically, such as sines, ramps and triangle waves, equation (2-27) can be applied directly to evaluate the coefficients. However, speech waveforms are generally more complicated and cannot be described mathematically. Therefore, a sampling technique must be used. The amplitude of the waveform is sampled regularly and, by using algorithms such as the Fast Walsh Transform, the coefficients can be evaluated. The Fast Walsh Transform is analogous to the Fast Fourier Transform. The latter is used to evaluate the coefficients needed for the Fourier series representation of a signal (10).

Using the Fast Walsh Transform, N samples of the signal are used simultaneously to generate N coefficients. The process is continuously repeated to generate other sets of N coefficients. This is necessary because generally speech waveforms are not periodic. Thus, the coefficients will change in value with time and must be updated by the algorithm. These sets of N coefficients may then be applied to equation (2-25) to generate the output signal.

The question obviously comes to mind as to how often should the sets of coefficients be updated. Also, how large should N be? Both of these factors will affect the accuracy of the generated output signal. If this signal is changing form rapidly, then a high update rate is needed. If the analysis of the signal shows that there are many high sequency components, then the value of N should be large. Both the update rate and the value of N will affect the rate at which the original signal is sampled.

Obviously, the determination of the update rate and the value of

N will be dependent on the nature of the speech waveform being analysed. This poses a difficult problem since very little speech analysis in the Walsh domain has been carried out. For this reason, the determination of these parameters is largely based on educational guess-work and some preliminary results of other researchers at McMaster University*. The hardware device described in the next chapter was designed to be flexible enough to allow relatively easy modification of the update rate and the value of N.

Recall also that not all N coefficients need be used in the generation of the speech waveform. Data compression is carried out by choosing only the M most dominant coefficients, in absolute value. Thus, equation (2-25) becomes:

$$f(t) \approx \sum_{n=1}^{M} C_{k_n} \, WAL(k_n, \theta) \qquad (2\text{-}29)$$

where $C_{k_n}$ is the $n^{th}$ most dominant coefficient selected from a set of the first N coefficients. The value of M must also be chosen, based on the characteristics of the speech waveform.

To select an appropriate value for N, a 'worst-case' speech waveform should be used. Analogous to the square wave requiring many terms to be reproduced accurately by a Fourier series, a sine wave requires many Walsh function terms to be reproduced accurately by a Walsh series. The vowel

---

* Y. Y. Shum and A. R. Elliott are developing a Fast Walsh Transform working on real-time analog signals with the capabilities of varying the number of samples N, as well as the time interval for the update. This will be applied to the analysis of speech.

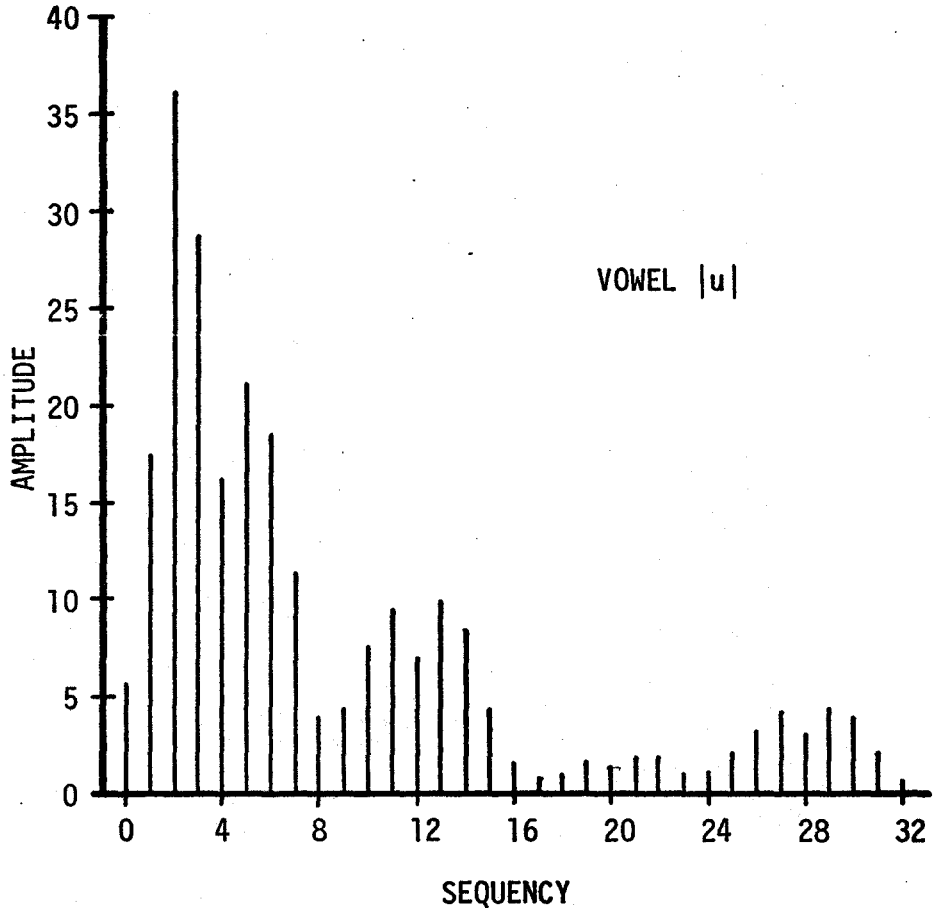
sounds such as |u| most closely approximate a sine wave. A sequency analysis of this sound is given by Campanella and Robinson (11) and their results, shown in Figure 2-7, illustrate that the coefficients of Walsh functions of high order are generally quite small. Thus, if N is set to be sixty-four, this should be adequate to represent most speech waveforms.

The update frequency is a compromise between accuracy of the generated waveform and the number of coefficients that must be produced in a given interval of time. As mentioned in the previous chapter, the sampling theorem implies a minimum update rate of 10.67milliseconds for a signal of bandwidth 3kHz. For flexibility, a factor of 2 was included in the hardware synthesizer leading to an update rate of 5milliseconds. A possibility to be considered at this time is the use of a variable update frequency. If the speech waveform were rapidly changing, such as it would with the hard consonant sounds, a high update frequency would be used. For vowel sounds, a lower frequency would be used. While this adds slightly to the complexity of the system, the saving in data needed to produce the sounds could be substantial. However, there is a basic limitation associated with using a variable update frequency which is discussed in section 4-1. For simplicity, a variable sample rate feature was not implemented in the hardware and a fixed rate of 5milliseconds was used.

With N = 64 and an update rate of 5milliseconds, the Fast Walsh Transform will require the original waveform to be sampled at a rate of 12.8kHz $(64/(5 \times 10^{-3})\text{Hz})$.

The next parameter to be evaluated is M. Again referring to the

Figure 2-7. Sequency Analysis of the Vowel Sound |u|




Reproduced by permission of S. J. Campanella

'worst-case' signal of the vowel |u| it is seen that the largest six coefficients have substantially greater amplitudes than the remainder. Again, leaving a margin of flexibility, it was decided that the eight most dominant coefficients for speech sounds analysed in the Walsh Domain (see Campanella and Robinson (11)) would probably be sufficient for intelligible speech reproduction. This conclusion is supported by independent speech synthesis accomplished by Bößwetter (6) using one or two cominant terms.

Finally, the range in the value of the coefficients must be chosen. Both positive and negative valued coefficients can be used. Within the field of sixty-four Walsh coefficients, the system presented here can only use the eight most dominant. Within this set of eight most dominant terms the ratio of the absolute value of the most dominant coefficient to the least dominant coefficient (ignoring any zero valued coefficients) for some basic waveforms is listed in Table 2-1. As can be seen, the maximum ratio is 48.5 to 1. Thus, again including a margin of flexibility, a ratio of 64 to 1 was chosen for the coefficients. They can have any integral value from -63 to +63.

In summary, the waveform synthesizer will be required to calculate the summation of equation (2-29) once every five milliseconds. The summation will consist of a maximum of eight terms selected from the first sixty-four Walsh functions. If only six terms are actually needed, for example, the remaining two can be set to zero with no detrimental effect. The value of $C_{k_n}$ can vary from -63 to +63. The next chapter discusses

Table 2.1 Ratio of Walsh Coefficients for Basic Waveforms

| Waveform | Most Dominant Coefficient (Abs. Val.) | Least Dominant Non-Zero Coefficient (Abs. Val.) | Ratio X (X:1) |
|---|---|---|---|
| Sine | 63 | 1.3 | 48.5 |
| Triangle | 48 | 1 | 16 |
| Ramp | 32 | 1 | 32 |
| Speech $\|u\|$ | 36 | 10 | 3.6 |

the hardware implementation of such a synthesizer.

# CHAPTER 3

## SYSTEM DESIGN AND IMPLEMENTATION

In order to generate the series of equation (2-29), subject to the constraint of using only the dominant eight terms of the first sixty-four Walsh functions, the design must be able to handle programable Walsh function inputs, multiply them by their coefficients, and add these terms together. It would be advantageous to do the total operation digitally since it has been presumed that the coefficients will be produced in binary form by a computer. Assuming a programable Walsh function generator exists, the system of Figure 3-1 will generate the required Walsh series. Each of the eight Walsh functions are multiplied by their coefficients and the results are summed in a binary adder. This sum is converted from a binary number to the required amplitude of the waveform by the digital-to-analog converter. The process is repeated each time any of the Walsh functions changes value so the output can be updated. The coefficients themselves would be periodically updated by the driving computer so the output will reproduce the required waveform.

Although this system is attractive in that it is entirely digital, it is not very efficient in that eight distinct Walsh function generators and multipliers are required. In the interests of economy, a multiplexing system, which would use one Walsh function generator and one multiplier to sequentially produce the required terms of the sum, is desirable. It now becomes necessary for the binary adder to have a memory element in it

Figure 3-1. Digital Waveform Synthesizer

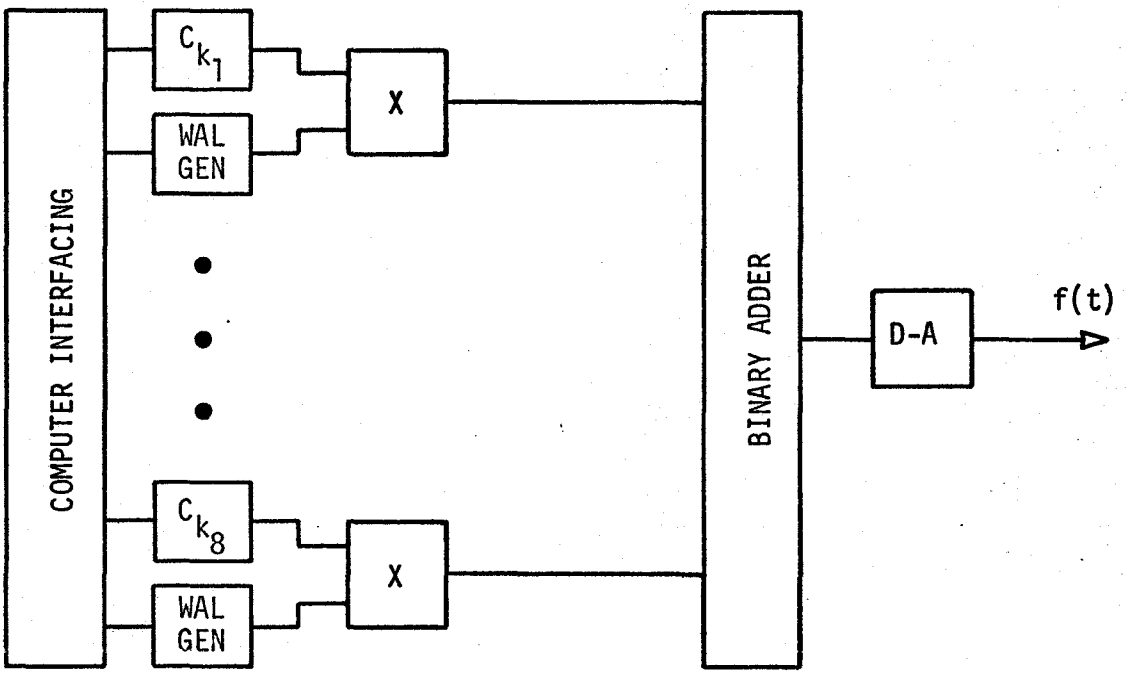
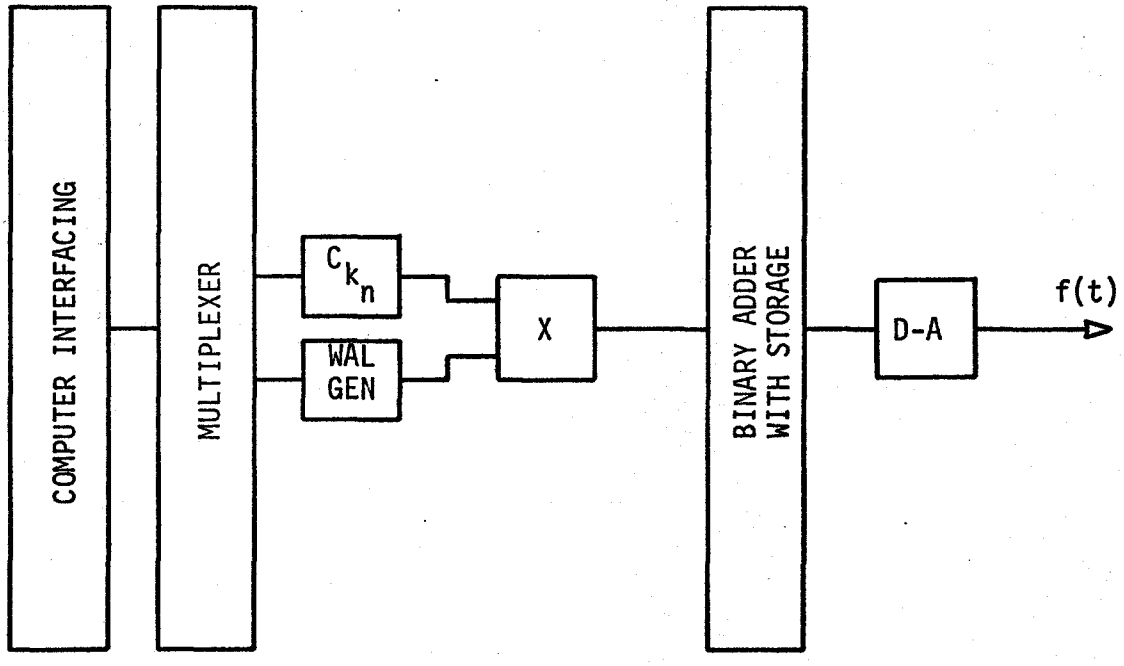



Figure 3-2. Multiplexed Digital Waveform Synthesizer



to store the partial sums as each additional term is being calculated. Such a system is shown in Figure 3-2.

This system is somewhat simpler than the first in that only one Walsh function generator and one multiplier are needed. On the other hand, it is made more complex in that some sort of storage element is needed in the adder to hold the partial sum as each additional term is added sequentially by the multiplexer. The multiplexer itself will also have to store eight words containing the desired eight coefficients and their corresponding Walsh function numbers. Control logic to sequentially select the coefficients is also necessary.

The other consideration with the multiplexed system is speed. It takes eight times as long to arrive at the sum to send to the D/A converter with the multiplex system than with the first system. However, since only audio waveforms are being generated, this is not a serious problem. More mention will be made of the timing problem in multiplex systems later in this section.

The major drawback to the system of Figure 3-2, however, is the multiplier. A circuit to multiply two binary numbers of arbitrary sign together is generally quite complex and slow. Great simplification can be obtained with the realization that the absolute value of the Walsh function is always unity. Thus, instead of multiplying the two numbers together, the Walsh function and sign of the coefficient can be logically combined. If the Walsh function and coefficient have the same sign, then the absolute value of the coefficient can be directly added to the partial

sum in the binary adder. If the signs are different, the absolute value of the coefficient should be subtracted from the partial sum in the adder. This system is shown in Figure 3-3.

The complexity of the adder circuitry has increased substantially from the first system. It would be desirable to modify this circuitry in order to simplify the system. One technique to do this is to convert the absolute value of the coefficient into a series of pulses. For instance, if $\left|C_{k_n}\right| = 37$, then thirty-seven pulses would be produced. These pulses could then be simply counted in a binary up/down counter. The direction of the count would again be determined by the Walsh function generator and the sign of the coefficient. By using such a system, the complex adder/subtractor with memory is replaced by a binary up/down counter, whose circuitry inherently contains the required memory.

A circuit is now needed to convert the absolute value of the coefficients into a string of pulses. Such a circuit is the rate multiplier to be described in section 3.3. Basically, the rate multiplier is a 'black box' having a clock input, a series of k control lines and an output. A binary number (in this case $\left|C_{k_n}\right|$ ) is placed on the control lines, and a clock signal of frequency $f_{IN}$ is placed on the clock input. The output consists of a series of pulses having frequency $f_{OUT}$.

$$f_{OUT} = f_{IN} \times \frac{\alpha}{\beta} \qquad (3\text{-}1)$$

where $\alpha$ is the binary number on the k control lines and $\beta = 2^k$.

For instance, if a six bit rate multiplier is fed with a 1kHz clock input and the number sixteen is placed on the control lines, the

Figure 3-3. Multiplexed Digital Waveform Synthesizer Without Multiplier

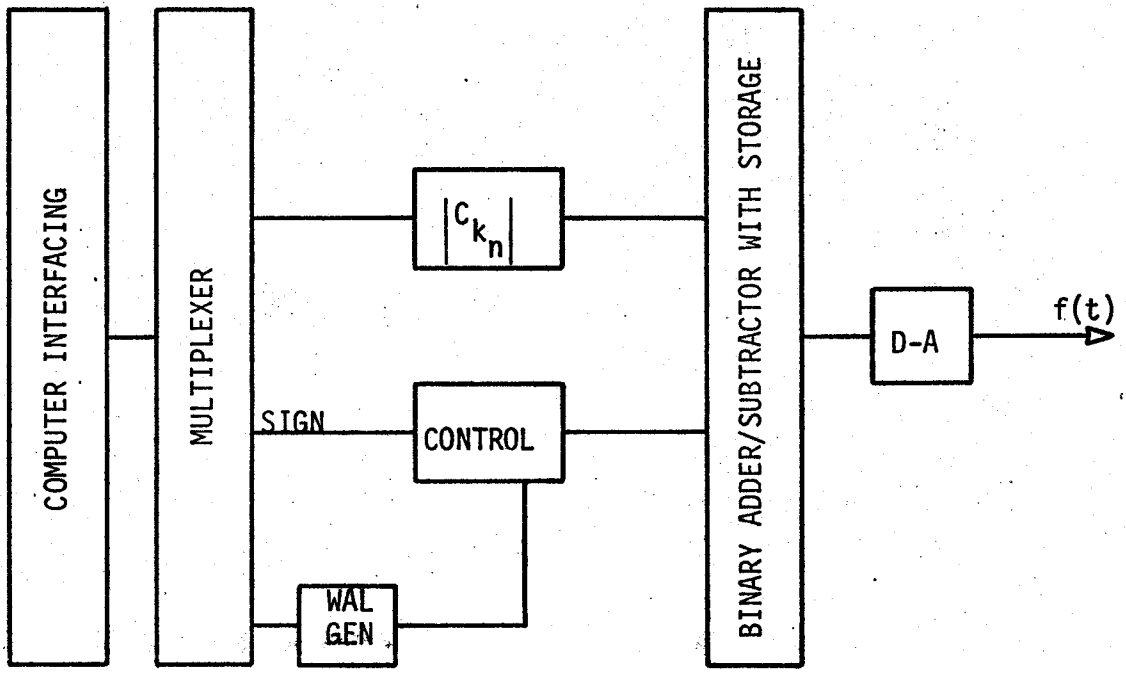
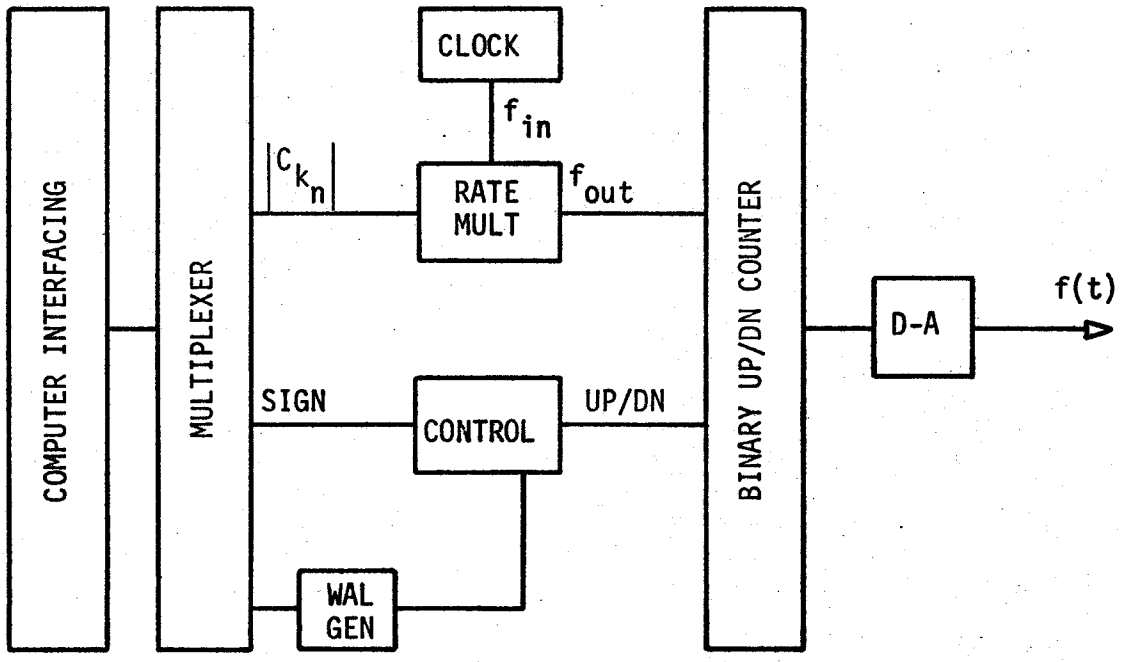



Figure 3-4. Multiplexed Digital Waveform Synthesizer Using a Rate Multiplier



output frequency would be:

$$f_{OUT} = 1\text{kHz} \times \frac{16}{64} \qquad (3\text{-}2)$$

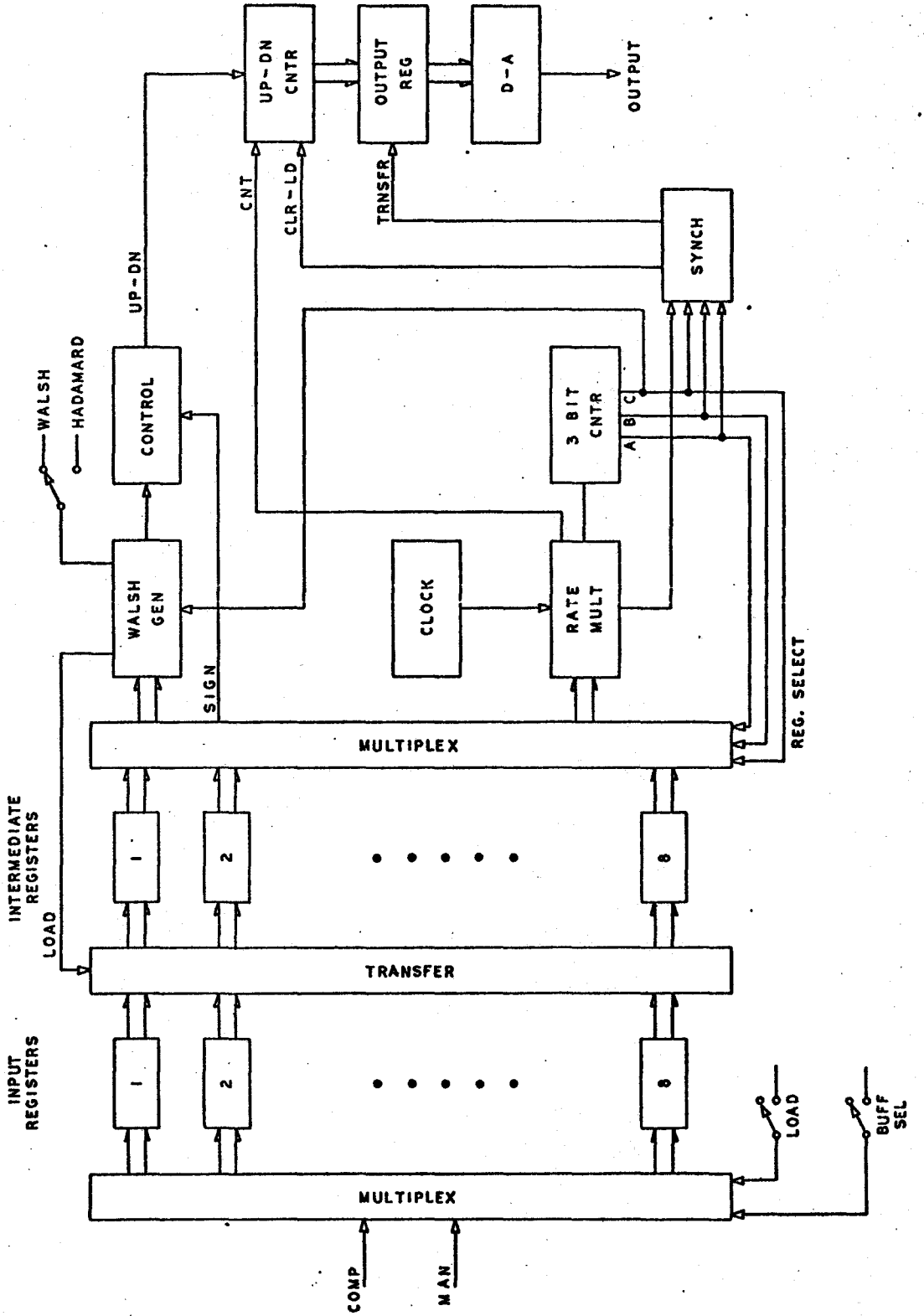
$$= 250\text{Hz}$$

Thus, by correct clocking of the rate multiplier, the output will consist of a series of pulses bearing a direct relationship to the binary number placed on the control lines. A system of waveform synthesis using a rate multiplier and an up/down counter is shown in Figure 3-4. This is essentially the layout of the synthesizer to be discussed in the following pages. A more detailed diagram of the system layout, with additional timing and interfacing circuitry, is given in Figure 3-5.

The eight input registers are loaded sequentially either directly from the computer or manually through the use of panel switches. A multiplexer is used to determine which register is being loaded with the data. The multiplexer can be computer controlled or advanced with the 'buffer select' switch in the manual mode.

At the beginning of the interval between updates of the coefficients, the Walsh function generator is set to the zero position number and a LOAD signal is sent to the registers. This transfers the data in the input registers to a set of intermediate registers. Thus, the loading of the input registers can be carried out entirely independently of the rest of the device as long as all eight input registers have been loaded in the time between updates of the coefficients. This length of time (which is 5 milliseconds for the system) will be referred to as the basic time

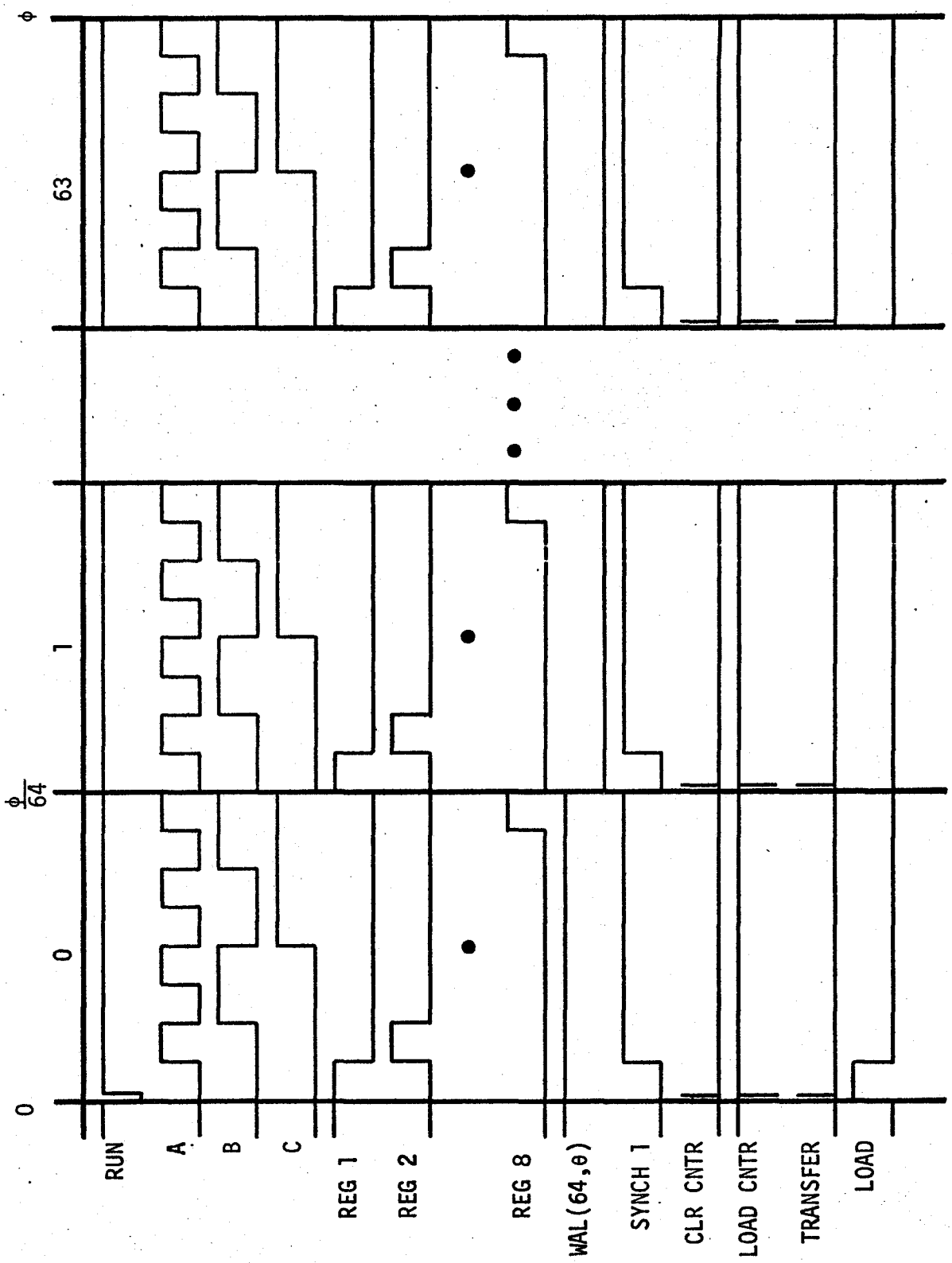
Figure 3-5. Waveform Synthesizer System Layout



interval and will be denoted by $\phi$.

To more easily understand the operation of the instrument, refer to the timing diagram, Figure 3-6. A multiplexer connected to the output of the intermediate registers is initially set to connect register one to the following circuitry. The coefficient in register one is applied to the rate multiplier which in turn provides from zero to sixty-three pulses in the interval $\phi/(64\times8)$. These are then fed to the up/down counter. The direction of the count is determined by the sign of the coefficient and the value of the Walsh function. At the end of this interval, a pulse is sent from the rate multiplier to the three bit counter, changing the levels on lines A, B and C. This advances the multiplexer to register 2 and the process is repeated. Finally, after an interval $\phi/64$, all eight registers will have been sampled and the up/down counter will contain the correct sum to be converted to the output signal. A synch pulse is derived from the states of lines A, B and C and results in the contents of the up/down counter being transferred to the output register, which in turn is connected to the digital-to-analog converter used to produce the analog signal f(t). At the same time, the counter is cleared so that a new sum can be evaluated. Finally, a signal is sent to the Walsh function generator to advance it to the next position number. The eight intermediate registers are again sequentially sampled to produce a new sum after another interval of $\phi/64$. This process is repeated a total of sixty-four times during the basic interval. Thus, sixty-four sums are evaluated and the output is updated sixty-four times during $\phi$. This is necessary since the highest order Walsh function to be generated will change value sixty-four times in the

Figure 3-6. Timing Waveforms for the Synthesizer



basic interval. At the end of the interval, a LOAD pulse is derived which transfers new data from the input registers to the intermediate registers and the whole process is repeated.

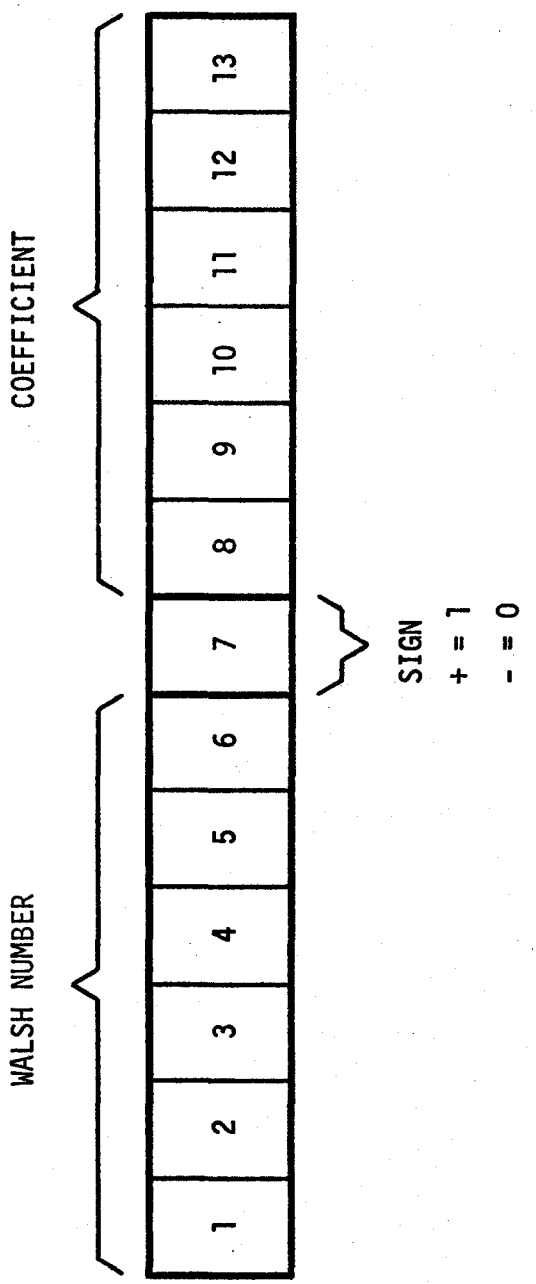
In summary, then, the basic interval $\phi$ of 5milliseconds is divided into sixty-four sections. A new sum is evaluated at the end of each of these sections (every 5/64milliseconds or about 78microseconds) and the output is updated. These sections are in turn subdivided into eight subsections of approximately 9.8microseconds, each devoted to one of the eight intermediate registers. During the interval of one of these subsections, up to sixty-four pulses must be provided by the rate multiplier. Thus, the period of the input waveform must be one sixty-fourth of the length of the subsection. This leads to a period of approximately 153nanoseconds or a frequency of 6.6MHz. The actual ratio of the interval time $\phi$ and the input clock period is $2^{15}$ or 32,768.

The following sections will deal individually with each part of the system in more detail than given above.

## 3.1 Input Registers and Multiplexing

Being a programable waveform synthesizer, the machine must have facilities for interfacing to a computer. For maximum flexibility, it was decided to include a set of buffer registers which could be loaded from the computer on a time basis completely independent of the rest of the speech synthesizer. The synthesizer itself would have to have a set of buffers also from which the appropriate coefficients could be obtained to generate the output signal. Data would be transferred from the first

Figure 3-7. Input Word Organization



set of registers to the second once every five milliseconds by a LOAD signal generated at the end of each basic interval by the synthesizer.

The data to be held in the registers must include a six-bit Walsh function number, a six-bit coefficient number and a sign bit. The word layout is shown in Figure 3-7.

As an optional feature, it is desirable to be able to load the registers manually. This facilitates check-out and allows the generation of basic waveforms whose coefficients can be determined mathematically as described in the previous chapter. For this reason, a set of thirteen switches, representing the thirteen bits of the data word, is included in the synthesizer.

Figure 3-8 shows the input registers and the input gating for the data. Data is fed to the input registers from either the thirteen switches, S1-S13, or the thirteen computer lines, C1-C13, depending on the position of the 'Mode' switch. The D-type flip-flops of the input registers are sequentially clocked by signals L1-L8 generated by the multiplexing circuitry. This is carried out until all eight registers have been filled. Note that the timing for the loading of the input registers is determined solely by the signals L1-L8. These, in turn, are controlled either manually, if in the manual mode, or by synch signals from the computer. Thus, the loading timing is independent of the rest of the synthesizer circuitry.

Signals L1-L8 are generated by the multiplexing circuitry shown in Figure 3-9. The recirculating shift-register has eight outputs, one

Figure 3-8. Input Registers and Data Control

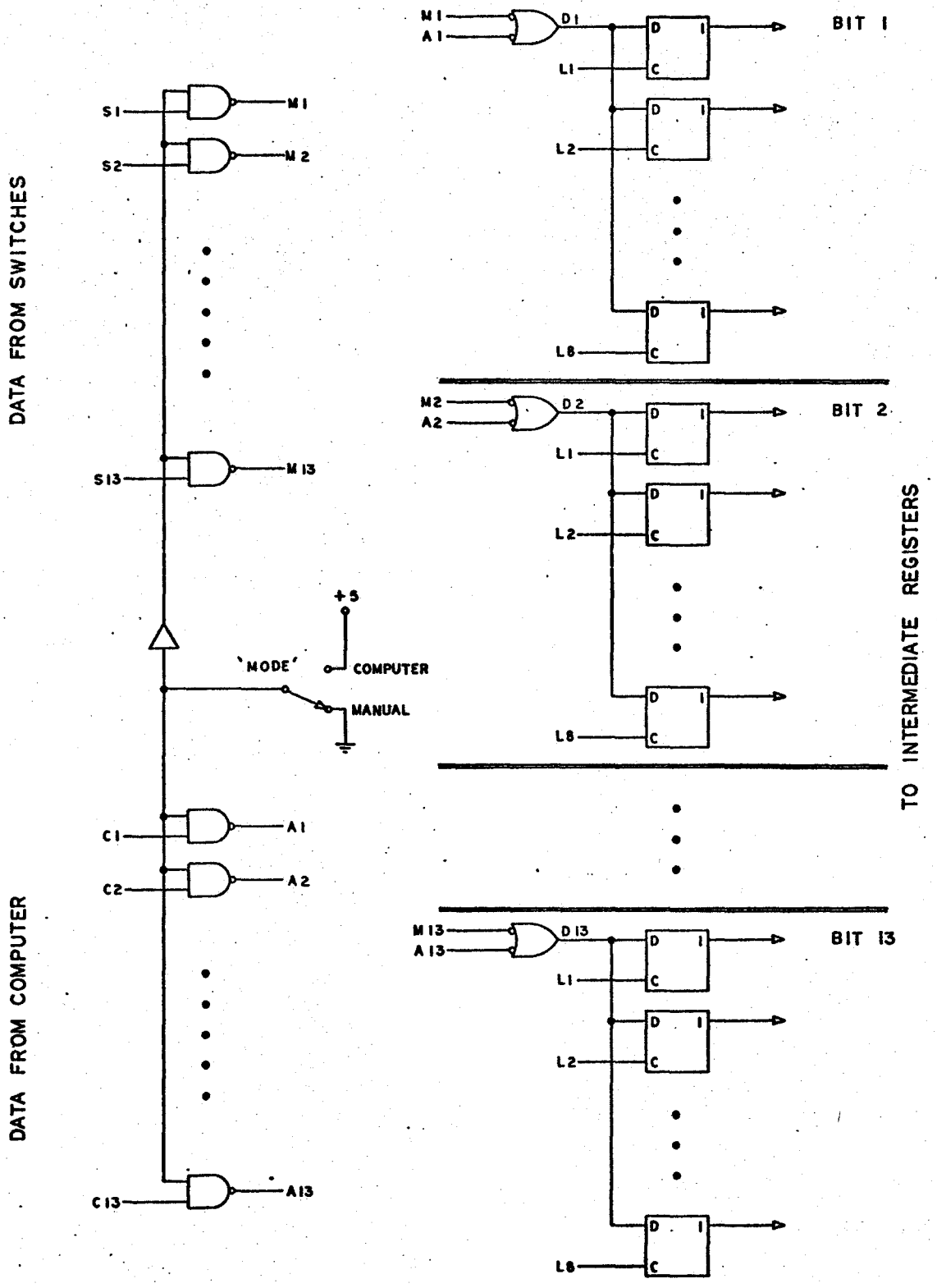
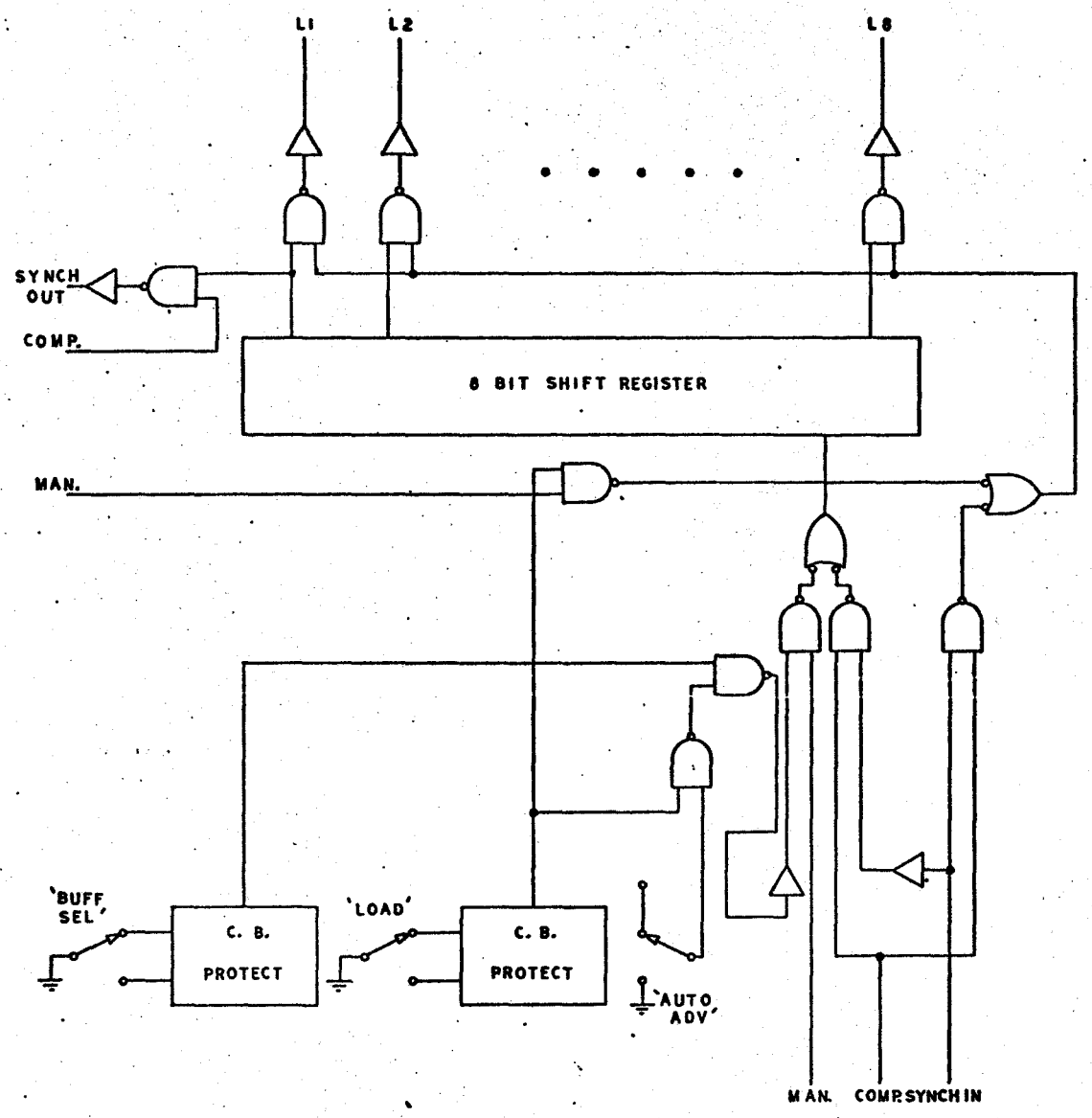
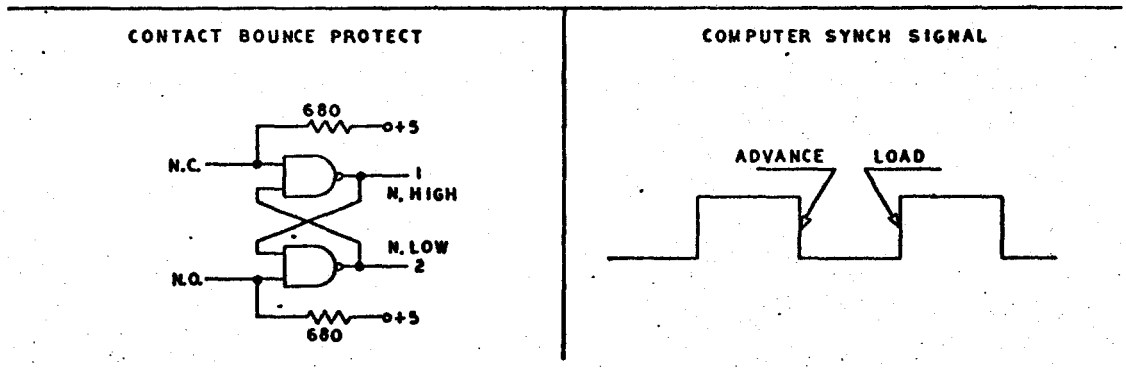


Figure 3-9. Input Multiplexing



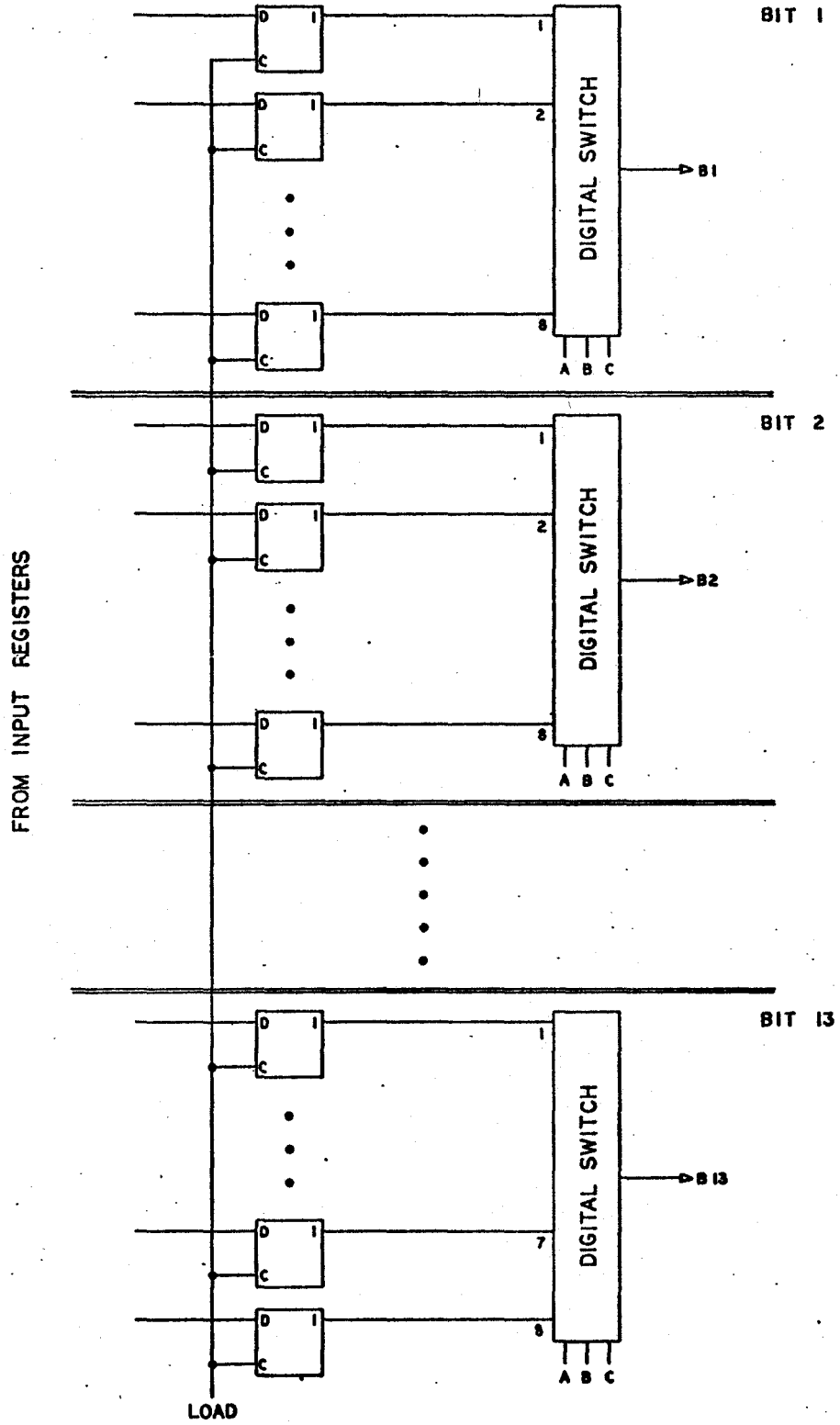
and only one of which is high at any given time. These output lines can be used to select which of the eight input registers is to be loaded. In the computer mode, the falling edge of a SYNCH signal will advance the shift-register. In the manual mode, two methods can be used to advance the shift-register. The push-button switch 'Buffer Select' will directly advance the shift-register when pressed, or, if the 'Auto Advance' switch is on, the 'Load' switch will also advance the register. The advance will take place on the falling edge of the signal. The automatic advance feature is useful when loading data sequentially into the registers. Pressing the 'Load' switch will load the input register and releasing the switch will automatically advance the shift-register so loading of the next input register can take place.

After the shift-register has selected the correct input register, the loading takes place by strobing the lines L1-L8. This is done in the manual mode by the 'Load' push-button or in the computer mode by the SYNCH signal from the computer.

When the eight registers have been loaded, the shift-register will be in the state with its first output line high. This line is monitored and forms an outgoing SYNCHOUT signal to the computer to signal that all the registers have been loaded.

Figure 3-10 shows the intermediate registers and the multiplexing used to sequentially connect them to the synthesizer circuitry. The data inputs of the intermediate register flip-flops are connected to the outputs of the input registers. The data is strobed into the intermediate

Figure 3-10. Intermediate Buffer Registers and Multiplexing



registers once every 5milliseconds (the update rate) by the LOAD signal, derived from the Walsh function generator. Note that if the data inputs are changed while the LOAD line is activated the outputs of the intermediate registers will also change. For this reason, loading of the input registers should not occur when transfer of data is taking place. The LOAD line could be monitored by the computer as a BUSY signal to prevent the computer from sending new data to the input registers during this interval.

The digital switches shown in Figure 3-10 are used to select the required one of eight intermediate registers. The level on the output line will be the same as the level on one of the input lines. Levels on A, B and C form a binary number from zero to seven that determines which of the eight inputs is monitored by the output line. By connecting A, B and C to a three-bit binary counter, therefore, inputs zero through seven will be sequentially 'connected' to the output line. Thus, the thirteen switches, all controlled by lines A, B and C, are used to sequentially connect intermediate registers one through eight to the synthesizer circuitry. Signals A, B and C are derived from the clock chain to be described in a later section. Figure 3-6 shows their relative timing and how the eight intermediate registers are sequentially monitored. It can be seen that lines A, B and C toggle through their eight states sixty-four times during the basic interval.

## 3.2 Walsh and Hadamard Function Generator

The Walsh and Hadamard function generator is based on the algor-

ithms discussed in sections 2-3 and 2-5. Due to the similarity of the algorithms, one generator can be used to generate both the Walsh and Hadamard functions. Suitable gating is used to select the correct output.

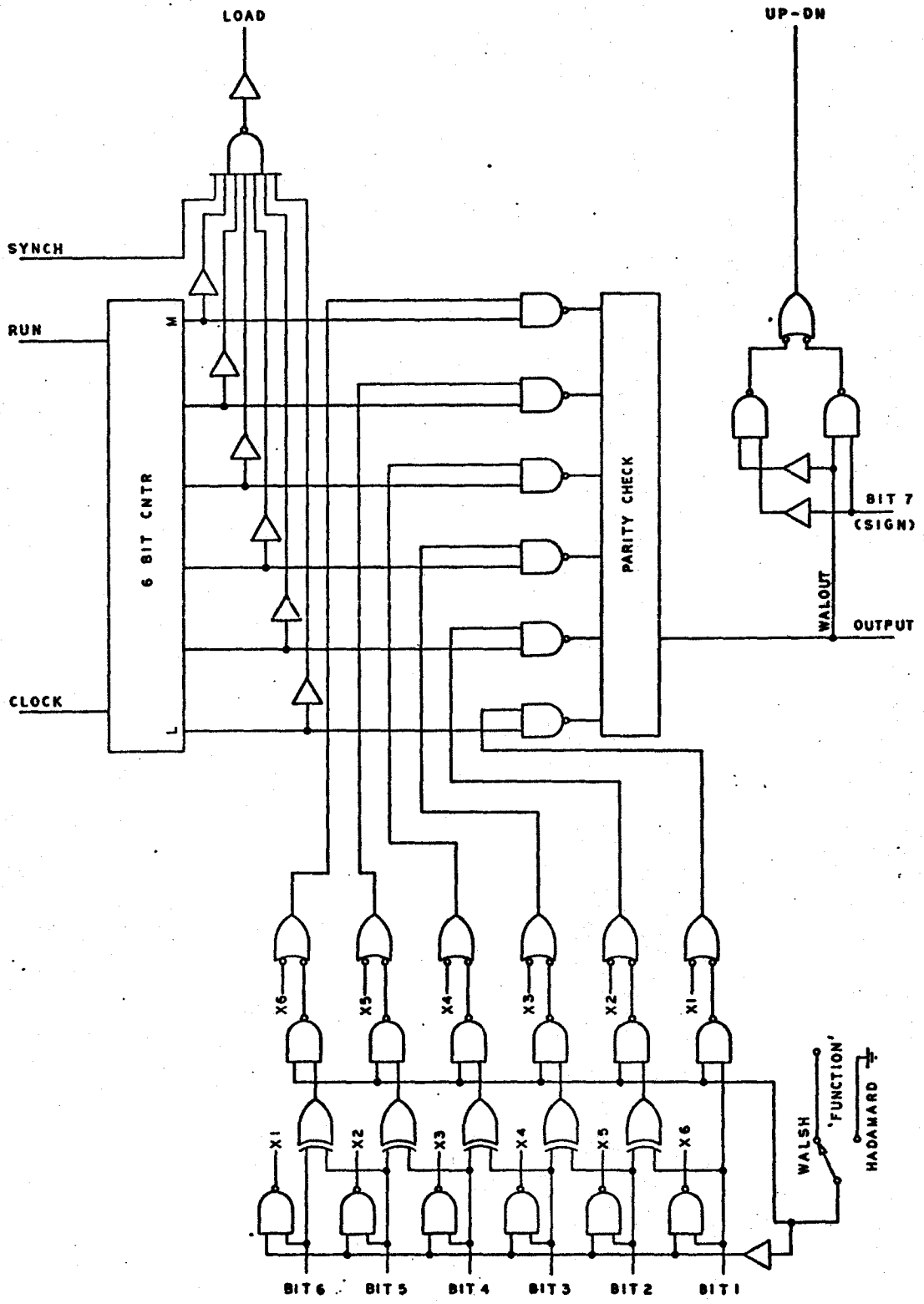
Since the position numbers are sequentially incremented from zero to sixty-three, a six-bit binary counter can be used to store the position number. Since the counter must be incremented sixty-four times during the basic interval, the signal on line C serves as a clock pulse to update the generator.

The binary coding for the Walsh (or Hadamard) function number is available from the output lines of digital switches one through six. For the Hadamard functions, the logical product of the function numbers and position numbers must be taken. This is done with six two-input NAND gates. Figure 3-11 shows the circuitry of the Walsh-Hadamard generator. The parity is monitored by a six-bit parity checker, which is essentially a series of exclusive-OR gates contained in one integrated circuit package.

The choice of type of function is made with a function selector switch. This controls the gating network shown at the bottom of Figure 3-11.

For Walsh functions, it is necessary first to convert the incoming binary coded number to a Gray code. This is done with five exclusive-OR gates performing modulo-2 addition as required by the algorithm of Figure 2-4. Finally, the order of the bits must be reversed before the logical product can be taken.

Figure 3-11. Walsh-Hadamard Function Generator



The six-bit binary counter, whose state represents the position number, can be reset to position zero with the RUN line. This is done every time the machine is started or stopped in order to reset all the circuitry to its initial value.

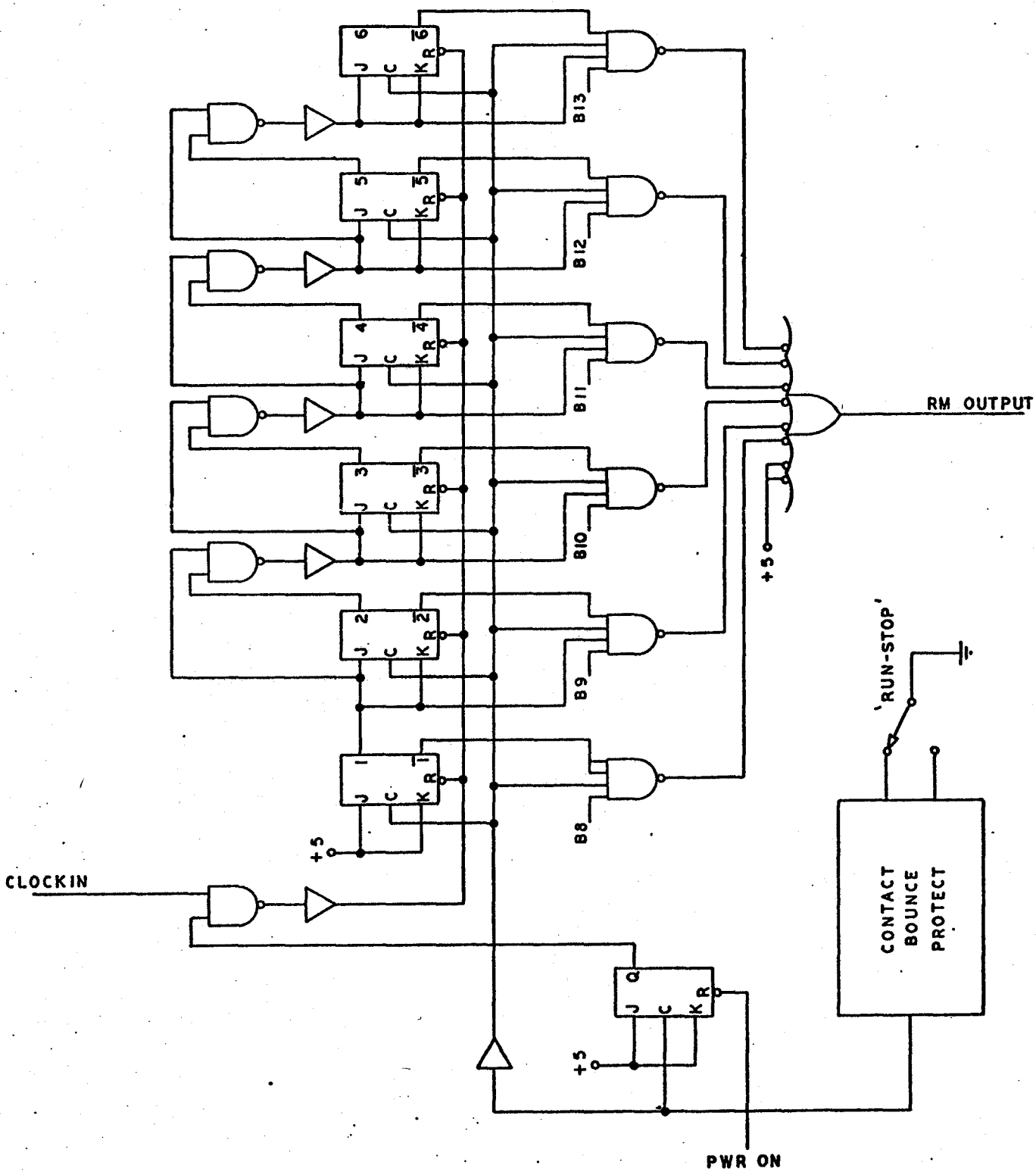
At the end of the basic interval, the six-bit counter will have returned to state zero. At this time, a new set of data must be loaded into the intermediate registers so that the output speech waveform can be updated. The LOAD signal to do this is generated by monitoring the counter for the zero position number. Then, the short SYNCH 1 pulse, derived from the clock chain, will be gated through to the clock inputs of the intermediate registers and the new data will be transferred.

As mentioned earlier, the Walsh-Hadamard function is used in conjunction with the sign of the coefficient to determine if the output binary up/down counter is to count up or count down. The output of the function generator is high to represent a functional value of +1 and zero to represent a value of -1. By combining this output with the coefficient sign bit, a signal is derived giving the direction of count for the output up/down counter. The three gates at the output of the function generator are used to derive the UP/DN signal which is used to control the output binary up/down counter.

### 3.3 Binary Rate Multiplier

Figure 3-12 shows the circuitry of the binary rate multiplier used in the synthesizer. It consists of a six-bit binary counter and a system of output gating. The counter can be reset at the beginning of operation by the RUN line.

Figure 3-12. Six Bit Rate Multiplier

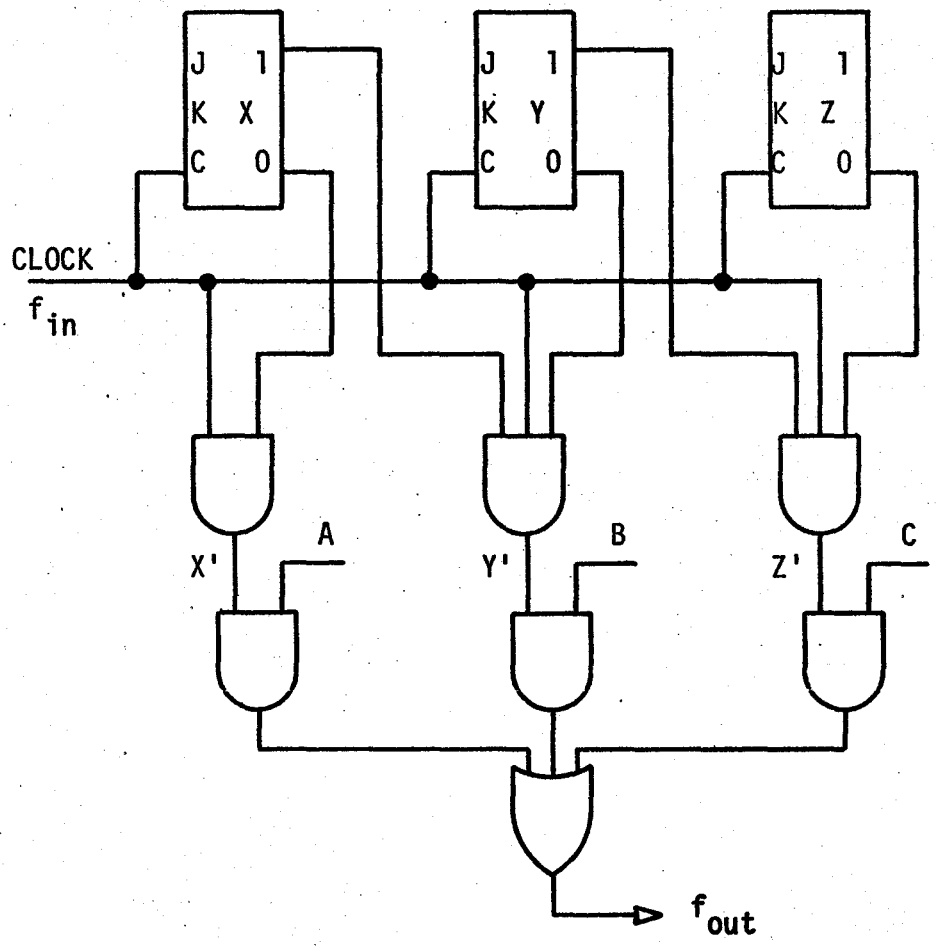


The RUN Signal, coming from the output of a 'RUN/STOP' switch, is also used to toggle a J-K flip-flop. Thus, activating the switch once will set the flip-flop and activating the switch a second time will reset the flip-flop. The output of the flip-flop is used to gate the incoming clock signal, provided by an external pulse generator, to the rate multiplier counter. Since all signals used when the synthesizer is running are derived from the rate multiplier counter, interrupting the incoming clock signal will effectively stop the synthesizer. Because the binary counter employed in this binary rate multiplier is not truly synchronous in that there is an accumulation of gate delays in each succeeding stage, the counter can be operated with a maximum clock input frequency of about 7.5MHz. However, for a basic interval of five milliseconds, the clock has to run at about a megahertz less than this so no problems were encountered. To improve the counter's speed, a fully synchronous circuit should be used.

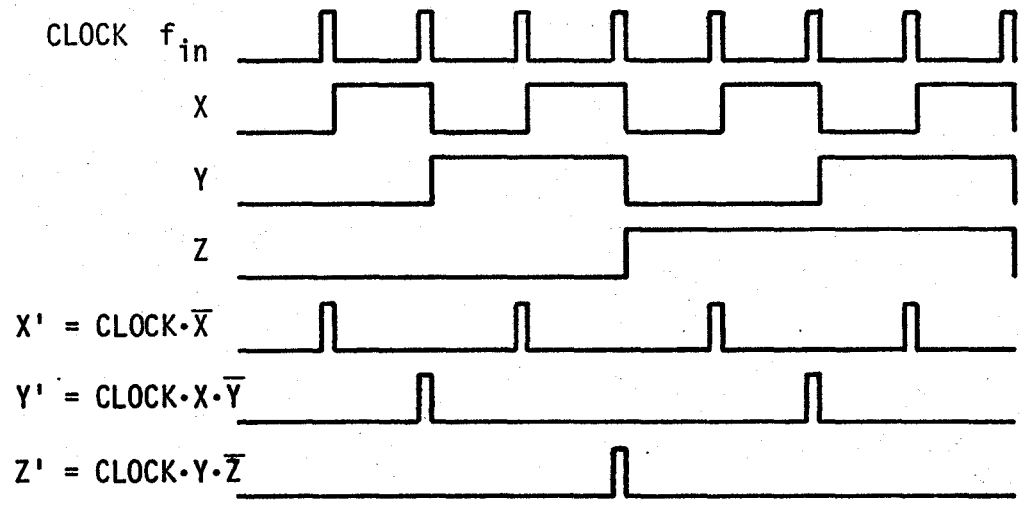
To understand the operation of the binary rate multiplier, refer to the simple three-bit rate multiplier of Figure 3-13. Most of the counter circuitry has been eliminated for simplicity. For a three-bit counter, there will be eight states. The timing diagram of Figure 3-13 therefore shows eight clock pulses driving the rate multiplier.

Signals X, Y and Z represent the outputs of the three flip-flops. Signals X', Y' and Z' are derived from these and the clock pulses. Note that line X' produces four pulses in the basic interval, line Y' produces two, and line Z' produces one. Since no two pulses occur at the same time,

Figure 3-13. Three Bit Binary Rate Multiplier




TIMING



lines X', Y' and Z' can be combined through gating to produce an output having any from zero to seven pulses. The lines are combined by gates controlled by A, B and C. Table 3-1 shows the relation between the number of pulses on the output line and the levels on lines A, B and C. It can be seen that the binary number given by A, B and C is the same as the number of pulses produced at the output of the rate multiplier in the interval required for the counter to toggle through all its states. The same principles can be applied to a six-bit (or β-bit) rate multiplier. In general, the input and output frequencies are related by equation (3-1) which is repeated here for convenience:

$$f_{OUT} = f_{IN} \times \frac{\alpha}{\beta} \tag{3-1}$$

where α is the binary number on the control lines and β is the number of states of the counter used with the binary rate multiplier.
For a six-bit binary rate multiplier:

$$f_{OUT} = f_{IN} \times \frac{\alpha}{64} \tag{3-3}$$

In this particular application, α is determined by bits eight through thirteen of the intermediate registers. Thus, in the interval $\phi/(64\times8)$ the rate multiplier is fed with sixty-four pulses and provides from zero to sixty-three pulses depending on the absolute value of the coefficient being sampled. More detail concerning binary rate multipliers and their uses is available in the literature (12, 13).

Table 3-1. Ratio of Input and Output Frequencies for a Three Bit Binary Rate Multiplier

| Inputs | | | | Lines Activated (4 pulses) | (2 pulses) | (1 pulse) | Output Pulses | Ratio $f_{IN}/f_{OUT}$ |
|---|---|---|---|---|---|---|---|---|
| a | A | B | C | X' | Y' | Z' | | |
| 0 | 0 | 0 | 0 | - | - | - | 0 | 0 |
| 1 | 0 | 0 | 1 | - | - | X | 1 | 1/8 |
| 2 | 0 | 1 | 0 | - | X | - | 2 | 2/8 |
| 3 | 0 | 1 | 1 | - | X | X | 3 | 3/8 |
| 4 | 1 | 0 | 0 | X | - | - | 4 | 4/8 |
| 5 | 1 | 0 | 1 | X | - | X | 5 | 5/8 |
| 6 | 1 | 1 | 0 | X | X | - | 6 | 6/8 |
| 7 | 1 | 1 | 1 | X | X | X | 7 | 7/8 |

$$f_{OUT} = f_{IN} \times a/8$$

## 3.4 Clocking System

No matter how many pulses are produced by the rate multiplier, its counter must toggle through all sixty-four states during every interval $\phi/(64\times8)$. From the timing diagram, it is seen that the three-bit counter, producing signals A, B and C, must change states once at the end of each of these subintervals. Thus, the three-bit counter can be driven directly by the six-bit counter of the rate multiplier. This is shown in Figure 3-14. The primary use of A, B and C is to supply the binary number to the digital switches so the correct intermediate register is connected to the synthesizer circuitry. Line C, which goes low once every $\phi/64$, is also used to clock the Walsh-Hadamard function generator. Finally, SYNCH 1 is formed from A, B and C to mark the beginning of each interval $\phi/64$.

$$\text{SYNCH 1} = \overline{\overline{A}.\overline{B}.\overline{C}} \qquad (3\text{-}4)$$

This signal is used in conjunction with the Walsh-Hadamard function generator to produce a LOAD pulse to transfer data from the input registers to the intermediate registers.

The SYNCH 1 pulse is also used to generate the LOAD CNTR, CLR CNTR and TRANSFER signals. The last signal is used to transfer the number in the binary up/down counter to the output buffer register. This must be done sixty-four times during the basic interval $\phi$. After transfer takes place, the up/down counter must be prepared to make a new count. For reasons to be discussed in the next section, two signals, LOAD CNTR and CLR CNTR, one being the inverse of the other, are needed.

Since the synthesizer is designed to run continuously, the transfer

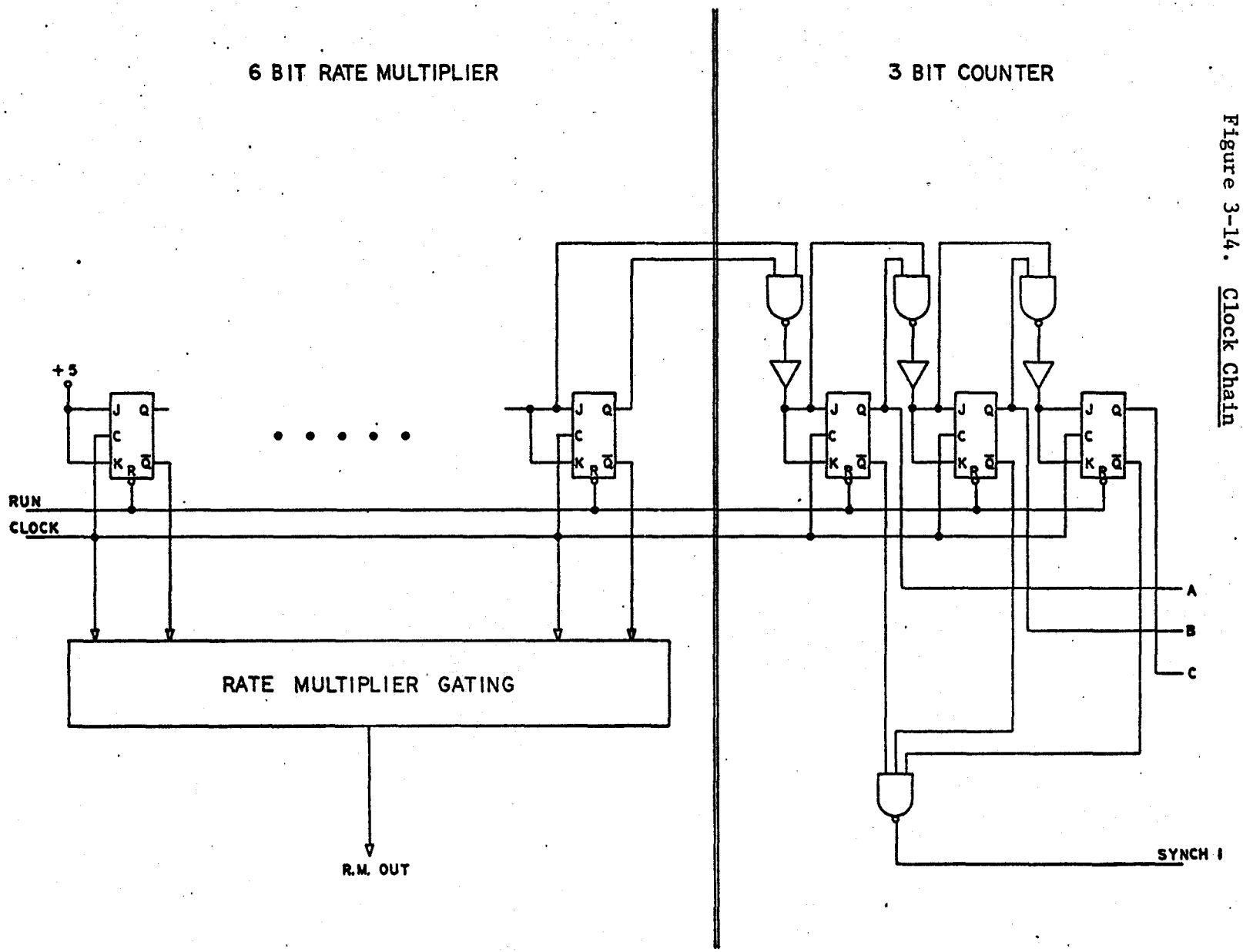
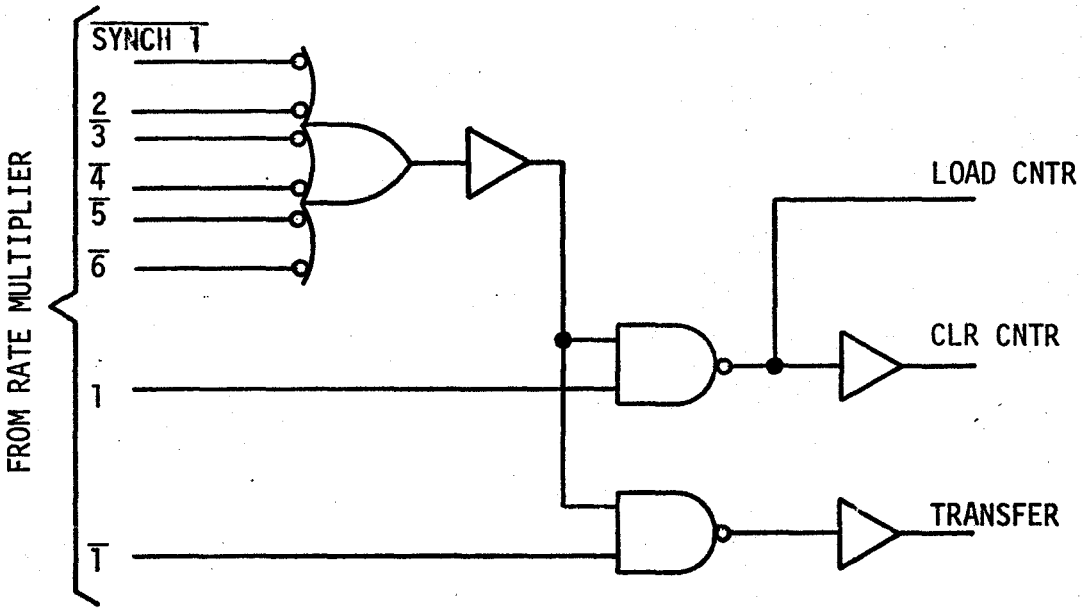



Figure 3-14. Clock Chain

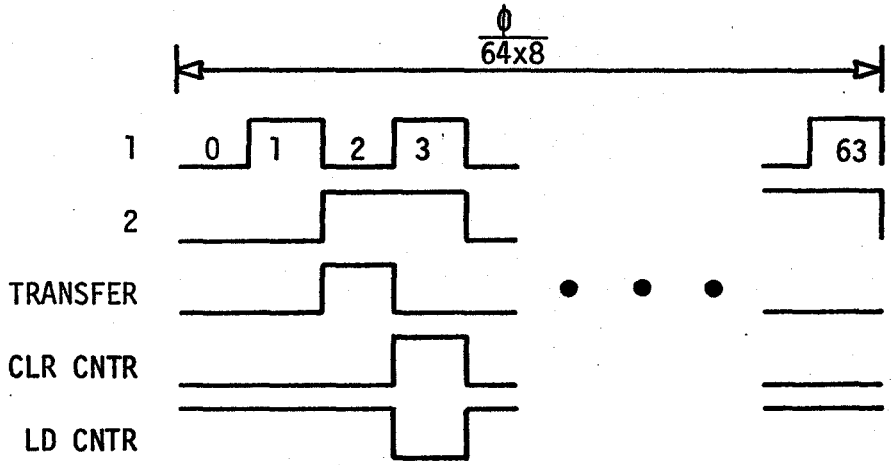
and resetting of the up/down counter must occur quickly at the end of every interval $\phi/64$. For this reason, the SYNCH 1 pulse is combined with a short pulse that occurs at the beginning of each interval $\phi/(64\times8)$. Such a short pulse can be formed by monitoring the state of the six-bit counter associated with the rate multiplier. When it is in its zero$^{th}$ state, and the SYNCH 1 pulse is on, the TRANSFER pulse should be formed. This pulse will last for only $\phi/2^{16}$. After that, when the rate multiplier counter is in its first state, the CLR CNTR and LOAD CNTR pulses can be generated. Unfortunately, due to timing problems caused by unequal gate delays, it was necessary to use the second and third states of the rate multiplier counter instead of the zero$^{th}$ and first to produce the TRANSFER and CLR and LOAD CNTR pulses respectively. These pulses, and their method of generation, are shown in Figure 3-15. The numbers on the lines feeding the gates represent the outputs of the six flip-flops in the binary counter of the rate multiplier. For instance, 2 represents the '1' output of the second least significant flip-flop while $\overline{6}$ represents the '0' output of the most significant flip-flop.

The TRANSFER and CLR and LOAD CNTR pulses occur during the interval when the rate multiplier is producing pulses according to the coefficient in register one. If pulses are produced by the rate multiplier before or during the generation of the synch pulses, they will not be counted by the up/down binary counter. For this reason, it is desirable to restrict the pulses being generated for the coefficient in register one to the interval after the CLR CNTR and LOAD CNTR pulses have been generated. By observing the operation of the rate multiplier, it becomes obvious that the only pulses

Figure 3-15. Generation of Synch Pulses




TIMING



that would arrive from the rate multiplier before the CLR CNTR and LOAD CNTR pulses are those generated by the first and second flip-flops in the rate multiplier counter. If the coefficient in register one is restricted to a value of 15 or less, no pulses will be generated in this time interval. This limitation is discussed in the next chapter and is seen to be unimportant.

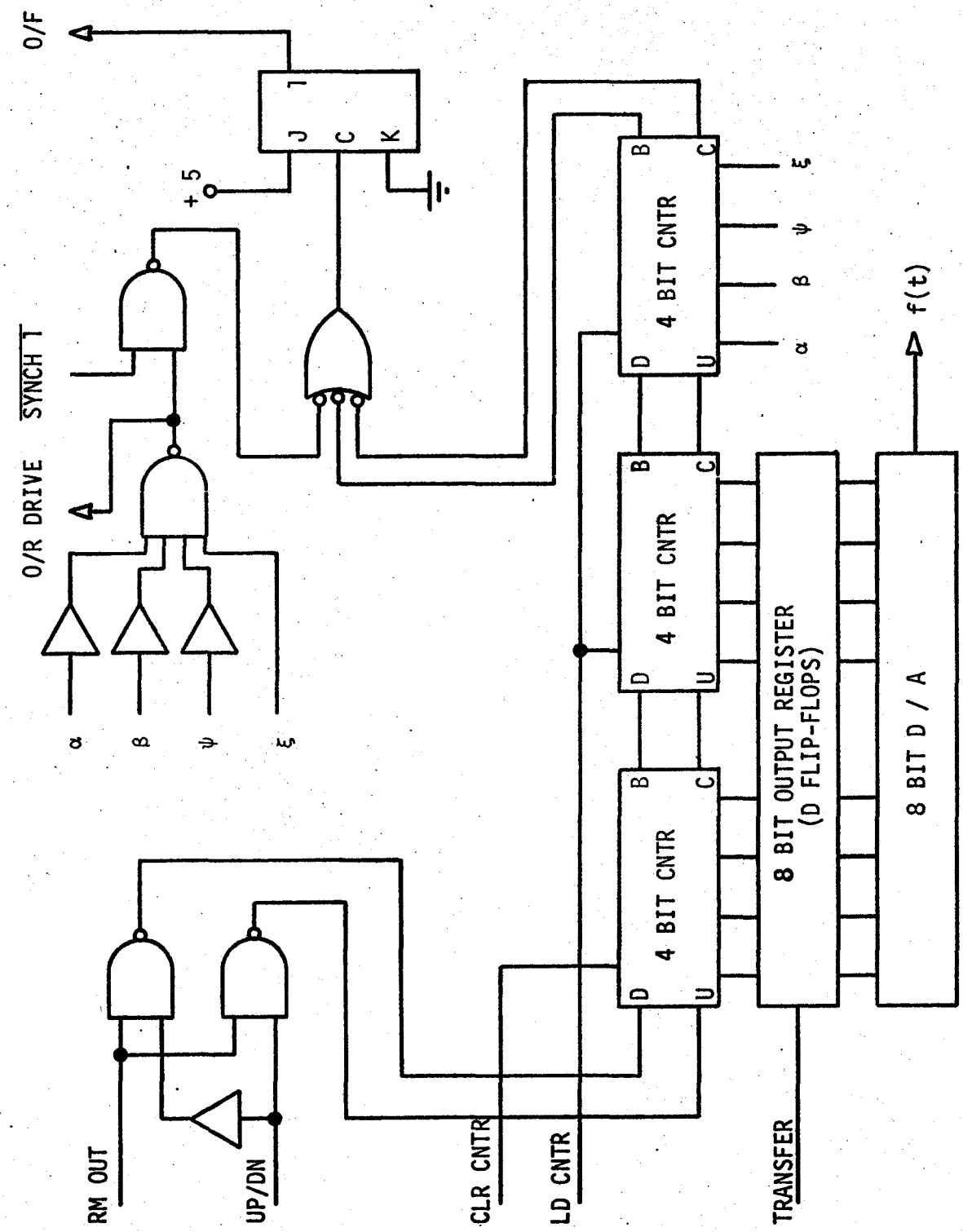
3.5 Output Circuitry

The output circuitry, shown in Figure 3-16, consists of three parts; the binary up/down counter, the output register and the digital-to-analog converter. Also associated with it is some overflow sensing logic.

The up/down counter consists of three cascaded four-bit up/down counters available in MSI integrated circuit form. The counters can be cleared or preset to any four-bit number independently of their clockings. They have two inputs, down count and up count. These two signals can be formed by gating the UP/DN signal from the Walsh function generator with the output of the rate multiplier as shown in Figure 3-16.

The number of bits required in the up/down counter is partly determined by the size of the digital-to-analog converter. Since eight coefficients, having absolute values from zero to sixty-three, are being summed, worst case design implies a maximum of $63 \times 8 = 504$ pulses being applied to the counter between resets. Since these can be counted either in the up or down direction, a total of 1008 different states are possible in the counter after the summing has been completed. A ten-bit counter and digital-to-analog converter would then be able to handle this range since $2^{10} = 1024$. Unfortunately, only an eight bit digital-to-analog converter was available.

Figure 3-16. Output Circuitry



Thus, two four-bit up/down counters are used to feed the converter. The counter therefore has a total of $2^8 = 256$ useful states (0 to 255). To guarantee that the output will lie in this range, the sum of the absolute values of the coefficients should be no greater than 127. The factor of one-half is necessary since the coefficients can either be positive or negative.

In order to facilitate the handling of positive and negative numbers, it is necessary to have the digital-to-analog converter centred in its range for a zero output signal. This is done by resetting the eight-bit up/down counter to the centre of its range at the beginning of every interval $\phi/64$. Therefore, the counter is set to the state 128 by the CLR CNTR and LOAD CNTR pulses. The eight output lines will then have the values $1000\ 0000_2$.

To facilitate the addition of a ten-bit digital-to-analog converter another four-bit up/down counter was included. It is referred to as a scratch-pad counter since it can conceivably be used with the eight-bit digital-to-analog converter. Since only the final sum must be in the range of the eight bits, it is conceivable that the partial sum, after the first five coefficients have been added, for instance, may exceed this range. The scratch-pad counter is used to accommodate these larger partial sums. This will not occur if the restriction of the sum of the absolute values of the coefficients being less than 127 is observed. The scratch-pad counter is also set to the centre of its range ($1000_2$) by the LOAD CNTR pulse to facilitate over-ranges in both the up and down directions.

The state of the scratch-pad counter is monitored by the overflow

sensing logic. If, at any time, its state is other than $1000_2$, indicating a partial sum outside the range of the digital-to-analog converter, the line O/R DRIVE, feeding a lamp driver, will go high. If the state of the scratch-pad counter is other than $1000_2$ at the time of transfer, indicating a final sum outside the range of the digital-to-analog converter, the O/R DRIVE line is gated by the SYNCH 1 signal to set a J-K flip-flop whose output is connected to another lamp driver. This informs the operator that the output signal is erroneous. The output of the flip-flop could also be used to flag the controlling computer to indicate an error if the synthesizer is in the computer mode.

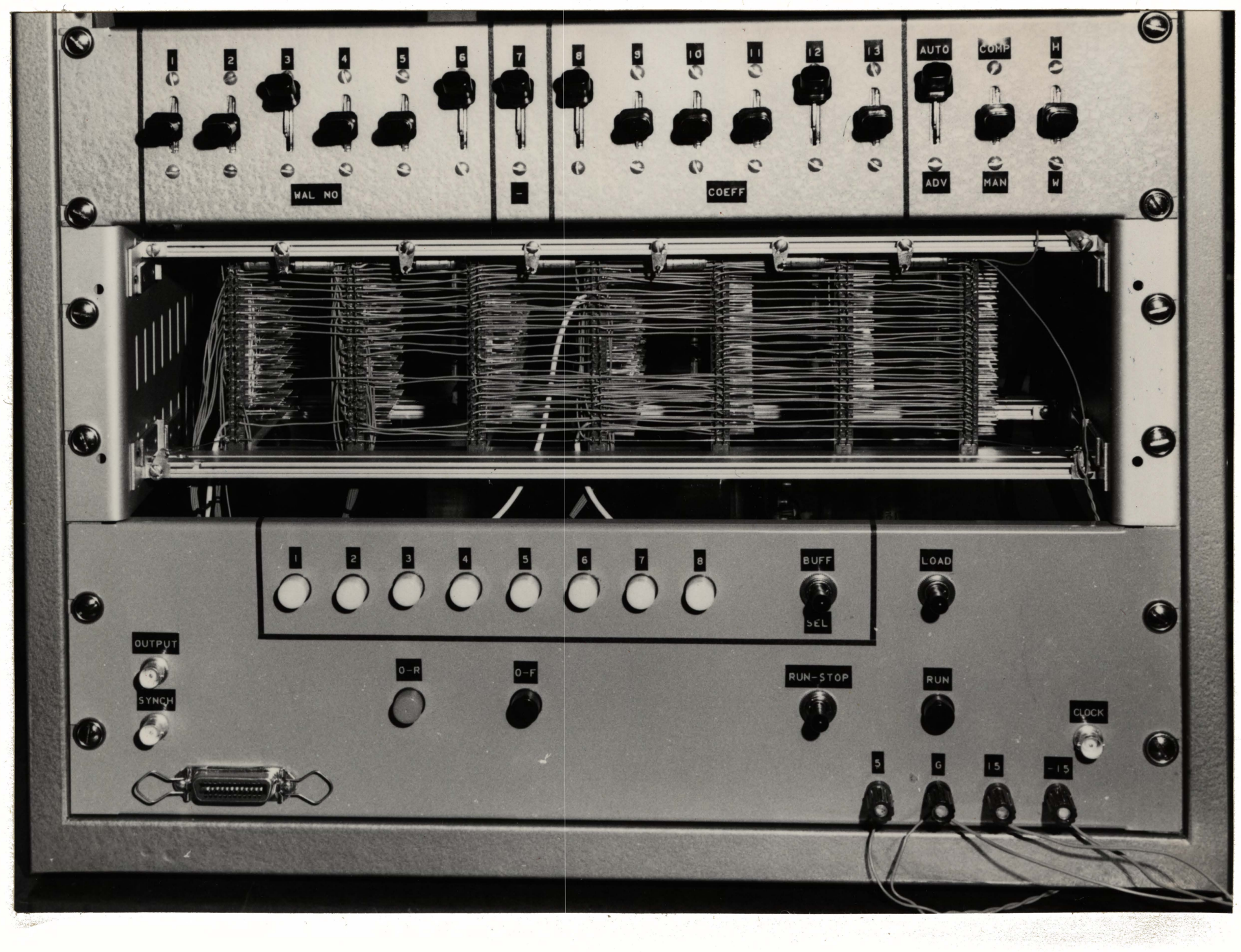
After the final sum has been calculated in the up/down counter, a TRANSFER pulse, derived from the clock chain, is used to transfer the data to an eight-bit output register consisting of eight D-type flip-flops. This register acts as a sample-and-hold, storing the final sum for the interval $\phi/64$ until the next final sum has been evaluated.

The outputs of the register are connected to a conventional eight-bit digital-to-analog converter consisting of a resistive ladder network and an operational amplifier. The output range is zero to plus five volts for an input from 0 to 255. Specifications for the digital-to-analog converter used are given in Appendix E. More general information on digital-to-analog conversion is given in the literature (14, 15).

## 3.6 Construction Details

Figure 3-17 shows a front view of the synthesizer. The upper panel contains the switches used to manually load the data. The 'Auto Advance'

Figure 3-17. Digital Speech Synthesizer - Front View



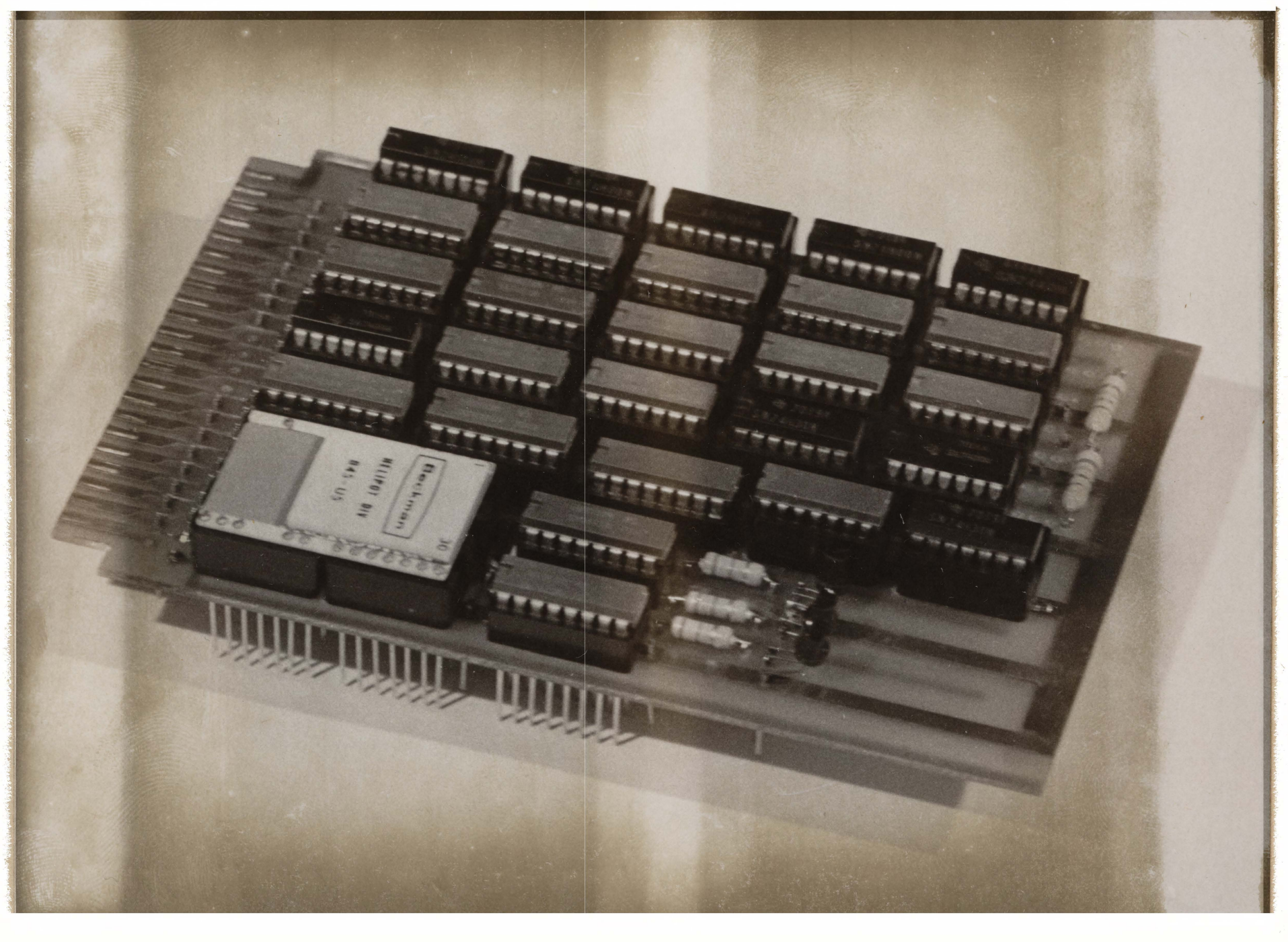
switch, the 'Mode' switch and the 'Function' switch are also contained here.

The bottom panel contains the lights indicating which register is to be loaded, the 'Buffer Select' switch, the 'Load' switch, and the 'Run-Stop' switch. Connectors for power, clock input, computer interfacing and the output of the synthesizer are also on this panel.

In the centre of the unit is the rack containing seven printed circuit boards on which all the digital circuitry is wired. For maximum ease in check-out, the connectors for the boards were left completely exposed. To facilitate changing of the wiring, wire-wrap sockets were used on the printed circuit boards to mount the integrated circuits. A typical board is shown in Figure 3-18. Its dimensions, like the other six boards, are 4" x 6". Boards with cladding on both sides were used so ground planes could be formed to isolate the circuitry. The seven board layouts are given in Appendix C. The signals on the pins of the board connectors are shown in Appendix D.

The actual circuitry used differs very little from that described in the previous sections. Major differences are due to the signal buffering required on several of the lines that must drive a large number of gates or flip-flops. For example, signals L1-L8 are buffered to form signals AL1-AL8 and BL1-BL8. All the wiring, except for the switches and connectors, was done on the seven printed circuit boards. The schematic diagrams associated with these boards are given in Appendix E.

Figure 3-18. Typical Printed Circuit Board



# CHAPTER 4

## RESULTS AND MACHINE LIMITATIONS

At the present time, the synthesizer has been constructed and manually tested for waveform synthesis. It has not yet been interfaced to a computer. An oscilloscope is used to monitor the output. Using the Walsh function coefficients listed in Appendix A, it was possible to reconstruct basic waveforms with the synthesizer. As Hadamard analysis had not been done on signals for this thesis, Walsh functions were used for all waveform synthesis.

Figure 4-1 shows the system in operation generating a ramp function. Synchronization of the oscilloscope is accomplished by using the LOAD pulses which occur once every basic interval $\phi$. Figure 4-2 shows the same system used to generate triangle waves.

In order to produce hard-copy representations of the output, a Sanborn 322 DC Amplifier-Recorder was used. Unfortunately, this device has a limited bandwidth resulting in overshoot and rounding of corners of square waves which are not true representations of the output signal. However, by slowing the recorder down to a time axis speed of one millimetre per second and operating the synthesizer at a factor of approximately $10^4$ slower than normal, reasonable traces were obtained. It is possible to alter the basic interval in this way by merely changing the input clock frequency since all other timing signals are derived from it. To double the length of the basic interval, it is only necessary to run the clock

Figure 4-1. System Used to Generate a Ramp Function

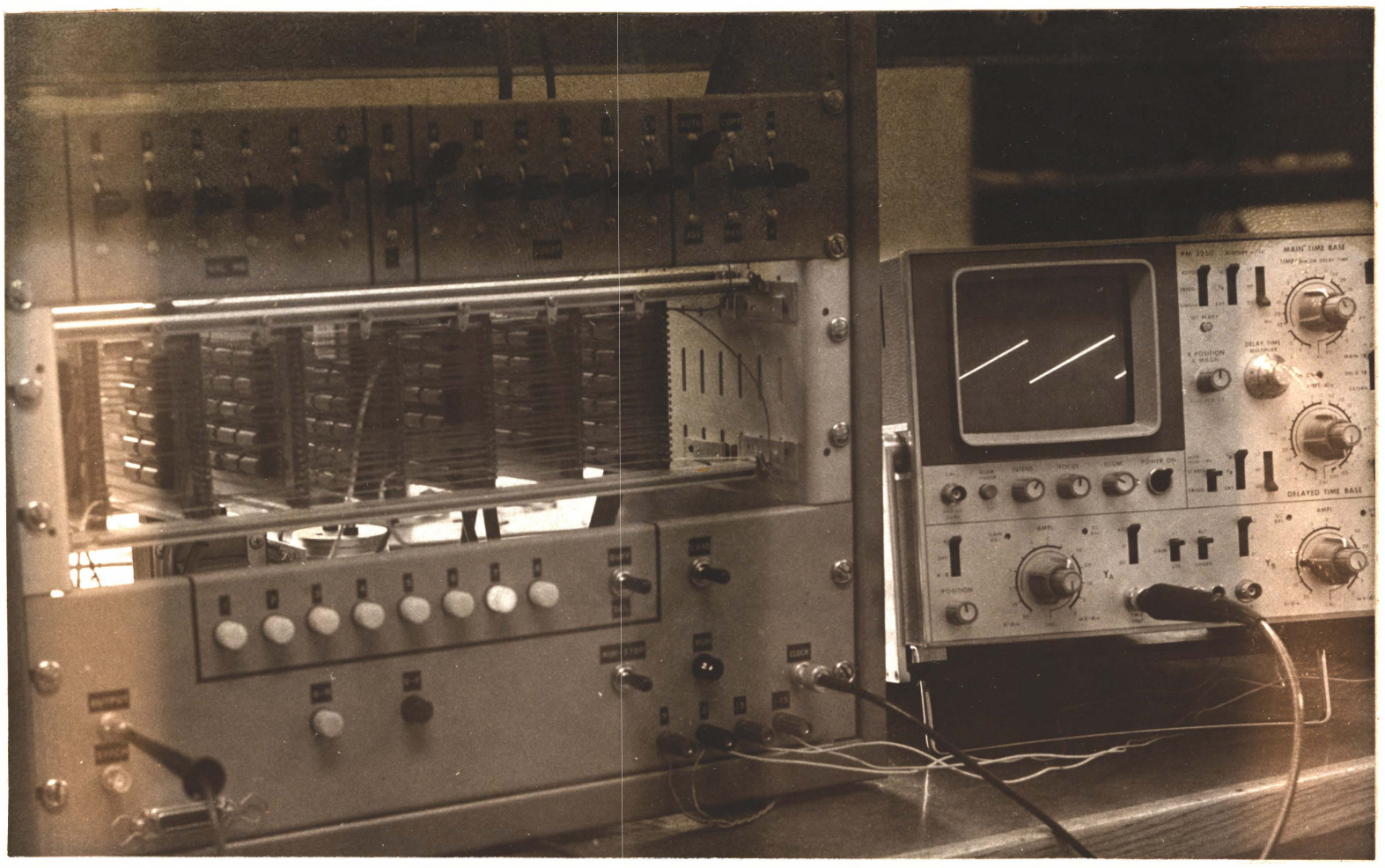
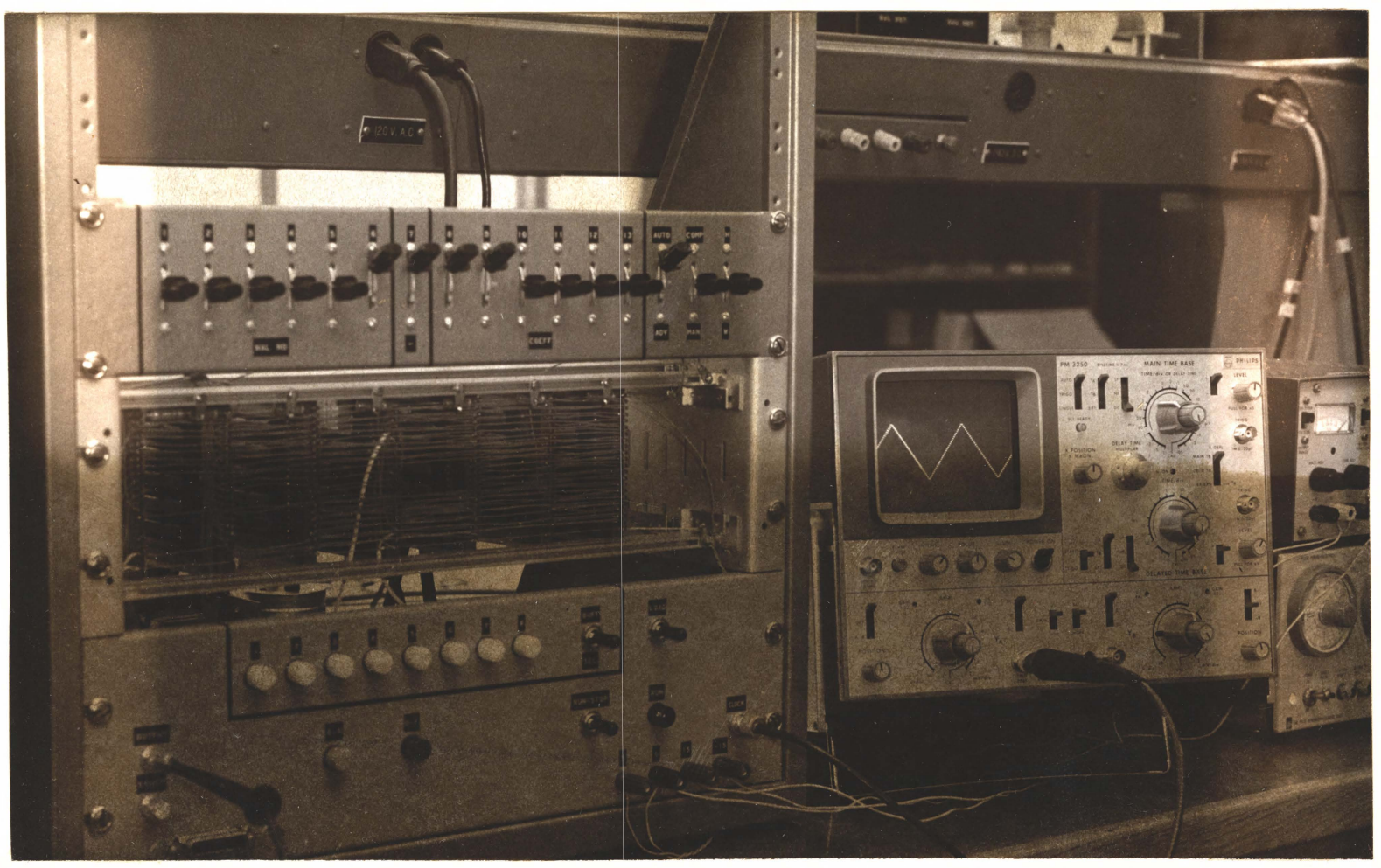



Figure 4-2. System Used to Generate a Triangle Function



at half its normal frequency. The results obtained with the Sanborn recorder on the unit running at low speed are discussed in section two of this chapter.

## 4.1 Sources of Error

There are essentially two classes of error in a machine of this type. The first can be referred to as mathematical error. This is due to the fact that a finite series representation of a signal f(t) can only be approximate. For truly accurate reproduction, an infinite number of terms would have to be summed. Accuracy is further limited in a mathematical sense by the quantization of the values of the coefficients used. In the case of the synthesizer constructed for this thesis, the amplitudes are quantized into 127 segments. Similarly, in the determination of the coefficients, the original signal was also quantized. Finally, for true representation of the output signal, it would be necessary to instantaneously update the series. In other words, $\phi$ should be reduced to an infinitesimal value.

The other class of errors can be referred to as machine errors. These are a result of the limitation of the components in carrying out the mathematical operations required to evaluate the output signal. Such errors can be reduced significantly by careful design of the synthesizer. In fact, in this machine, there are essentially only two sources of machine errors. The first is the digital-to-analog converter whose output can only be approximately related to the binary number on its input leads. As its error is quoted as being plus or minus one half of the

least significant bit (approximately 0.2%) this error is insignificant. Generally, speech is quite intelligible to the average listener if amplitude errors are less than 5% rms (16). The other source of machine error is due to the finite time required to transfer data from the up/down counter to the output register and to reset the up/down counter. As mentioned earlier, there will be no error in transferring the data (i.e. loss of data) if the coefficient in register one is restricted to an absolute value of fifteen or less. For the worst case, when the coefficient in register one is sixty-three, a total of four pulses will be lost. If this is the only coefficient being used in the sum, the total error due to these lost pulses would be 4/63 x 100% = 6.35% which would be significant. However, by ordering the coefficients so that the smallest is placed in register one, the only time sixty-three would be placed there would be if all the coefficients had absolute value sixty-three. Thus, four pulses would be lost out of a total of 63 x 8 = 504 pulses leading to a data transfer error which will appear at the output as an amplitude error of less than 0.8%.

Actually, the above case would never arise because only an eight-bit digital-to-analog converter is used. Thus, the total number of pulses produced during the basic interval is limited to the range of 0 to 127. Therefore, if coefficients are ranked so the smallest appears in register one, its maximum size can only be fifteen. In this case, no pulses will be missed from register one.

The limit on the value of the coefficients imposed by the sum of

their absolute values being less than 128 can be met by suitable scaling of the coefficients. In practice, however, it is highly unlikely that extreme scaling will be necessary. In fact, for the waveforms shown in the next section, no scaling was found to be necessary. In most cases, the coefficients were actually scaled up to give a larger amplitude to the output signal.

Along with the quantization of the coefficients, there is a quantization of the output signal. Since an eight-bit digital-to-analog converter is used, the output is quantized into 255 parts. This results in an error of approximately 0.4%. In summary, then, the quantization of the coefficients leads to a quantization of the output giving an error of 0.4% of the maximum peak-to-peak value of the output signal. It was indicated that scaling of the coefficients would introduce further error. However, if scaling is done only to ensure that the output signal does not exceed the range of the digital-to-analog converter, this error will be contained in the basic 0.4% quantization error of the output signal since the sum of the absolute values of the coefficients gives the peak value of the output signal.

Referring to equation (2-29), it is obvious that a main source of error in the representation of f(t) is due to the finite length of the series:

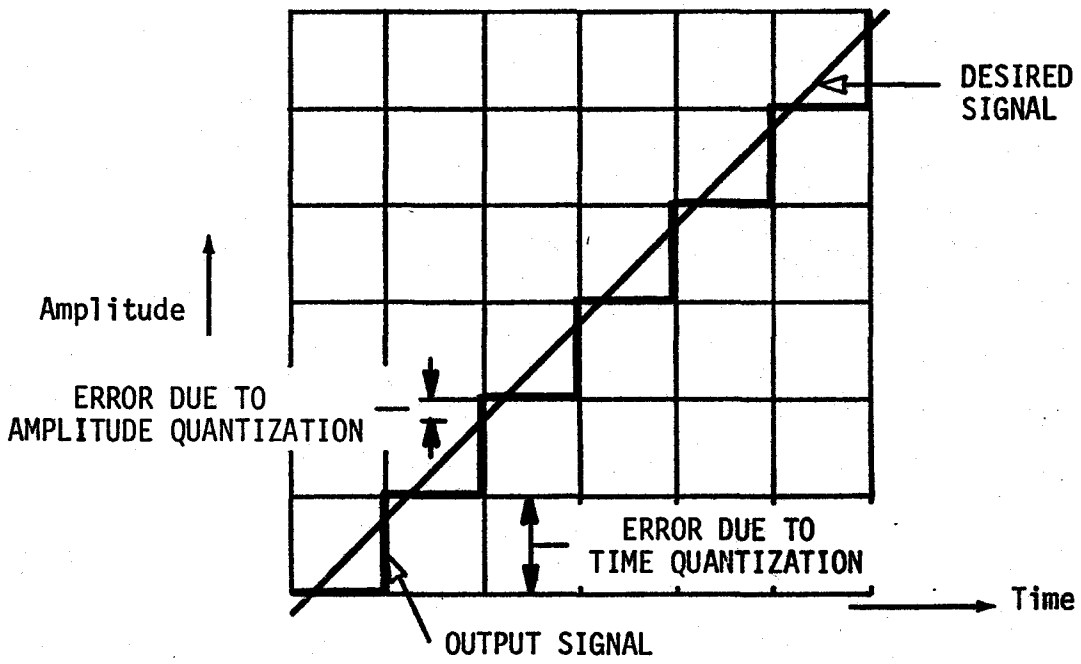
$$f(t) \simeq \sum_{n=1}^{M} C_{k_n} \, WAL(k_n, \theta) \qquad (2\text{-}29)$$

In this case, M has been chosen to be eight. The amount of error associated with ignoring the terms of order nine and higher will differ according to the nature of f(t). For rectangular pulses, of duty cycle 0.5, 0.25 and 0.125, there is no error caused by this truncation. The amount of error can be calculated by summing the coefficients of order higher than nine and comparing them to the sum of the eight most dominant coefficients. Analogous to the square wave requiring many terms to be reproduced accurately by a Fourier series, a sine wave requires many Walsh functions to be reproduced accurately with a Walsh series. Analysis of the sine wave, given in the next section, will therefore give a good indication of the maximum error to be expected due to the truncation of the series given in equation (2-29).

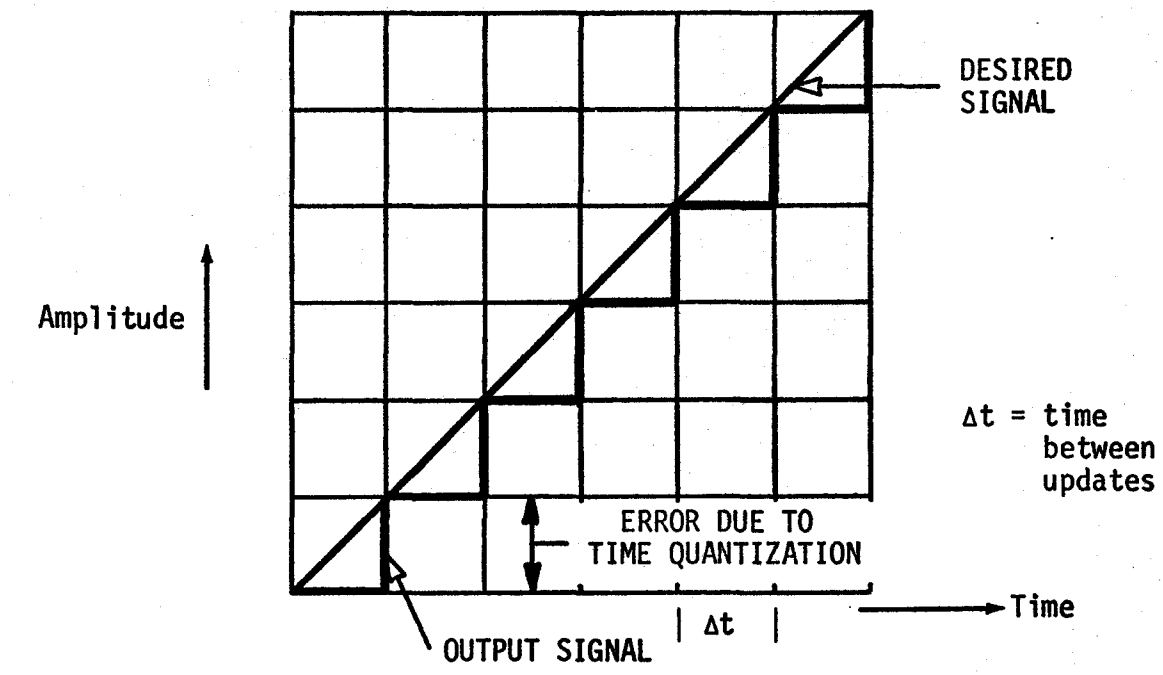
Recall that the eight terms used to make up the sum representing the output signal can be selected from only the first sixty-four Walsh functions. This means that the highest sequency component will be sixty-four. The output signal is, therefore, changed at a maximum rate of $\phi/64$ or approximately 78microseconds. The amount of error induced by this time quantization will depend on the rate of change of f(t). Assuming f(t) is limited to 3kHz, the sampling theorem predicts that there will be adequate representation of the signal. The updating of the output every 78microseconds will lead to an erroneous frequency content of about 12.8kHz superimposed on f(t). This can be eliminated by passing f(t) through a low pass filter.

Figure 4-3, using a ramp function as the desired output illustrates

Figure 4-3. Sources of Error Due to Quantization of the Output Signal




A. Time and Amplitude Quantization




B. Time Quantization Only Assuming Perfect Coefficient Transfer

how amplitude quantization of the output signal, and time quantization affect the amplitude of the signal output from the speech synthesizer. Figure 4-3(a) assumes that coefficients are rounded off to the nearest integer values.

When the speech synthesizer is interfaced to a computer, new sets of data will be loaded every five milliseconds. If the original signal being sampled is not periodic, or if its period is not a multiple of $\phi$ each set of coefficients will be different. This means that the output signal will change shape once every 5milliseconds as it attempts to follow the original signal. The result is that there will be an error signal having a frequency of 1/5000 = 200Hz, superimposed on f(t). The magnitude of this error signal will depend on how abruptly f(t) changes with each update at the end of the basic interval $\phi$. Since most information contained in speech is in the band from 300Hz to 3kHz, this error signal can be reduced by incorporating a high pass filter on the output to remove signals of frequencies below 300Hz. Alternatively, a band pass filter cutting off at 300Hz and 3kHz would eliminate this error signal and the one occurring at the rate of $\phi/64$. Note that if a variable update rate were used, as mentioned in section 2.6, the maximum rate would be about 250Hz or an interval of 4milliseconds. This would allow filtering to separate the error signal from the desired speech signal.

It was stated earlier that ideally the basic interval $\phi$ would be very short. This would allow the output signal to follow more accurately abrupt changes in the original signal. However, a compromise between the

desirability of a short interval $\phi$ and the limitation of a maximum update rate of 250Hz leads to an interval of approximately 4milliseconds. Thus, there can be a maximum of a 4millisecond delay in the change of the output signal. Although no experimental work has been done on this, it is expected that this delay will have an insignificant effect on the sound of the speech generated by the synthesizer. Due to the fact that the maximum allowable clock frequency used to drive the counter in the rate multiplier circuit was around 7MHz, a basic interval of 5milliseconds was chosen.

## 4.2 Generation of Basic Waveforms

With an understanding of the various sources of error, experimental results of some waveforms obtained from the unit are presented. These are taken directly from the digital-to-analog converter so no filtering has been done on the signal, except that inherent in the recorder. As a result, there is a strong 12.8kHz signal superimposed on the outputs. Since the periods of the waveforms were chosen to coincide with $\phi$, there will be no 200Hz error signal.

Figure 4-4 shows the generation of a sine wave. As mentioned earlier, this waveform is one of the most difficult to generate with a Walsh series. Thus, the greatest source of error is due to the truncation of the number of terms used in the series. This is shown by how quickly the output signal improves as more terms are added to the series. Figure 4-4 shows the sine wave generated with the seven most dominant terms. The total error in this signal was measured to be less than 2% of peak value. This is well within the acceptable range for speech.

Figure 4-4. Generation of the Sine Wave Using the Seven Most Dominant Terms

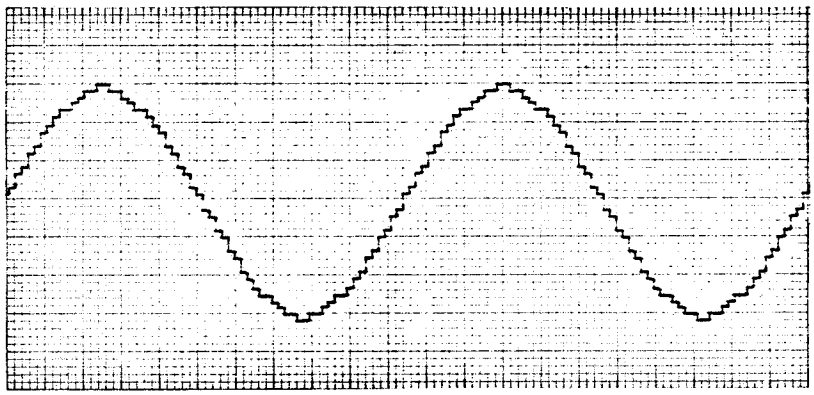
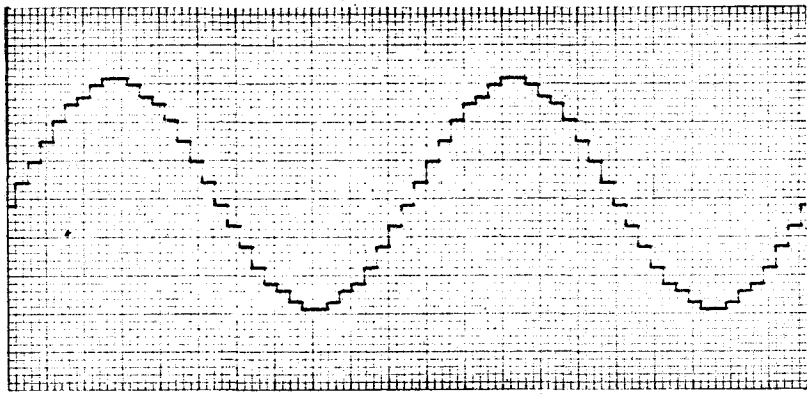
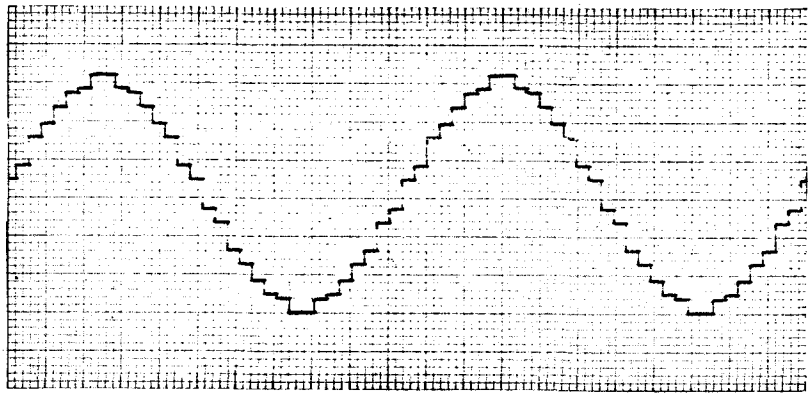


Figure 4-5. Development of the Sine Wave

A. Six most dominant terms



B. Five most dominant terms



C. Four most dominant terms

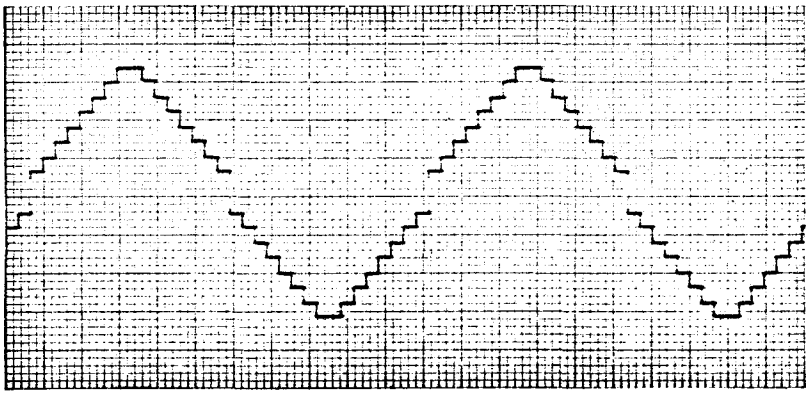
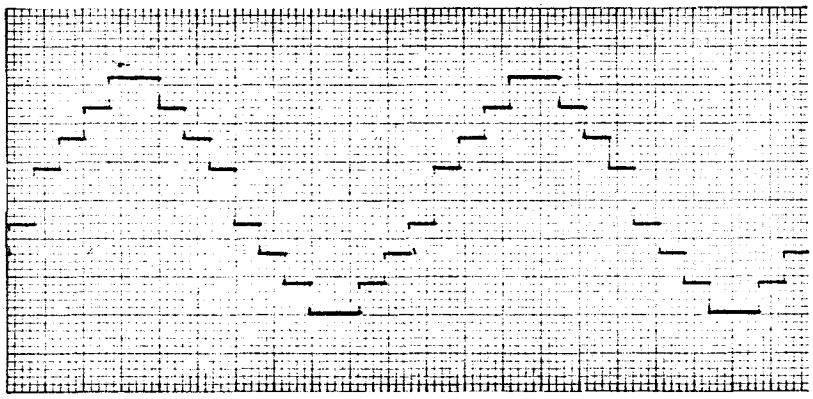
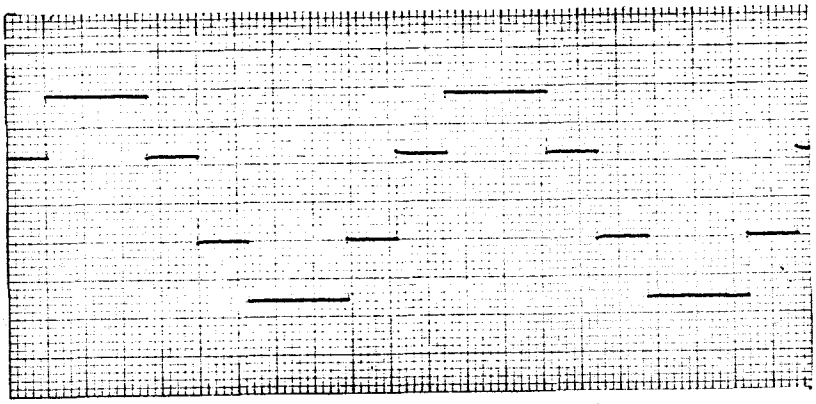


Figure 4-5. (cont'd)

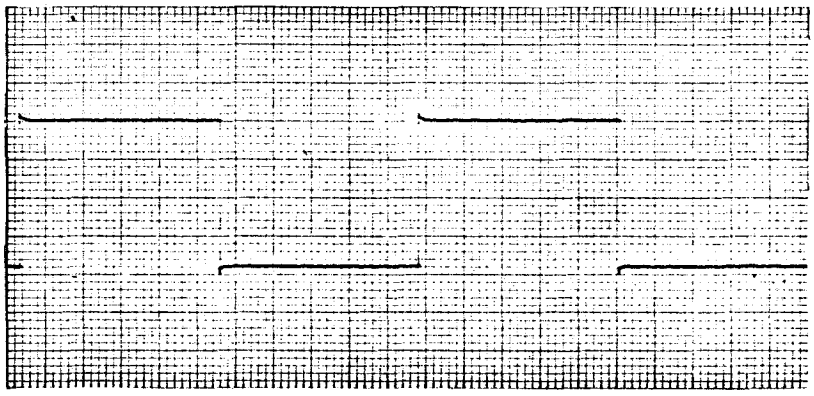
D. Three most dominant terms



E. Two most dominant terms



F. Most dominant term



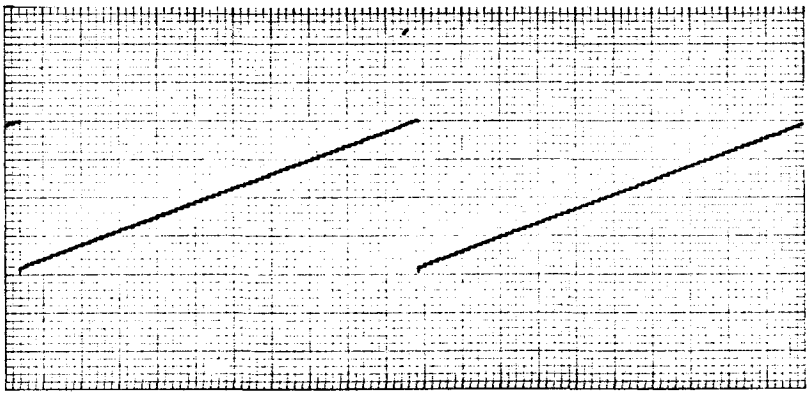
Referring to Appendix A for the sine wave coefficient analysis, it is seen that the remainder of the first sixty-four coefficients total in absolute value 3.6% of the sum of the absolute value of all the coefficients. This would represent a peak-to-peak value error giving an error of 1.8% of peak value. Thus, it is seen that most of the 2% error measured for this waveform was due to truncation of the Walsh series which would be expected for the sine wave.

Figure 4-5 shows the sine wave generated with fewer of the dominant terms being used. As can be seen, the quality of the sine wave quickly deteriorates. In Part F, only the most dominant term is used (WAL(1,$\theta$)). This term defines the sine wave as being an odd function.

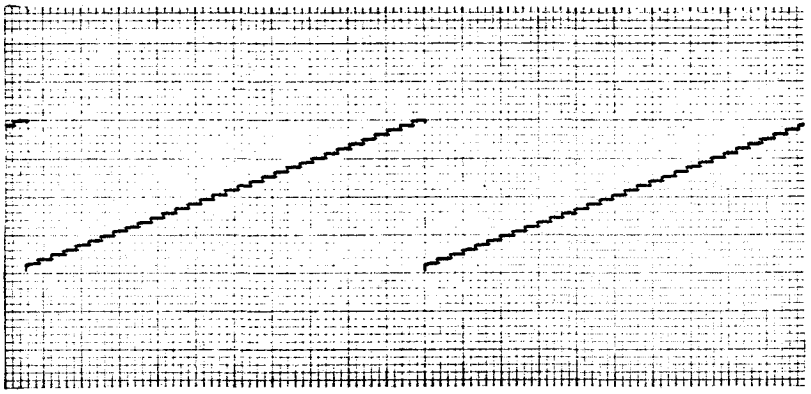
The next function to be analyzed is the ramp, shown in Figure 4-6. Again, it is obvious that the waveform is better represented when more coefficients are used. Note in this case, however, that there are only six non-zero coefficients among the first sixty-four Walsh functions. Thus, the error in the waveforms is not primarily due to truncation of the Walsh series but due to the restriction on the range of the Walsh functions available. In other words, the error is due to time quantization rather than ampltidue quantization of the output waveform. When all six terms are used, the error is approximately $1/(63\times2) \times 100\% = 0.78\%$ of peak-to-peak value or 1.56% of peak value. This value was arrived at by considering the fact that the difference between quantization levels is 1/64. Thus, the output signal will differ by a value of plus or minus half of 1/64.

Figure 4-6. Generation of the Ramp Function

A. Six most dominant terms



B. Five most dominant terms



C. Four most dominant terms

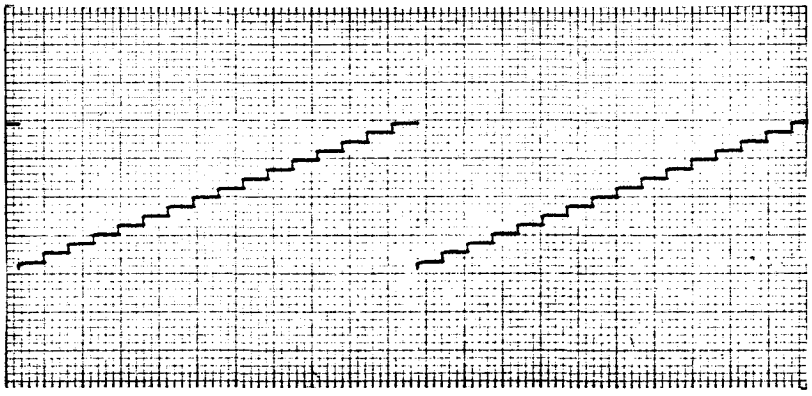
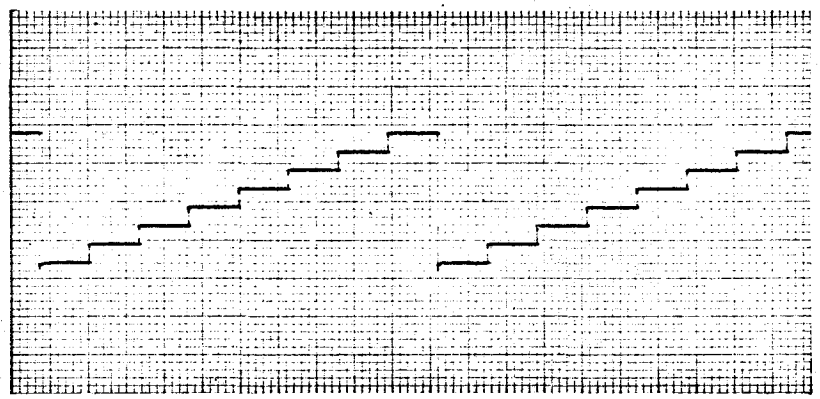
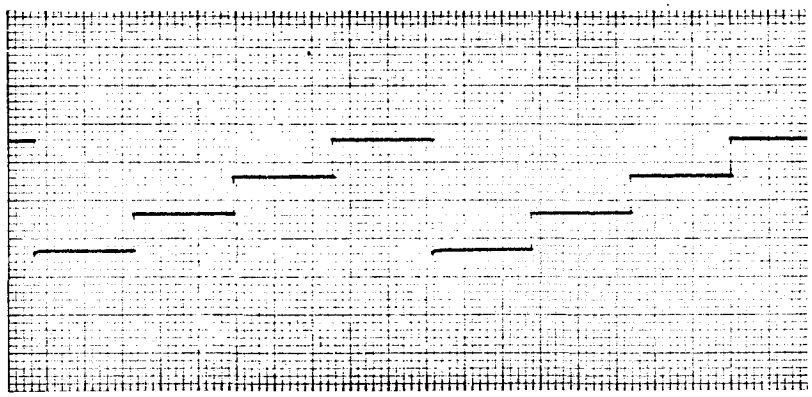


Figure 4-6. (cont'd)

D. Three most dominant terms



E. Two most dominant terms



F. Most dominant term

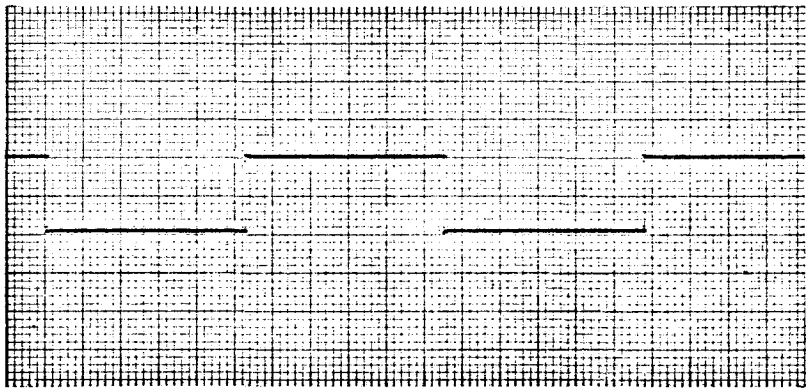


Figure B-5. (cont'd)

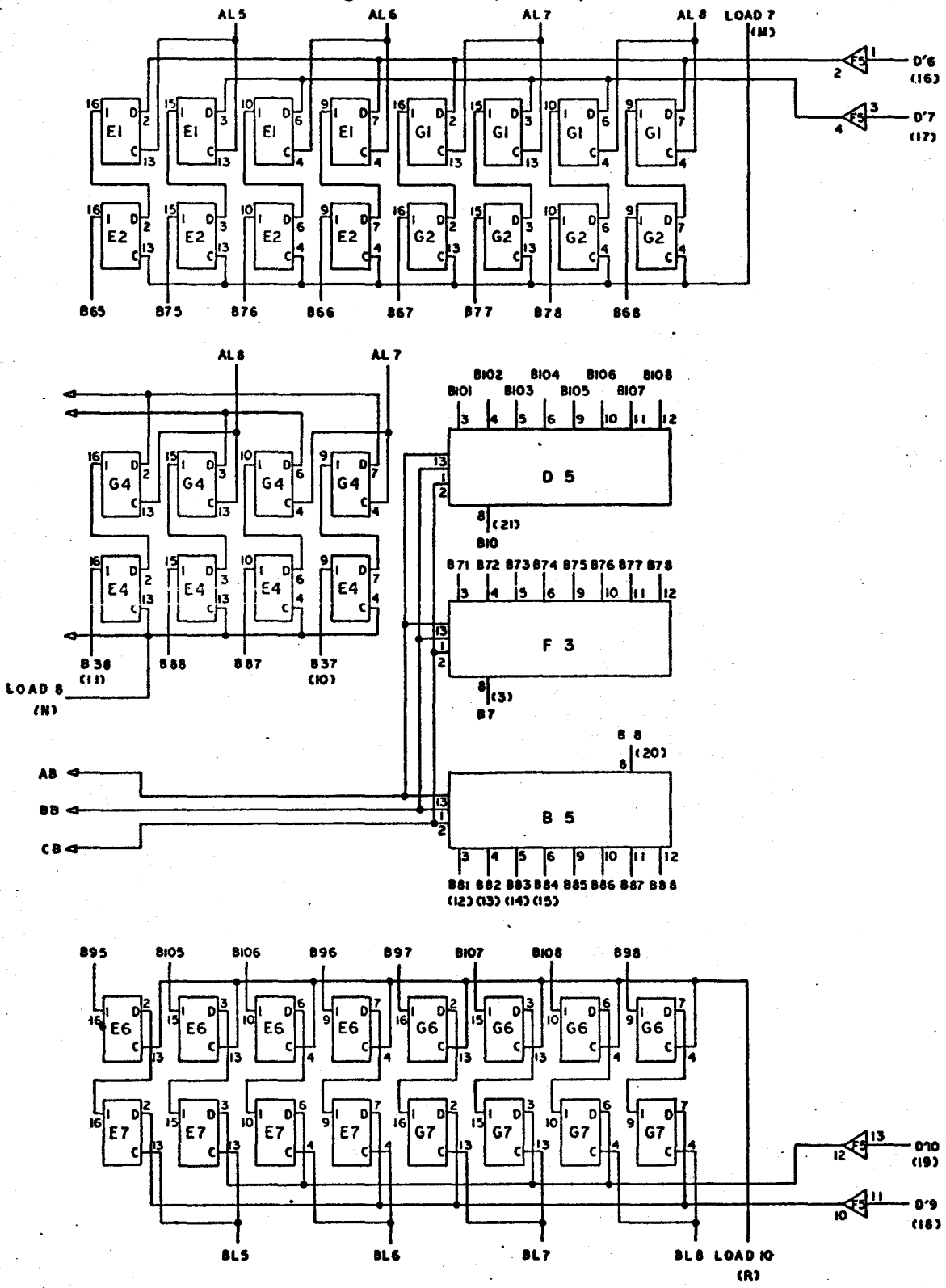


Figure B-6. Circuit Diagram - Board F

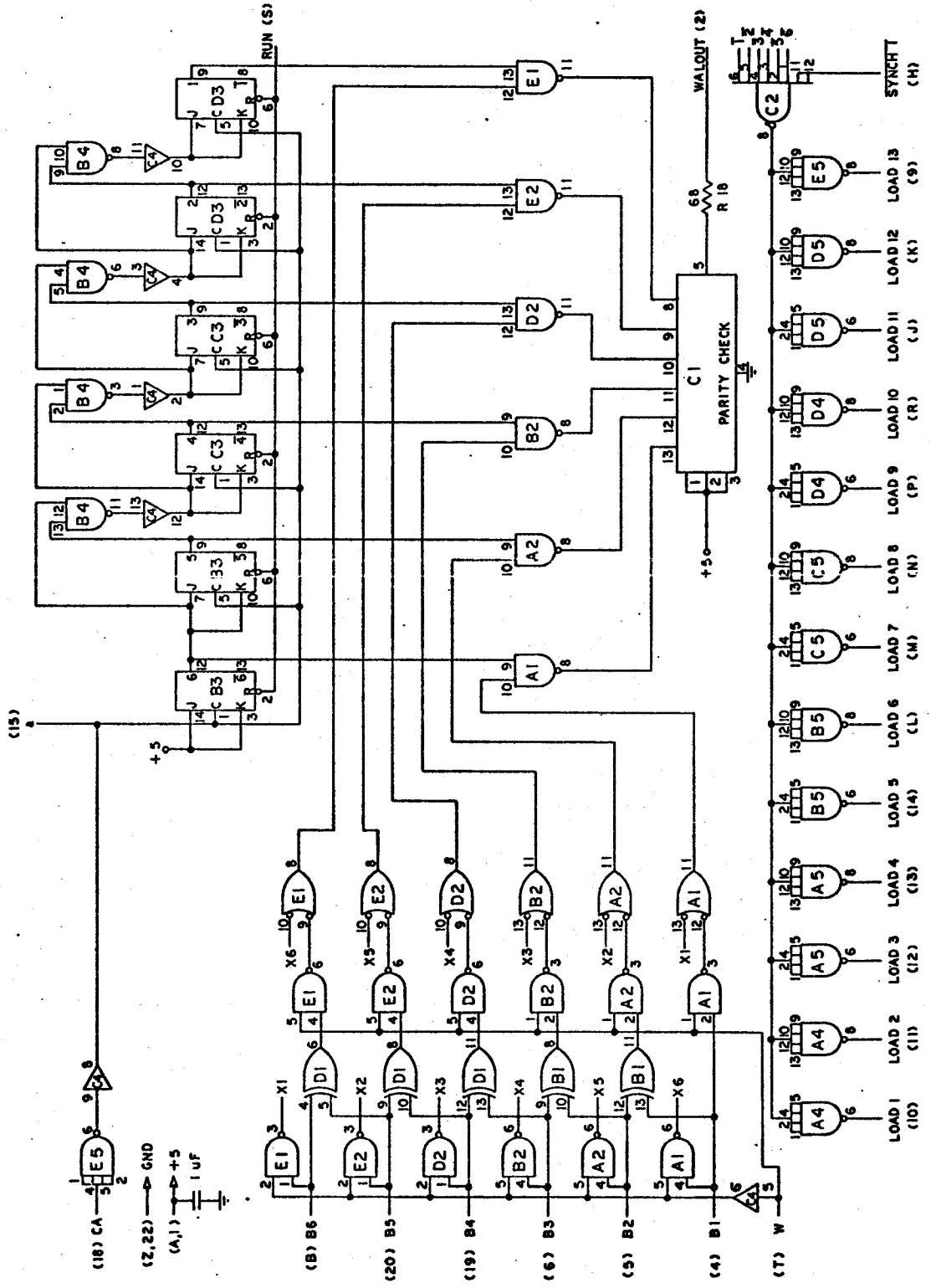


Figure B-7. Circuit Diagram - Board G

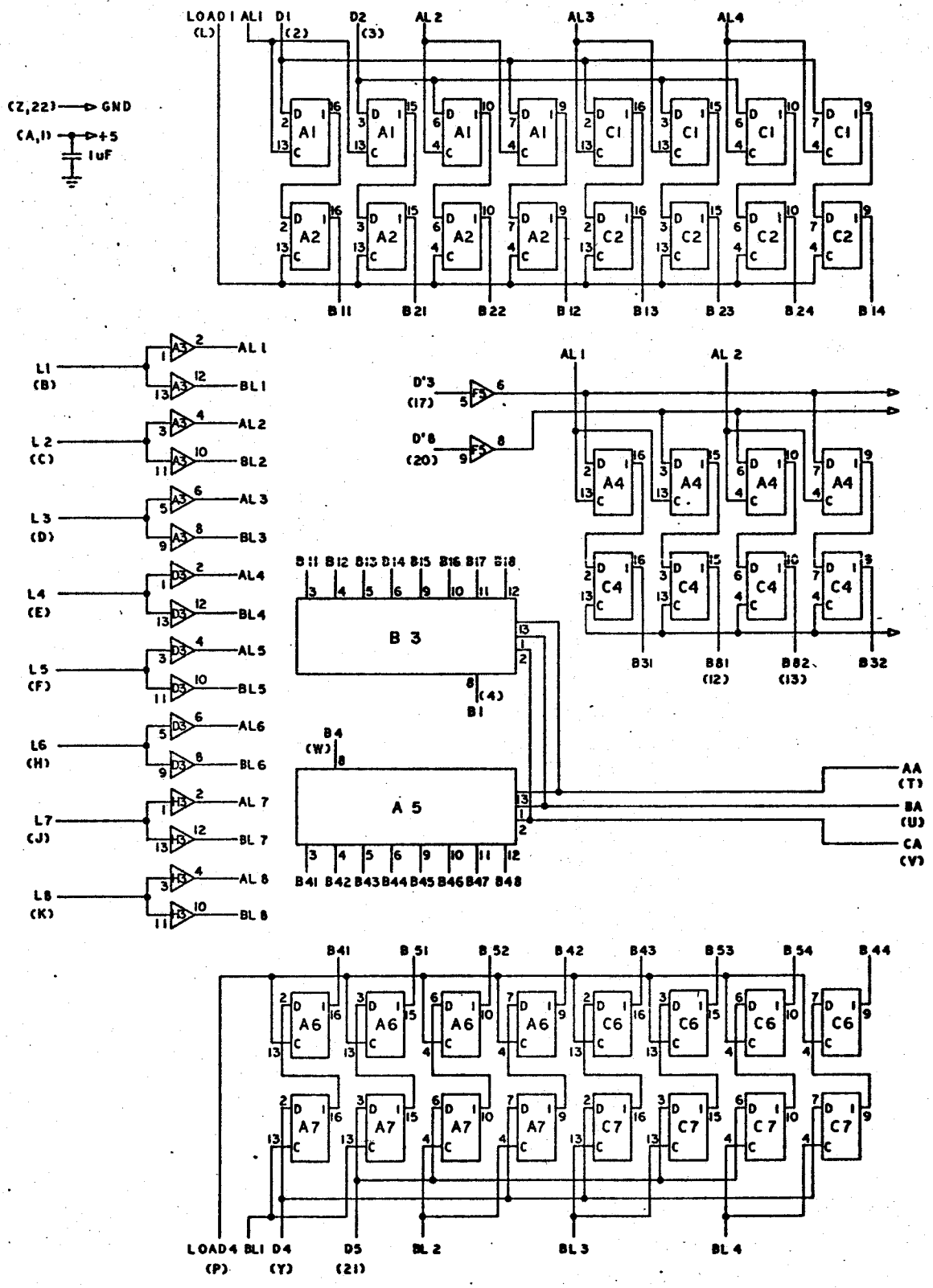
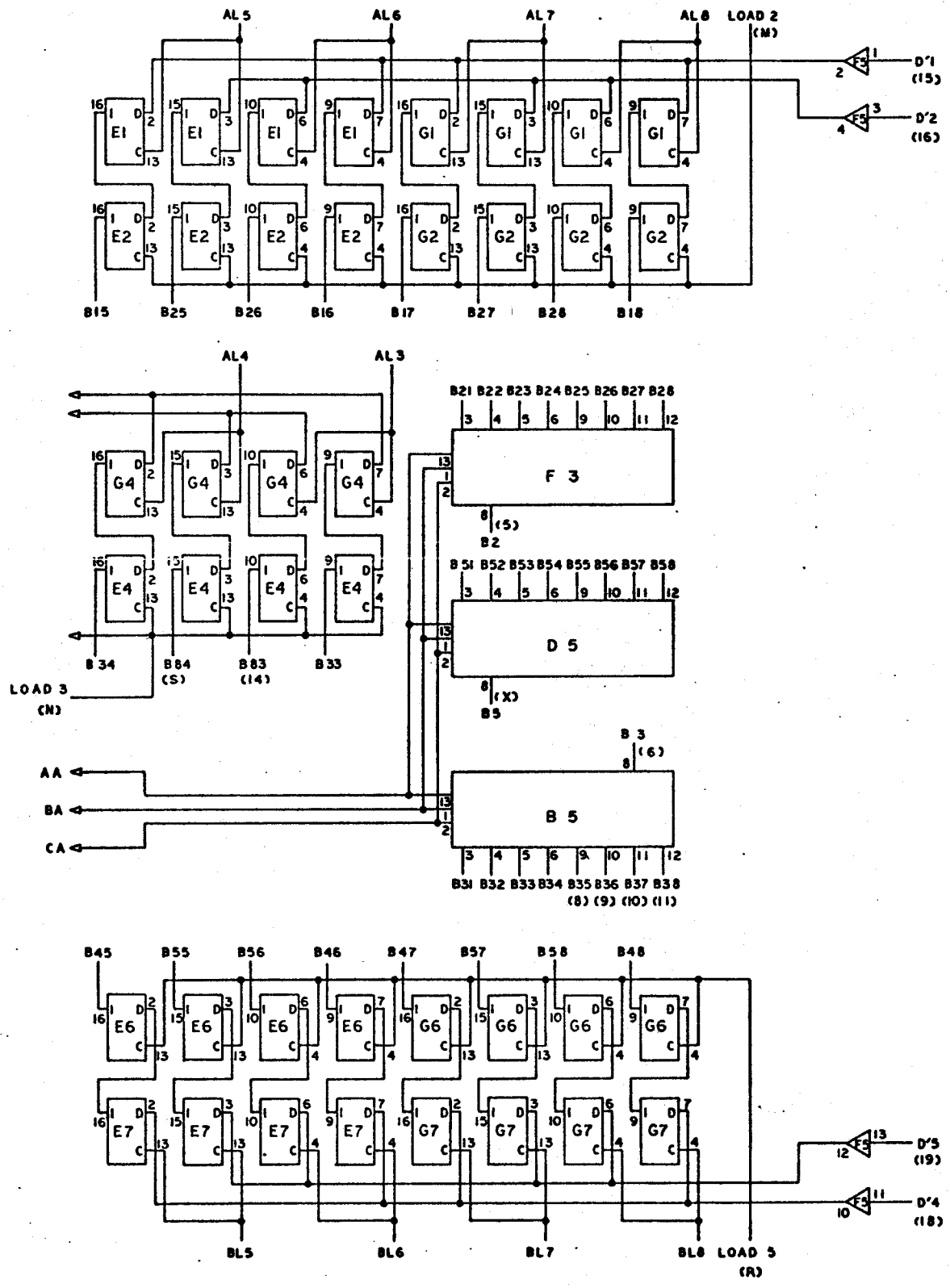


Figure B-7. (cont'd)



# APPENDIX C

## CIRCUIT BOARD LAYOUTS

The following pages show the physical layout of the seven circuit boards. The diagrams are of the top side (integrated circuit side) of the boards as opposed to the pin side. Numbers refer to Texas Instrument part numbers for the integrated circuits with three exceptions. The digital switches on boards C, E and G are National Semiconductor part numbers DM8210 and the shift-register on board A is a National Semiconductor part number DM8570N. The digital-to-analog converter is a Beckman unit, part number 845-U5. All integrated circuits are mounted with pin one towards the front (connector edge) with the exception of the digital-to-analog converter. Figures shown near the connector edge of the board refer to pin numbers for external connection. The numbers and letters shown surrounding the boards form a grid system for identifying the integrated circuits. These correspond to the identifying figures on the circuit diagrams of Appendix B.

Figure C-1. Circuit Board Layout - Board A

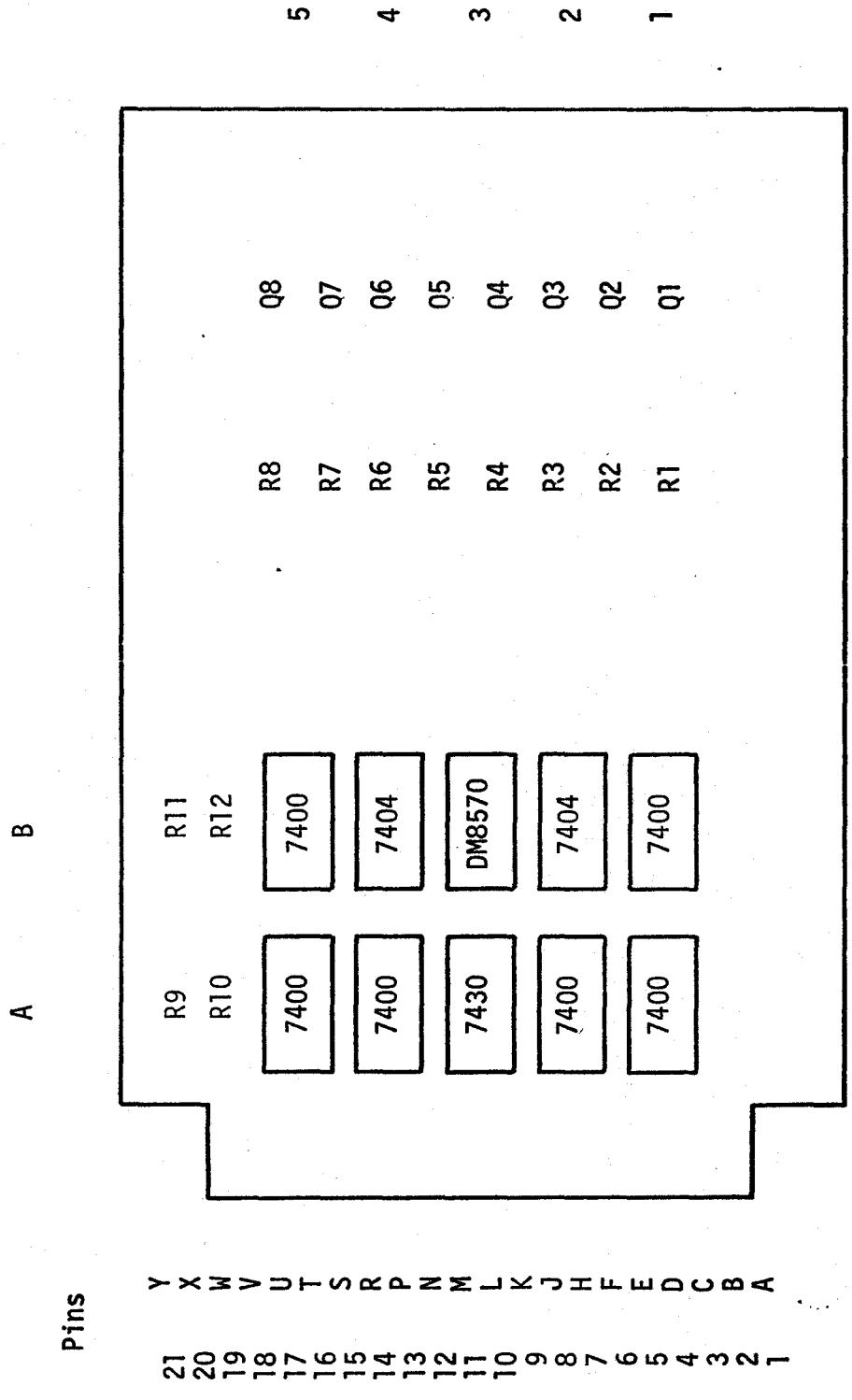


Figure C-2. Circuit Board Layout - Board B

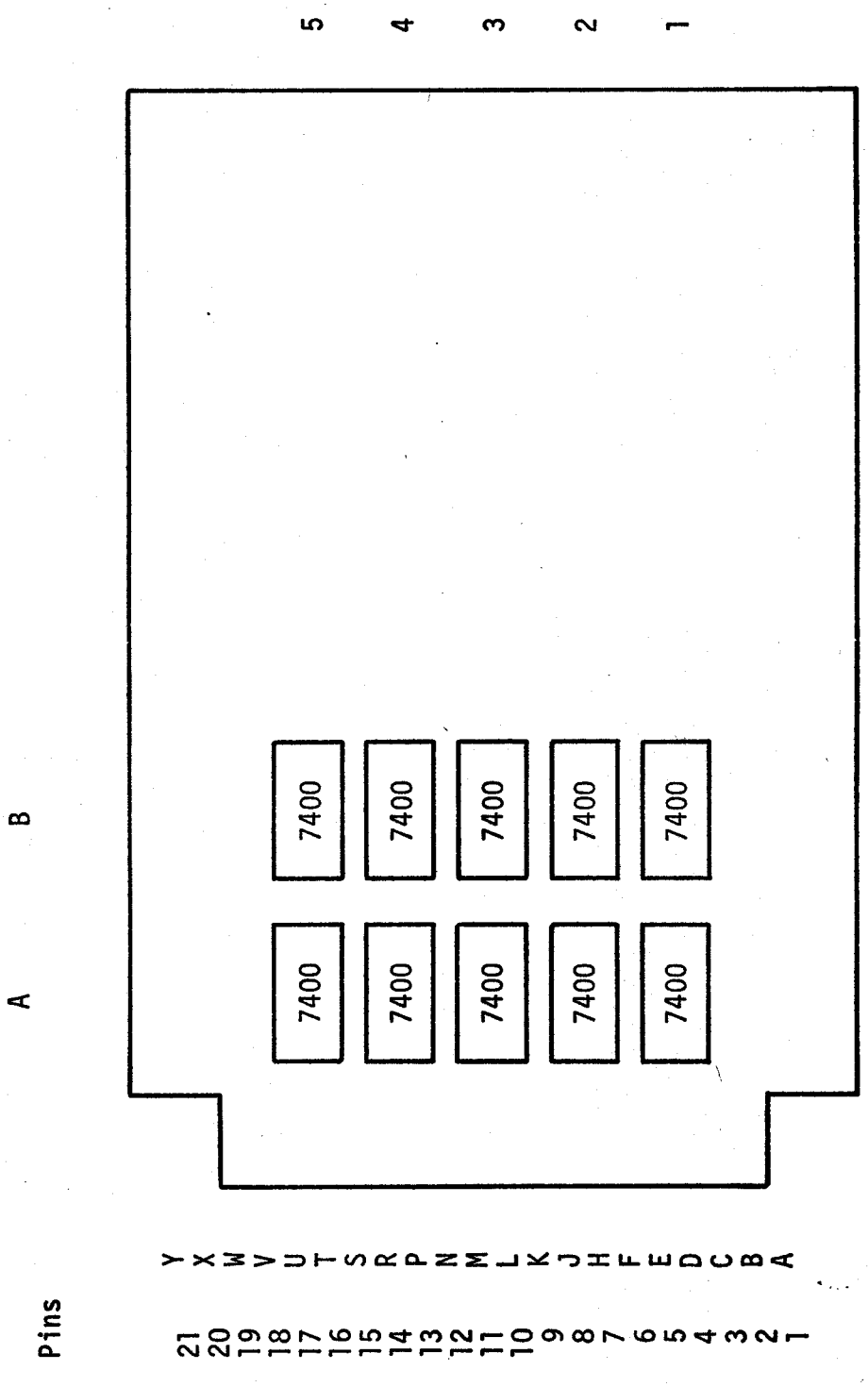


Figure C-3. Circuit Board Layout - Board C

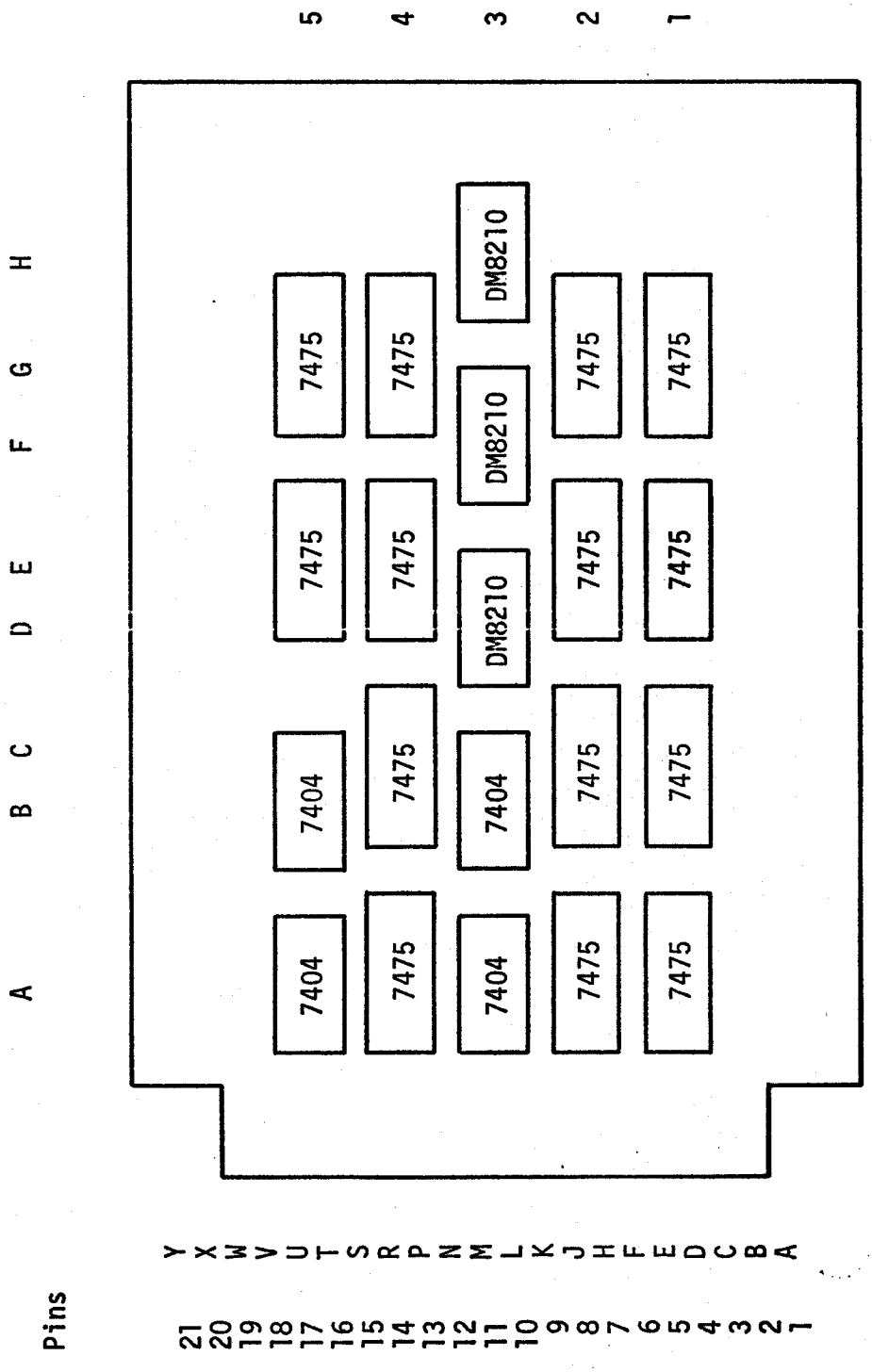


Figure C-4. Circuit Board Layout - Board D

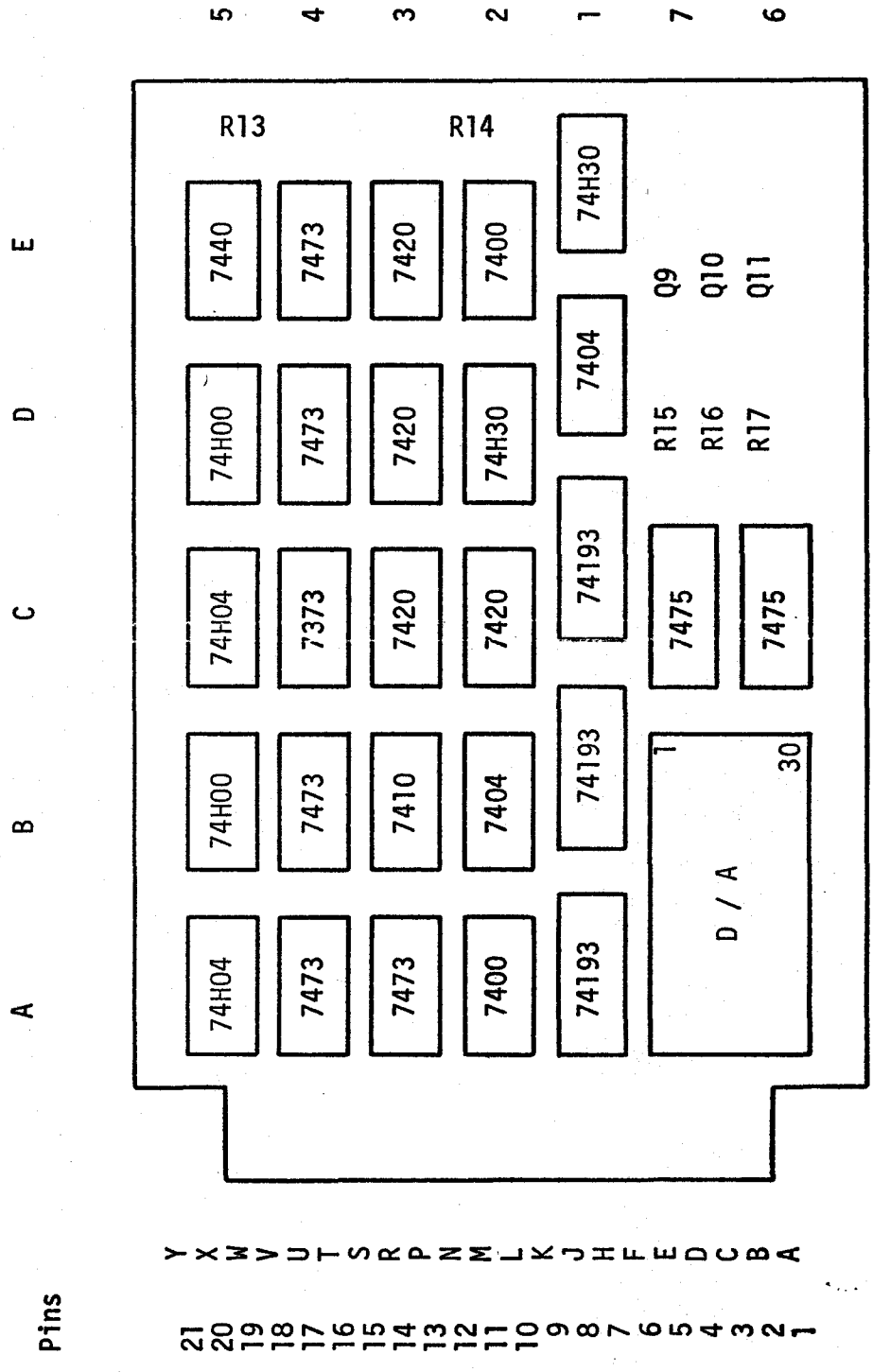


Figure C-5. Circuit Board Layout - Board E

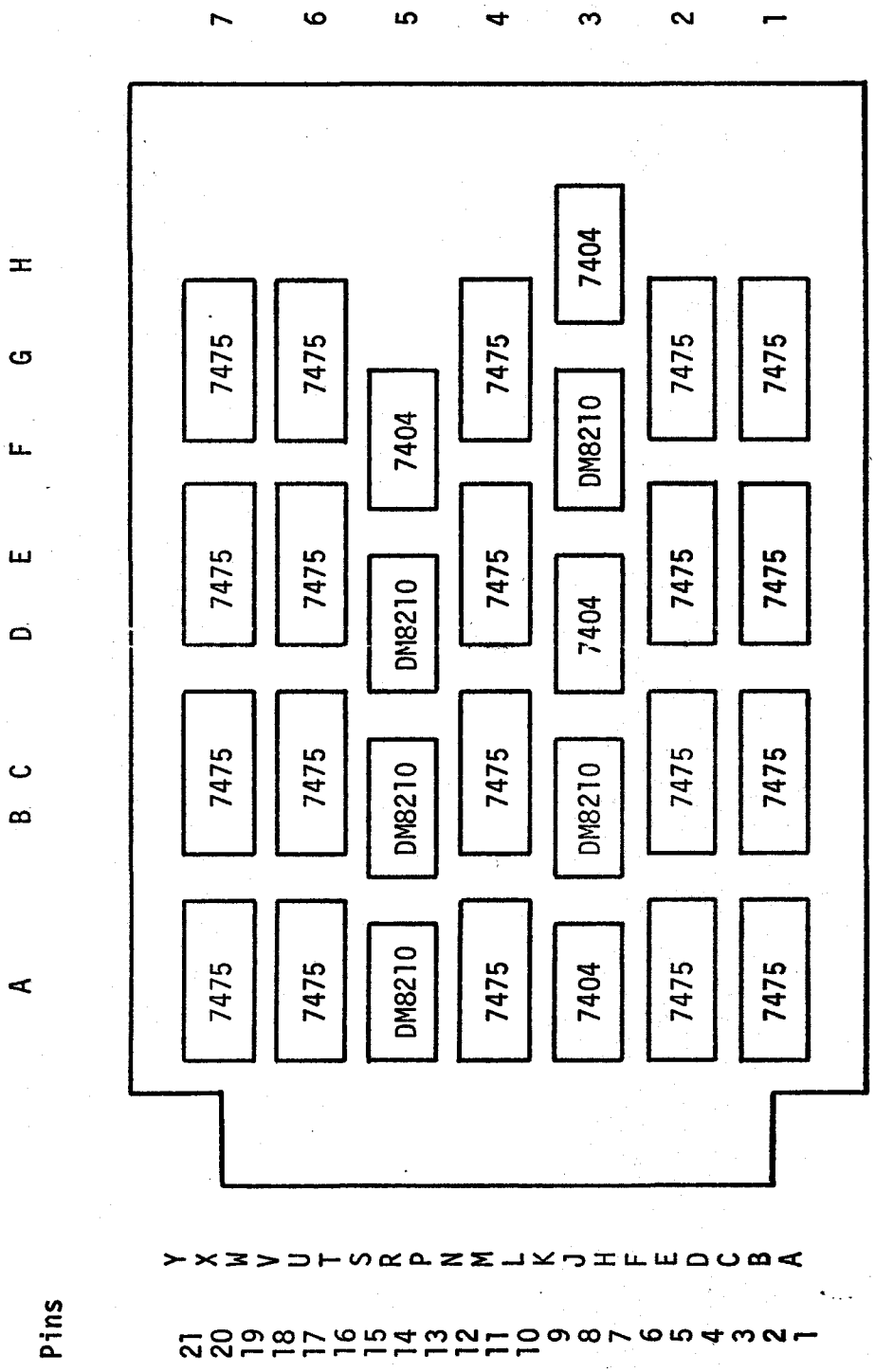


Figure C-6. Circuit Board Layout - Board F

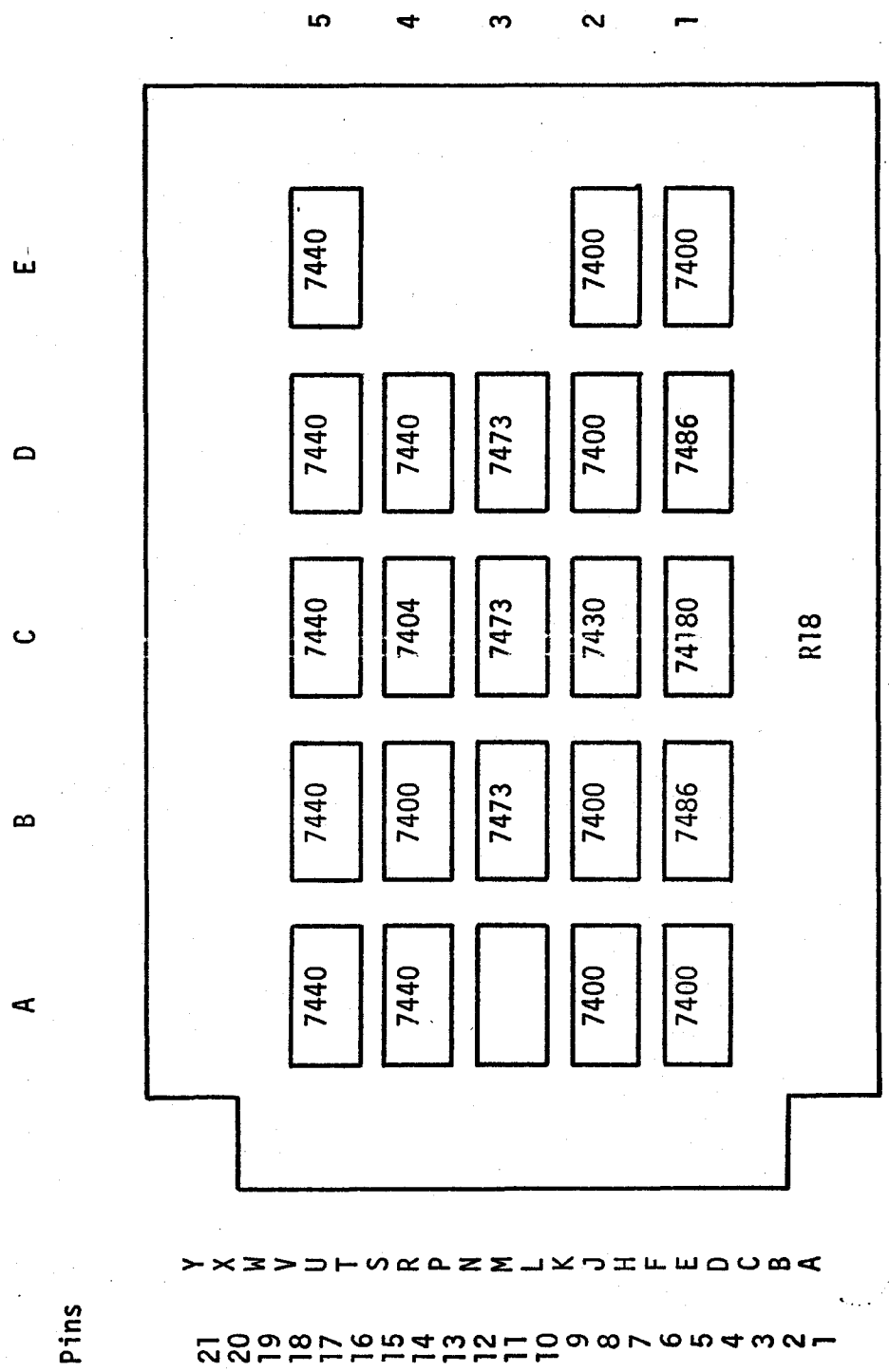
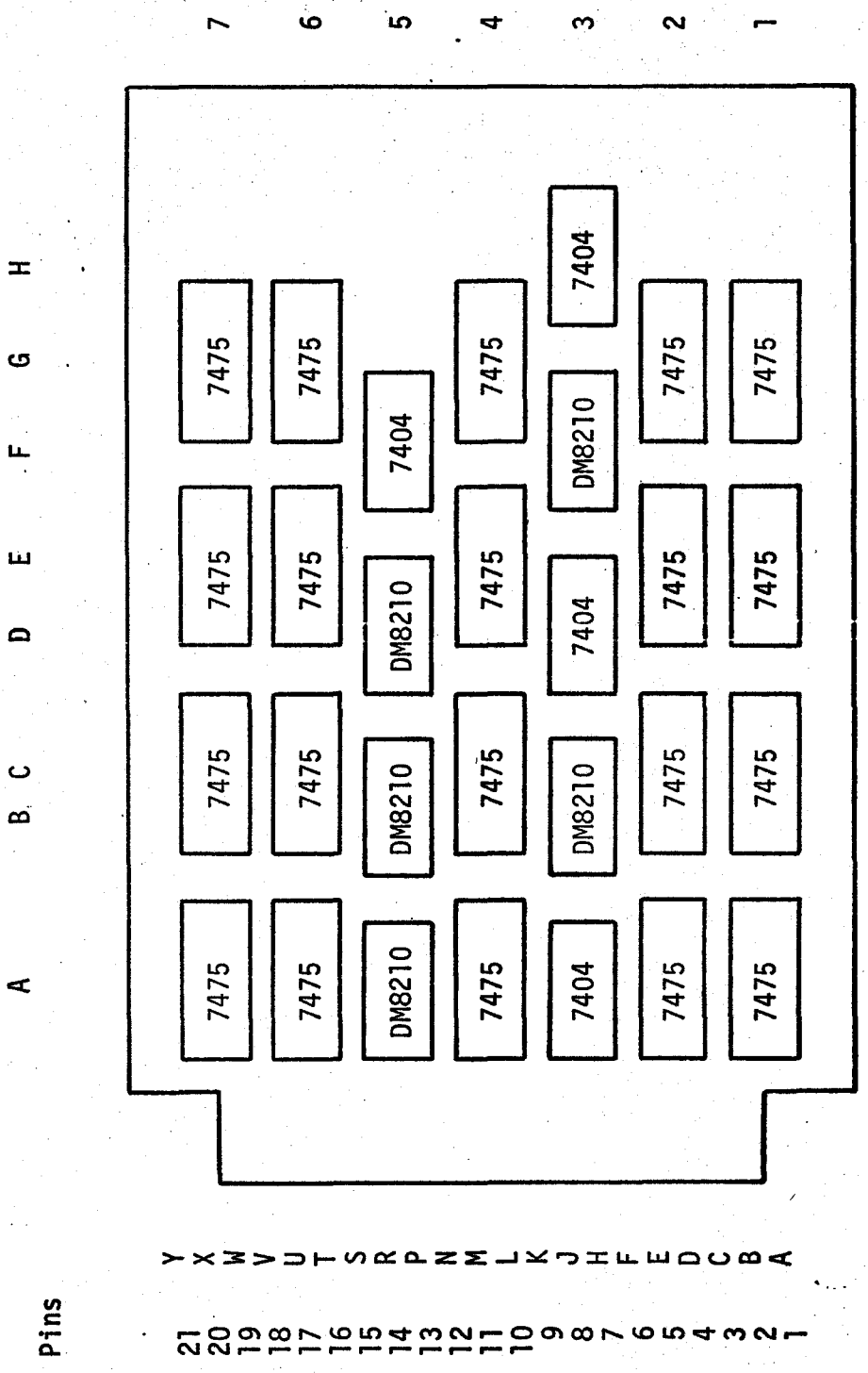


Figure C-7. Circuit Board Layout - Board G



# APPENDIX D

## CIRCUIT BOARD CONNECTOR SIGNALS

The following diagram shows the signals that appear on the seven circuit board connectors. Bracketed signals indicate that their source or destination is external (to lights, switches, etc.) The source of all other signals is the board at which the signal name is underlined. Signals with a double underline are test points only and do not connect to other boards.

| Pin | Board A | | Board B | | Board C | | Board D | |
|---|---|---|---|---|---|---|---|---|
| A 1 | (VCC) | VCC | VCC | (S1) | VCC | VCC | VCC | VCC |
| B 2 | MAN | L1 | MAN | D1 | L1 | D1 | (+15) | WALOUT |
| C 3 | (R1L) | L2 | (C1) | D2 | L2 | D2 | (-15) | B7 |
| D 4 | (R2L) | L3 | (S2) | D6 | L3 | D6 | (RUN GRN) | CLR CNTR |
| E 5 | (R3L) | L4 | (C2) | D7 | L4 | D7 | (OUTPUT) | (O/R) |
| F 6 | (R4L) | L5 | (S3) | D3 | L5 | D3 | GND | (O/F) |
| H 7 | (R5L) | L6 | (C3) | D8 | L6 | D8 | SYNCH 1 | O/F SET |
| J 8 | (R6L) | L7 | (S4) | (C4) | L7 | LOAD 11 | SYNCH 1 | RM OUT |
| K 9 | (R7L) | L8 | (S5) | (C5) | L8 | LOAD 12 | UP/DN | TRNSFR |
| L 10 | (R8L) | | (S6) | D11 | D11 | LOAD 13 | | |
| M 11 | (SEL) | | (C6) | D12 | D12 | B11 | B11 | |
| N 12 | (LD IN) | | (S7) | D13 | D13 | B12 | B12 | |
| P 13 | (SYNCH IN) | | (S8) | (C7) | AB | B13 | B13 | |
| R 14 | (AUTO ADV) | | (S9) | (C8) | BB | CB | | (RUNIN) |
| S 15 | | | (C9) | D9 | D9 | D'1 | RUN | (RUNIN) |
| T 16 | | | (S10) | (C10) | D'6 | D'2 | AB | AA |
| U 17 | | | (S11) | (C11) | D'7 | D'3 | BB | BA |
| V 18 | S | (SEL) | (S12) | (C12) | D'9 | D'4 | CB | CA |
| W 19 | | (LD IN) | (S13) | (C13) | D'10 | D'5 | (PWR ON) | B8 |
| X 20 | SYNCH OUT | COMP | (COMP) | D10 | D10 | D'8 | (CLOCK IN) | B9 |
| Y 21 | | | D4 | D5 | D4 | D5 | CLOCK | B10 |
| Z 22 | (GND) | GND | GND | GND | GND | GND | GND | GND |

Figure D-1. Circuit Board Connector Signals

| Pin | Board E | | Board F | | Board G | |
|---|---|---|---|---|---|---|
| A 1 | VCC | VCC | VCC | VCC | VCC | VCC |
| B 2 | L1 | B6 | B6 | WALOUT | L1 | D1 |
| C 3 | L2 | B7 | | | L2 | D2 |
| D 4 | L3 | D6 | | B1 | L3 | B1 |
| E 5 | L4 | D7 | | B2 | L4 | B2 |
| F 6 | L5 | D3 | | B3 | L5 | B3 |
| H 7 | L6 | D8 | SYNCH 1 | | L6 | |
| J 8 | L7 | B35 | LOAD 11 | | L7 | B35 |
| K 9 | L8 | B36 | LOAD 12 | LOAD 13 | L8 | B36 |
| L 10 | LOAD 6 | B37 | LOAD 6 | LOAD 1 | LOAD 1 | B37 |
| M 11 | LOAD 7 | B38 | LOAD 7 | LOAD 2 | LOAD 2 | B38 |
| N 12 | LOAD 8 | B81 | LOAD 8 | LOAD 3 | LOAD 3 | B81 |
| P 13 | LOAD 9 | B82 | LOAD 9 | LOAD 4 | LOAD 4 | B82 |
| R 14 | LOAD 10 | B83 | LOAD 10 | LOAD 5 | LOAD 5 | B83 |
| S 15 | D9 | B84 | RUN | a | B84 | D'1 |
| T 16 | AB | D'6 | (W) | | AA | D'2 |
| U 17 | BB | D'7 | | | BA | D'3 |
| V 18 | CB | D'9 | | CA | CA | D'4 |
| W 19 | | D'10 | | B4 | B4 | D'5 |
| X 20 | D10 | B8 | | B5 | B5 | D'8 |
| Y 21 | B9 | B10 | | | D4 | D5 |
| Z 22 | GND | GND | GND | GND | GND | GND |

Figure D-1. (cont'd)

# APPENDIX E

## DIGITAL-TO-ANALOG CONVERTER SPECIFICATIONS

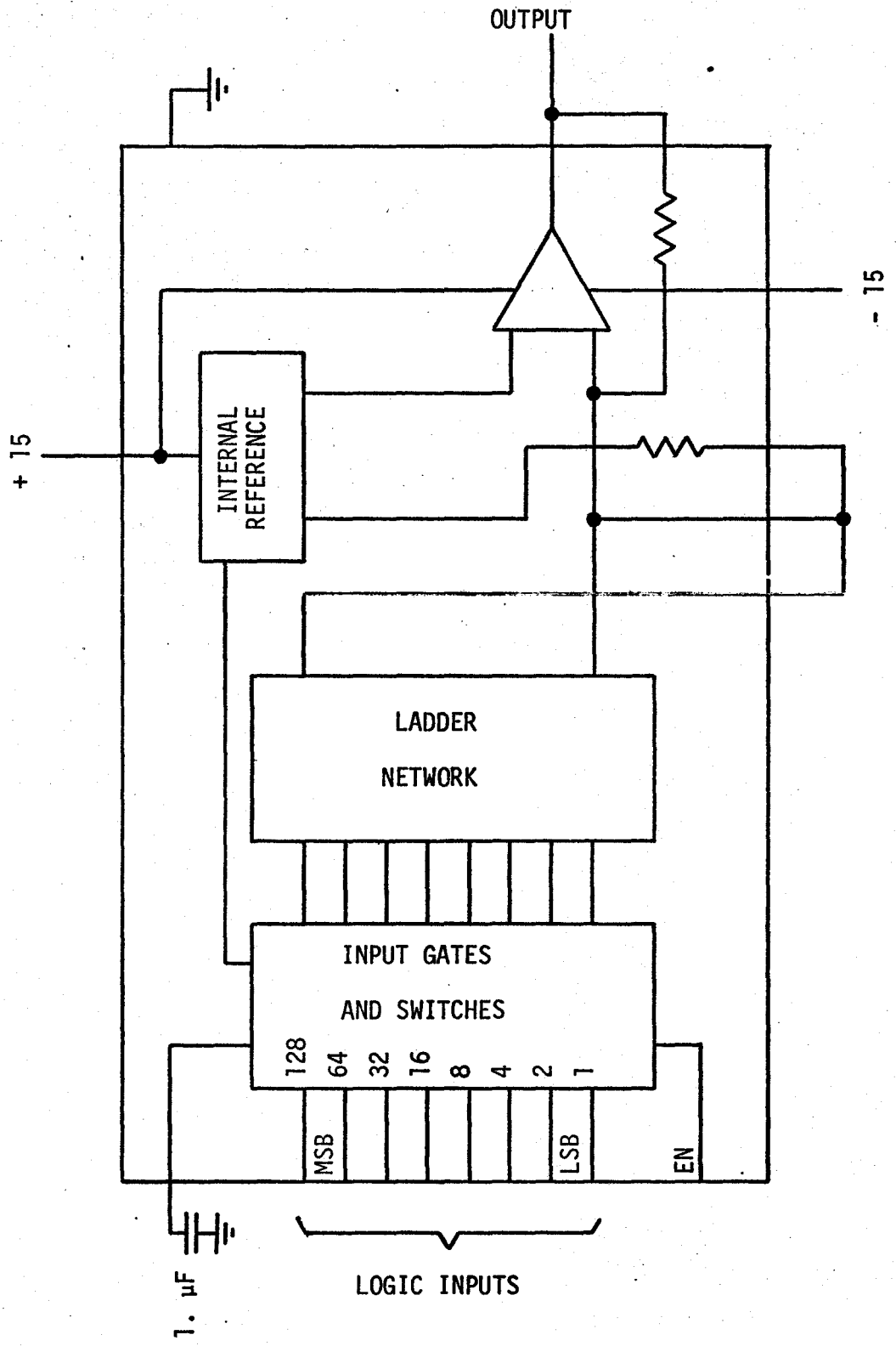
The table below gives the specifications of the Beckman model 845-U5 digital-to-analog converter. The next page gives a block diagram of the circuitry.

Table E-1. Digital-to-Analog Converter Specifications

| Parameter | Min. | Typical | Max. | Units |
|---|---|---|---|---|
| $V_{IN}(1)$ | 2.0 | | * | volts |
| $V_{IN}(0)$ | * | | 0.5 | volts |
| Output Current | | | ±2.5 | mA |
| Output Slew Rate | 0.3 | 0.5 | | V/μsec |
| Transfer Accuracy | | ±0.25 | ±0.5 | LSB |
| Power Supply Voltage | | | | |
| V+ | +14.55 | +15.0 | +15.75 | volts |
| V- | -14.25 | -15.0 | -15.75 | volts |
| Power Supply Current | | | | |
| I+ | | | 60 | mA |
| I- | | | 10 | mA |
| Power Supply Dissipation | | | 1.11 | watts |
| Output Voltage | 0 | | +5.0 | volts |
| Input Word | | 8 | | bits |
| Operating Temperature | -20 | | +85 | $^{o}C$ |

* Data not supplied by manufacturer

Figure E-1. Digital-to-Analog Converter Block Diagram



# APPENDIX F

## DIGITAL SPEECH SYNTHESIZER SPECIFICATIONS

1. Type of Transform: Walsh or Hadamard (manually selected)
2. Input:
   a) Up to 8 input registers can be loaded serially, either manually or by computer.
   b) Each register contains the Walsh or Hadamard function number and its coefficient with sign:

| | |
|---|---|
| BITS 1-6 | Walsh (Hadamard) Number |
| 7 | Sign |
| 8-13 | Coefficient |

   c) Coefficients can have an integer value from -63 to +63.
   d) Walsh functions can be selected from any of the first 64 Walsh functions.
3. Clocking:
   a) An external pulse generator supplies the clocking signal.
      Maximum frequency: 6.67MHz
      Minimum frequency: none
      Pulse width: 40-70nanoseconds
      Amplitude: 0 to +5 VDC
   b) The update frequency (having period $\phi$) is related to the clock frequency by the following:

$$\text{UPDATE FREQ.} = 2^{-15} \times \text{CLOCK FREQ.}$$
$$\approx 3.05 \times 10^{-5} \times \text{CLOCK FREQ.}$$

4. Output:

    a) Output range: 0 to +5 VDC, centred at +2.5V DC (no signal)

    b) Maximum Output Current: 2.5mA

    c) Error: Less than 4% of peak value for sine wave

5. Power Requirements:

    +5V DC @ 3.5A regulated

    ±15V DC @ 200mA

6. Environmental Operating Range:

    0 - 70°C ambient

7. Dimensions:

    21"W x 23"H x 13"D

## LIST OF REFERENCES


1. Harry F. Olson, 'Electronic Music Synthesis for Recordings' IEEE Spectrum, Vol. 8, No. 4, p. 28, April 1971.

2. J.L. Flanagan et al, 'Synthetic Voices for Computers' IEEE Spectrum, Vol. 7, No. 10, p. 23, October 1970.

3. E.O. Brigham, R.E. Morrow, 'The Fast Fourier Transform' IEEE Spectrum, Vol. 4, No. 12, pp. 63-70, December 1967.

4. J.L. Walsh, 'A Closed Set of Orthogonal Functions' Amer. J. of Mathematics, Vol. 55, pp. 5-24, 1923.

5. Karl-Hans Siemens, 'Digital Walsh-Fourier Analyser for Periodic Waveforms' McMaster University, Hamilton, Ontario, Canada, M. Eng. Thesis, 1969, pp. 92-98.

6. C. Böpwetter, 'Analog Sequency Analysis and Synthesis of Voice Signals' Proc. Symp. Appl. Walsh Functions, Washington, 1970, pp. 220-229.

7. Henning F. Harmuth, 'Transmission of Information by Orthogonal Functions' Springer-Verlag, New York/Heidelberg/Berlin, 1970.

8. Fredrick J. Lebert, 'Walsh Function Generator for a Million Different Functions' Proc. Symp. Appl. Walsh Functions, Washington, 1970, pp. 52-54.

9. K.H. Siemens, R. Kitai, 'Walsh Function Evaluation and Generation Using Binary Coding' to be published.

10. J.W. Cooley, J.W. Tukey, 'An algorithm for the Machine Calculation of Complex Fourier Series', Math. of Computation, Vol. 19, No. 90, 1965, pp. 297-301.

11. S.J. Campanella, G.S. Robinson, 'Digital Sequency Decomposition of Voice Signals', Proc. Symp. Appl. Walsh Functions, Washington, 1970, pp. 230-237.

12. 'TTL Integrated Circuits from Texas Instruments' Texas Instruments Inc., Bulletin CB-112, pages 2-12 to 2-15.

13. W.L. Toft, 'A High Speed Rate Multiplier and its Application in a Digital FM Demodulator', McMaster University, Hamilton, Ontario, Canada, B. Eng. Thesis, 1971.

14. J. Millman, H. Taub, 'Pulse, Digital and Switching Waveforms' McGraw-Hill Book Co., New York, 1965, pp. 674-675.

15. J.R. Naylor, 'Digital and Analog Signal Applications of Operational Amplifiers', IEEE Spectrum, Vol. 8, No. 5, pp. 79-87, May 1971.

16. A.R. Elliott, 'A Digitally Controlled Speech Synthesizer' Case-Western Reserve University, Ph.D. Thesis, 1968.

An interesting feature about the ramp function is that if the most dominant term is removed a new ramp function, having twice the frequency and half the amplitude, is generated. This is illustrated in Figure 4-7. In general, by removing the n most dominant terms from the original ramp function, a new ramp function is formed having n times the frequency and 1/n times the amplitude. Another feature of the ramp function is that its phase can be shifted $180^{o}$ by changing the sign of the most dominant term.

The triangle wave of Figure 4-8 has much the same error characteristics as the ramp function. Again, the error is not due to the truncation of the Walsh series but to the limited range of the Walsh functions. In Part A of Figure 4-8, five terms are used to generate the output signal. In the region $0.25 \leq \theta < 0.75$ there are thirty-two different quantization levels. Thus, the error in this representation is $1/(32\times2) \times 100\% = 1.56\%$ of peak-to-peak value, or 3.12% of peak value.

The generation of the triangle using fewer dominant terms is shown in parts B through E of Figure 4-8. The deterioration of the waveforms is evident. In fact, the error approximately doubles every time one coefficient is removed. This is obvious by observing that the number of quantization levels is cut in half every time a coefficient is removed.

The ramp and triangle functions are in a rather small class of waveforms in that they have a sparse number of low order terms in their Walsh series. For this reason, their representation is limited by the number of Walsh functions available from the synthesizer. Such a condition

Figure 4-7. Ramp Function With Period $\phi/2$

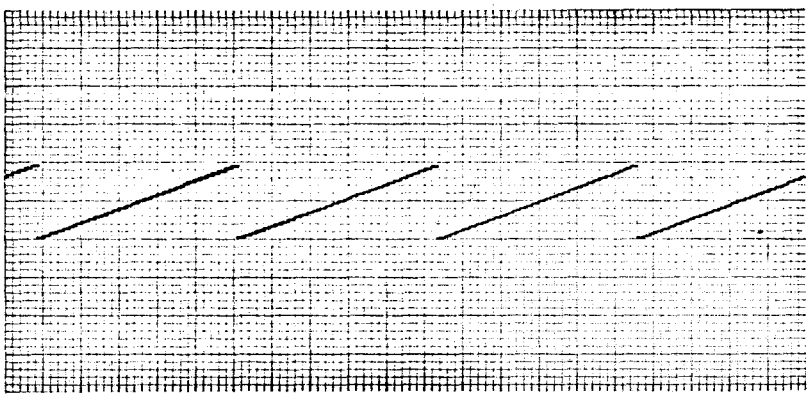
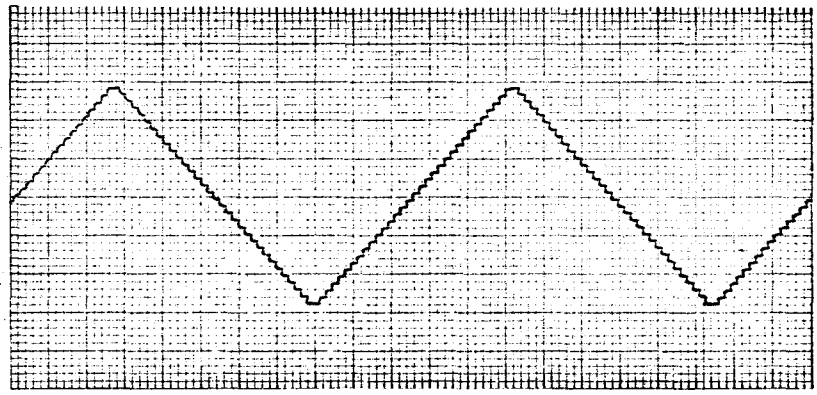
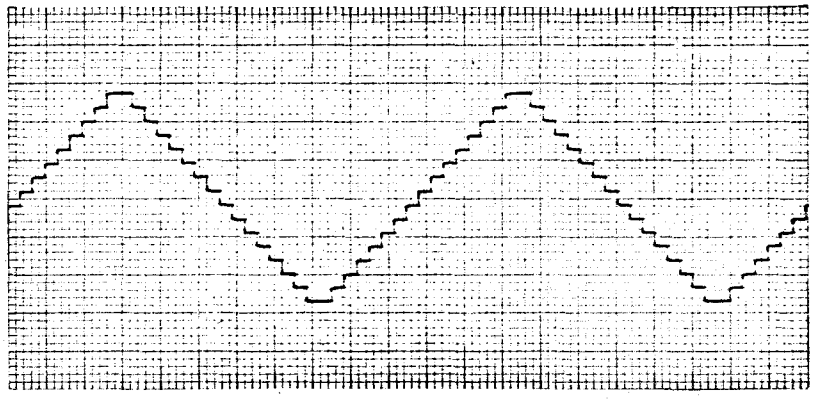


Figure 4-8. Generation of the Triangle Function

A. Five most dominant terms



B. Four most dominant terms



C. Three most dominant terms

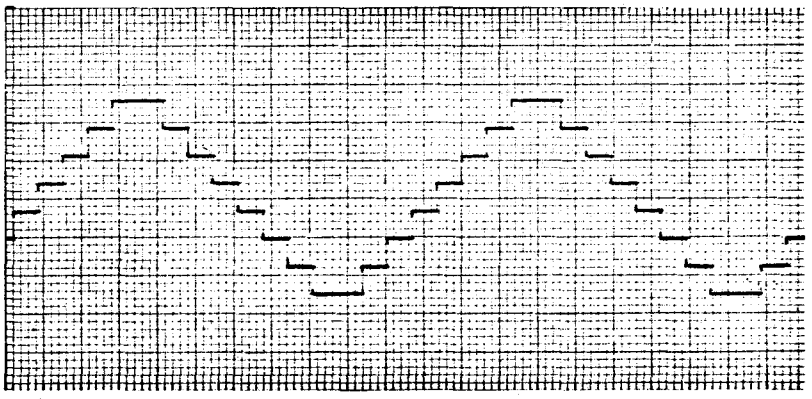
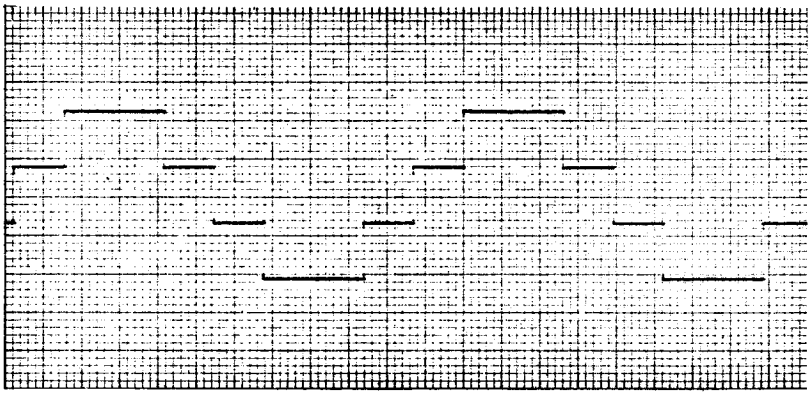
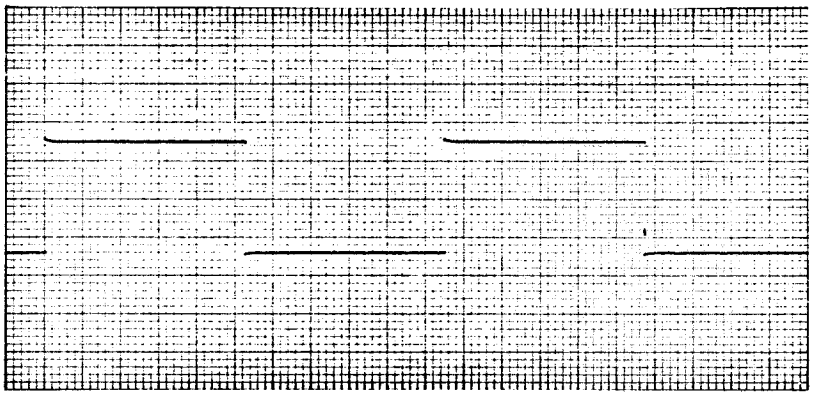


Figure 4-8 (cont'd)

D. Two most dominant terms



E. Most dominant term



is not expected to happen with speech waveforms. In any event, for both the triangle wave and ramp functions, the total error is much less than the maximum allowable for speech. Moreover, most of this error can be eliminated by low pass filtering of the output signal.

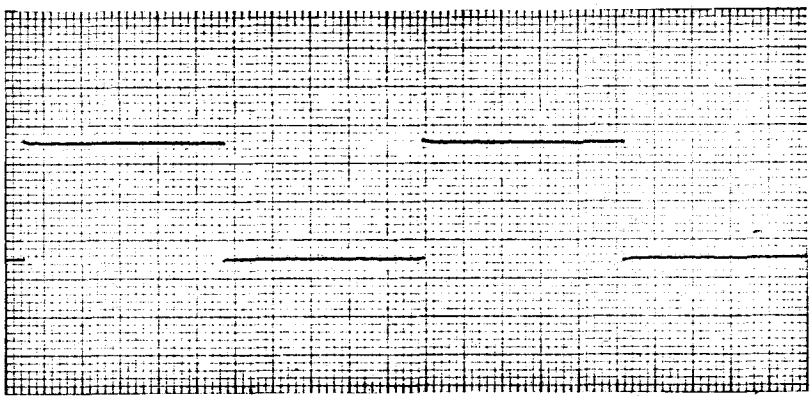
With a Fourier series, waveforms such as sines and cosines can be represented exactly with only one term in the series. This is true since the orthogonal functions on which the Fourier series is based are the sine and cosine waveforms. Similarly, it is possible to represent the Walsh or Hadamard functions with only a single term in the Walsh series. Thus, the synthesizer can be used as a convenient, programable source of Walsh and Hadamard functions.

The Walsh series can also represent other waveforms exactly with a finite number of terms. Figure 4-9 shows the generation of a series of pulses of varying duty ratios, all represented exactly by a small number of terms in the Walsh series. The square wave of Part A is obtained by using only one term. The pulse train, having duty ratio of 0.25, is formed with three terms, all having the same coefficient, while the pulse train of duty ratio 0.125 is formed with seven terms. In each case, the only source of error is that associated with the digital-to-analog converter.

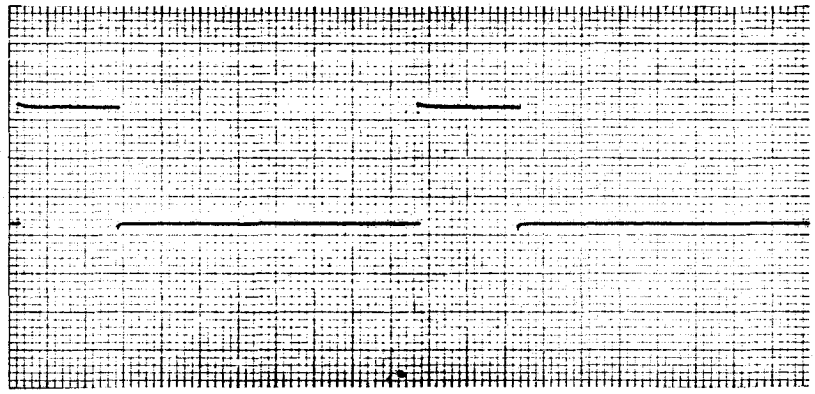
In summary, the basic waveforms shown in this section can easily be generated by using eight or less terms in the Walsh series representation. Error can be substantially reduced by putting the output signal through a bandpass filter to eliminate the 200Hz and 12.8kHz error signals. It is expected, then, that intelligible speech waveforms can be generated

Figure 4-9. Generation of Pulse Trains

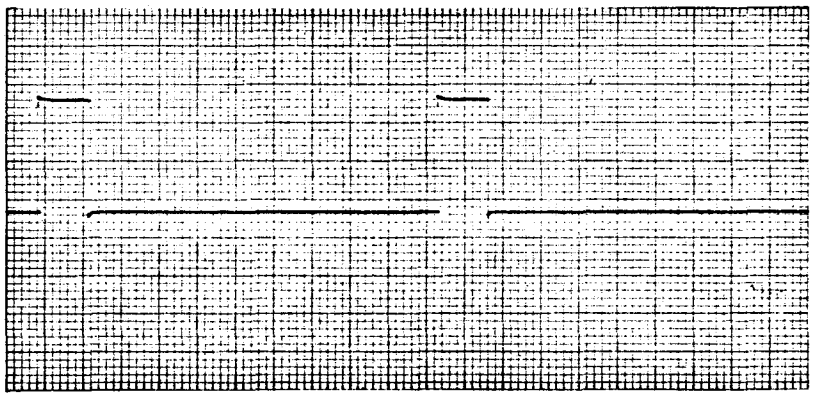
A. Duty Ratio = 0.5



B. Duty Ratio = 0.25



C. Duty Ratio = 0.125



with the synthesizer once it is suitably interfaced to a computer.

# CHAPTER 5

# CONCLUSIONS

The purpose of this project was to design and construct a hardware implementation of a Walsh series synthesizer to eventually be used in conjunction with a computer for speech synthesis. At the present time, the device has been built and checked out by generating the basic waveforms illustrated in the preceding chapter. An estimate of the errors to be expected with the machine was made and confirmed by measurements taken on the generated waveforms. There is no reason to believe that the unit will be unable to generate acceptable quality speech when interfaced to a computer. Unfortunately, this generation cannot be carried out until the required set of coefficients is available from the computer. This data can be obtained from a Fast Walsh Transform which is presently being developed. Suitable computer interfacing will also have to be done in order for speech to be generated. However, this should be a relatively easy job due to the flexibility of the input circuitry of the synthesizer. Another feature to be added is an audio output stage driven by the digital-to-analog converter which will allow subjective testing of the quality of the speech waveforms produced by the machine. Incorporated with this would be the bandpass filter used to eliminate the 200Hz and 12.8kHz error signals.

Once this work has been carried out, it should be possible to generate audible speech waveforms. Optimization of the system can then

be carried out on a qualitative basis. Parameters that should be adjusted to improve the efficiency of the synthesizer are the basic interval $\phi$, the number of terms used in the Walsh series, the range in values of the coefficients and the range of available Walsh functions. It may also be necessary to expand the size of the digital-to-analog converter to accommodate these changes.

Once it has been optimized, a simpler unit could be designed by removing some of the flexibility and redundancy designed into the original synthesizer. Then, the unit would become a practical and economical audio output device for the computer. This, in itself, could be quite useful. As an example, consider the following industrial application. Quite frequently, a salesman on the road will require data from the home office concerning inventories or customer information. Assuming this data is available on the computer at the home office, he could simply telephone the computer, relay information to it using a 'touch tone' dial on the telephone, and get a vocal response from the speech synthesizer interfaced to the computer. A complete data link is thus available between the salesman and the computer wherever there is a telephone with a 'touch tone' dial. This could greatly increase the salesman's efficiency due to the fast, available communications link to the computer.

A more specialized example of a use of the speech synthesizer is in the medical field. Much work has been done to date on the design of a practical reading machine for the blind. Most of the problems associated with the input circuitry and pattern recognition needed to identify

the written symbol have been solved. The main obstacle appears to be in the output device. Tactile stimulators have been used with questionable success. The obvious solution would be to use a speech synthesizer as the output device. The coefficients needed to reproduce the sounds of the letters or even simple words could be called from the computer under control of the pattern recognition software used to determine the written symbols. Unfortunately, with present technology, such a system would be quite complex and therefore not very practical either physically or economically for the average blind person. However, the cost and size of suitable computers is decreasing at a very high rate, so that such a system could possibly become quite practical in the near future.

Due to the data compression inherent with the Walsh series, a speech synthesizer might be useful in a communications link. The analog input signal would be analyzed by a Fast Walsh Transform and only the most dominant terms saved. These could be then sent over the transmission channel to a receiver. The output of the receiver would then be connected to a speech synthesizer to reconstruct the analog waveform. Using the synthesizer described in this thesis, data reduction of a factor of eight can be obtained, thus increasing the efficiency of the transmission channel substantially.

The above-mentioned uses of the synthesizer are all based on the generation of speech waveforms. It is evident, however, that basic waveforms such as sines, cosines, pulses, ramps and triangle waves can also be generated. Data needed to produce these could be obtained from a

small read-only memory since a maximum of only eight computer words is needed to produce any of the basic waveforms. Essentially, any periodic waveform could be reproduced by storing the correct coefficients in the memory. The advantage of using a system such as this as a function generator is that waveforms of very low frequencies can be generated. Recall that for an input clock frequency of about 6.5MHz the output signal had a frequency of 200Hz. The ratio between input and output frequencies is $2^{15}$. Thus, if the input clock frequency were set at 0.1Hz, which is easily obtainable from most clock circuits, the output signal will have a period of approximately ninety-one hours. The output period is limited only by the lower frequency limit of the input clock.

For the synthesizer to be used as a function generator, a reduction in the error in the representation of the waveform would be desirable. This can be accomplished by taking more terms in the Walsh series. Also, by generating Walsh functions of order higher than sixty-four, time quantization errors would be reduced. If this were to be done, a larger digital-to-analog converter would be needed. Thus, the increase in accuracy will be obtained at a cost of more circuitry. However, the total cost of the unit would still be such as to make the system quite economical as a low frequency function generator.

In conclusion, although the unit has been constructed and operates as it was designed to, there is still a great deal of work that could be done to make the unit practical. The specifications of the synthesizer as it was described in Chapter 3 are given in Appendix G. Optimization

brought about by analysis of the speech waveforms generated by the unit may lead to some changes in these specifications. However, it is believed that the choice of parameters used in this synthesizer are quite close to optimum.

# APPENDIX

# APPENDIX A

## FIRST SIXTY-FOUR TERMS IN THE WALSH SERIES FOR BASIC WAVEFORMS

The following coefficients were obtained by computer program using a slow transform technique. The waveforms were sampled sixty-four times over their period and equation (2-28) applied to determine the coefficient values. Scaling of the coefficients was used to present an output waveform of maximum amplitude to simplify error analysis.

Table A-1 Sine Wave Coefficients $f(t) = \sin(2\pi\theta)$ where $\theta = t/T$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 63.000 | 17 | -1.235 | 33 | -0.305 | 49 | -0.616 |
| 2 | 0.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | 0.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | -26.095 | 21 | 0.511 | 37 | 0.127 | 53 | 0.255 |
| 6 | 0.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | 0.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | -5.192 | 25 | -2.570 | 41 | 0.025 | 57 | -1.281 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | -12.530 | 29 | -6.205 | 45 | 0.060 | 61 | -3.095 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

Table A-2. Cosine Wave Coefficients $f(t) = \cos(2\pi\theta)$ where $\theta = t/T$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 0.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 63.000 | 18 | -1.235 | 34 | -0.305 | 50 | -0.616 |
| 3 | 0.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | 0.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 26.095 | 22 | -0.511 | 38 | -0.127 | 54 | -0.255 |
| 7 | 0.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | -5.192 | 26 | -2.570 | 42 | 0.025 | 58 | -1.281 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | 0.000 | 29 | 0.000 | 45 | 0.000 | 61 | 0.000 |
| 14 | 12.530 | 30 | 6.205 | 46 | -0.060 | 62 | 3.095 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

Table A-3. Ramp Function Coefficients $f(t) = \theta - 0.5$ where $\theta = t/T$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | -32.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 0.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | -16.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | 0.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 0.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | -8.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | 0.000 | 29 | 0.000 | 45 | 0.000 | 61 | 0.000 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | -4.000 | 31 | -2.000 | 47 | 0.000 | 63 | -1.000 |

Table A-4. Triangle Function Coefficients $f(t) = 4\theta \quad 0 \leq \theta < 0.25$
$= 2-4\theta \quad 0.25 \leq \theta < 0.75$
where $\theta = t/T$
$= 4\theta-4 \quad 0.75 \leq \theta < 1.00$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 48.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 0.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | 0.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | -24.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 0.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | 0.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | -12.000 | 29 | -6.000 | 45 | 0.000 | 61 | -3.000 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

Table A-5 Rectangular Pulse, Duty Ratio 0.5 $f(t) = 1.0 \quad 0 \leq \theta < 0.5$
where $\theta = t/T$
$= 0.0 \quad 0.5 \leq \theta < 1.0$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 50.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 0.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | 0.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | 0.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 0.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | 0.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | 0.000 | 29 | 0.000 | 45 | 0.000 | 61 | 0.000 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

Table A-6. Rectangular Pulse, Duty Ratio 0.25 f(t) = 1.0 $0 \leq \theta < 0.25$ where $\theta = t/T$
= 0.0 $0.25 \leq \theta < 1.0$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 25.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 25.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | 25.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 0.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | 0.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 0.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | 0.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | 0.000 | 29 | 0.000 | 45 | 0.000 | 61 | 0.000 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

Table A-7. Rectangular Pulse, Duty Ratio 0.125 f(t) = 1.0 $0 \leq \theta < 0.125$ where $\theta = t/T$
=0.0 $0.125 \leq \theta < 1.0$

| n | C(n) | n | C(n) | n | C(n) | n | C(n) |
|---|---|---|---|---|---|---|---|
| 0 | 0.000 | 16 | 0.000 | 32 | 0.000 | 48 | 0.000 |
| 1 | 12.000 | 17 | 0.000 | 33 | 0.000 | 49 | 0.000 |
| 2 | 12.000 | 18 | 0.000 | 34 | 0.000 | 50 | 0.000 |
| 3 | 12.000 | 19 | 0.000 | 35 | 0.000 | 51 | 0.000 |
| 4 | 12.000 | 20 | 0.000 | 36 | 0.000 | 52 | 0.000 |
| 5 | 12.000 | 21 | 0.000 | 37 | 0.000 | 53 | 0.000 |
| 6 | 12.000 | 22 | 0.000 | 38 | 0.000 | 54 | 0.000 |
| 7 | 12.000 | 23 | 0.000 | 39 | 0.000 | 55 | 0.000 |
| 8 | 0.000 | 24 | 0.000 | 40 | 0.000 | 56 | 0.000 |
| 9 | 0.000 | 25 | 0.000 | 41 | 0.000 | 57 | 0.000 |
| 10 | 0.000 | 26 | 0.000 | 42 | 0.000 | 58 | 0.000 |
| 11 | 0.000 | 27 | 0.000 | 43 | 0.000 | 59 | 0.000 |
| 12 | 0.000 | 28 | 0.000 | 44 | 0.000 | 60 | 0.000 |
| 13 | 0.000 | 29 | 0.000 | 45 | 0.000 | 61 | 0.000 |
| 14 | 0.000 | 30 | 0.000 | 46 | 0.000 | 62 | 0.000 |
| 15 | 0.000 | 31 | 0.000 | 47 | 0.000 | 63 | 0.000 |

# APPENDIX B

## CIRCUIT DIAGRAMS

The following pages show the circuit diagrams for the seven boards used in the synthesizer. All the electronics for the synthesizer, other than switches, connectors and lights, are contained on these boards. They are labelled A through G, denoting the order in which they are installed in the board rack. Board A is located on the left side of the machine and board G is on the right side. Note that bracketed figures represent pin numbers for external connection to the boards. The number and letter contained in each gate gives the location of the gate on the circuit board while the numbers on each connection refer to the integrated circuit pin number. Refer to Appendix C for physical layouts of the boards.

The following is a list of the functions provided by each board:

Board A: Input Register Multiplexing

Board B: Input Gating

Board C: Input and Intermediate Registers, Bits 11, 12, 13

Board D: Rate Multiplier, Clock Chain and Output

Board E: Input and Intermediate Registers, Bits 6-10

Board F: Walsh-Hadamard Function Generator

Board G: Input and Intermediate Registers, Bits 1-5

Figure B-1. Circuit Diagram - Board A

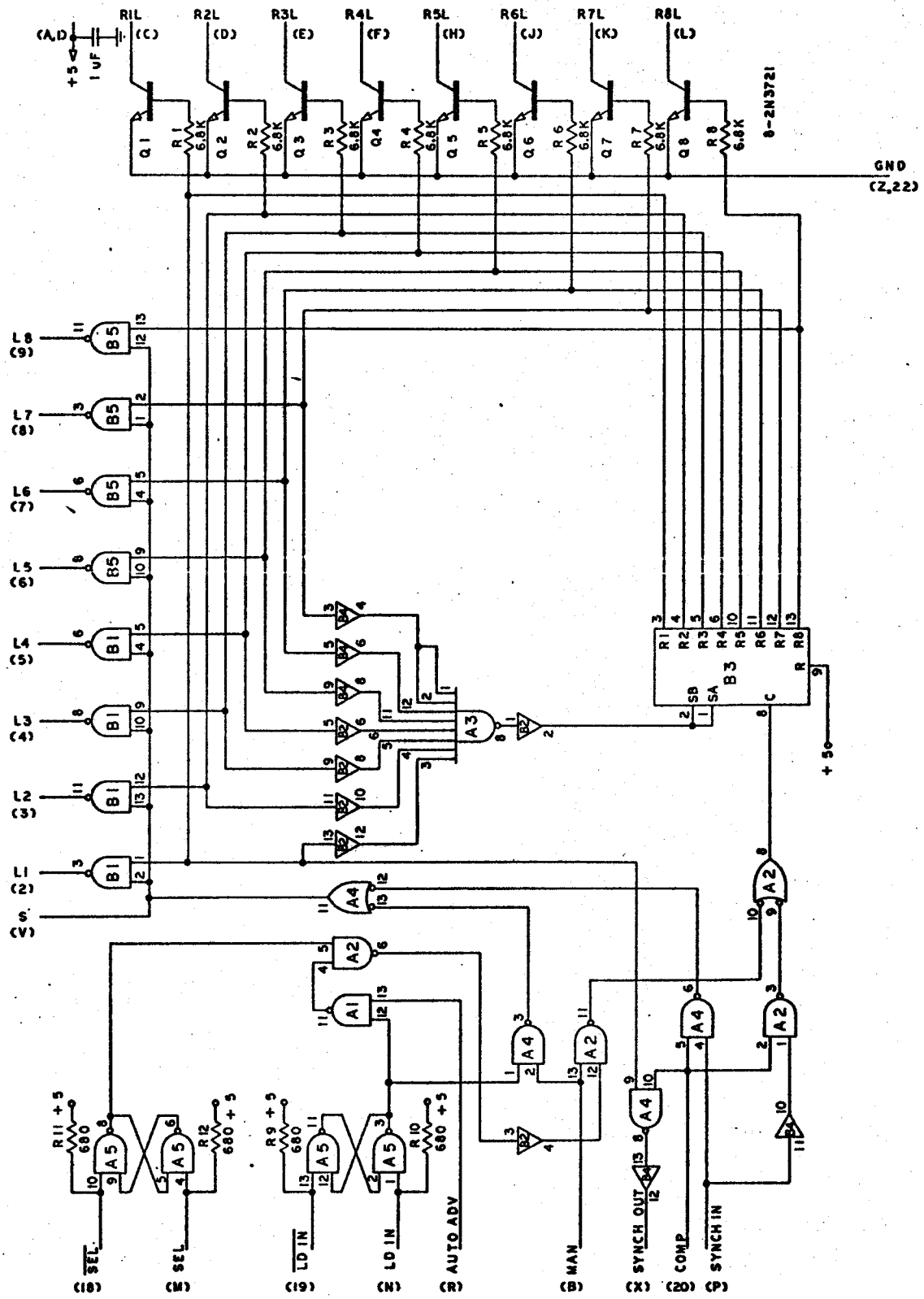


Figure B-2. Circuit Diagram - Board B

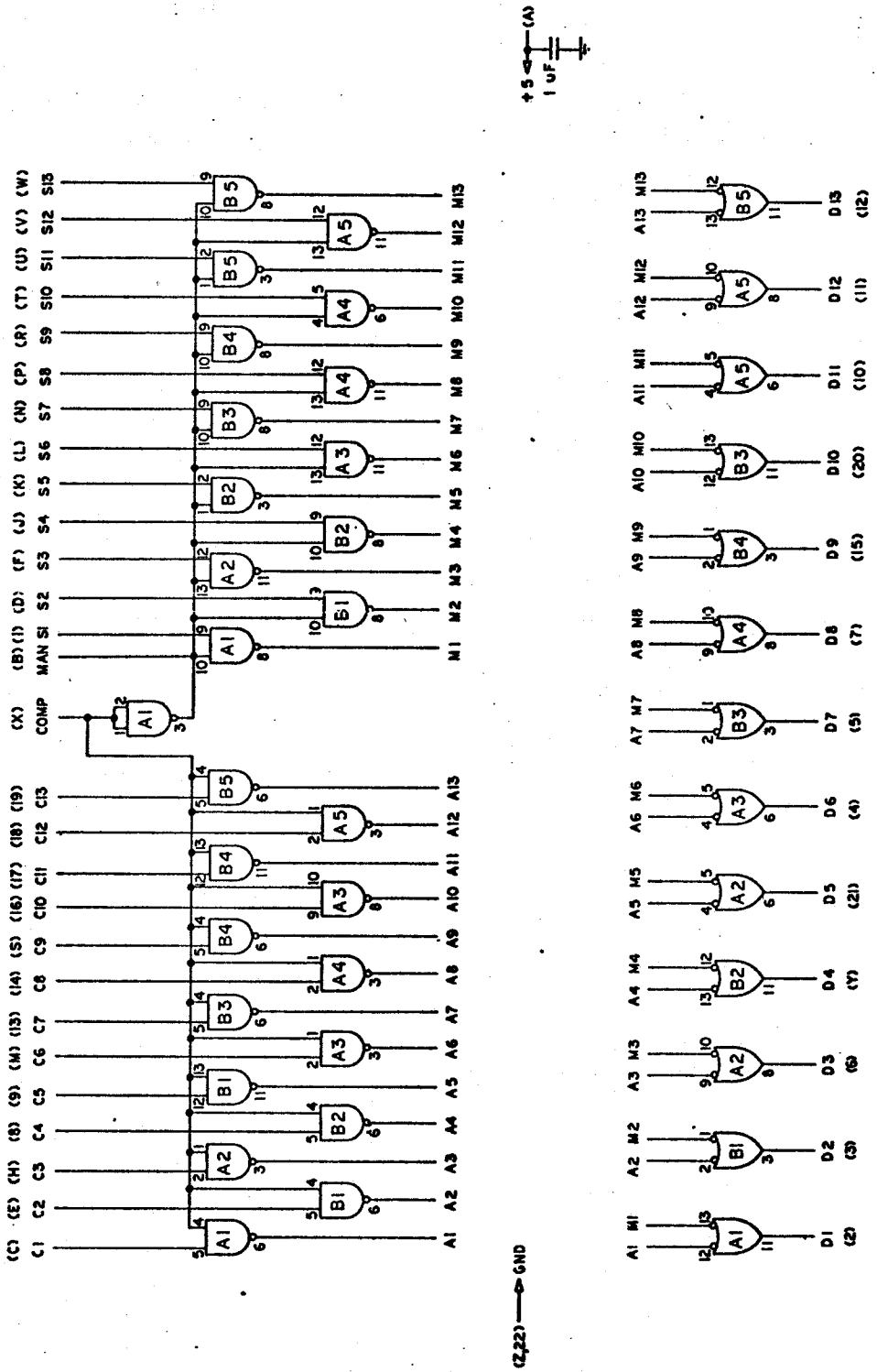


Figure B-3. Circuit Diagram - Board C

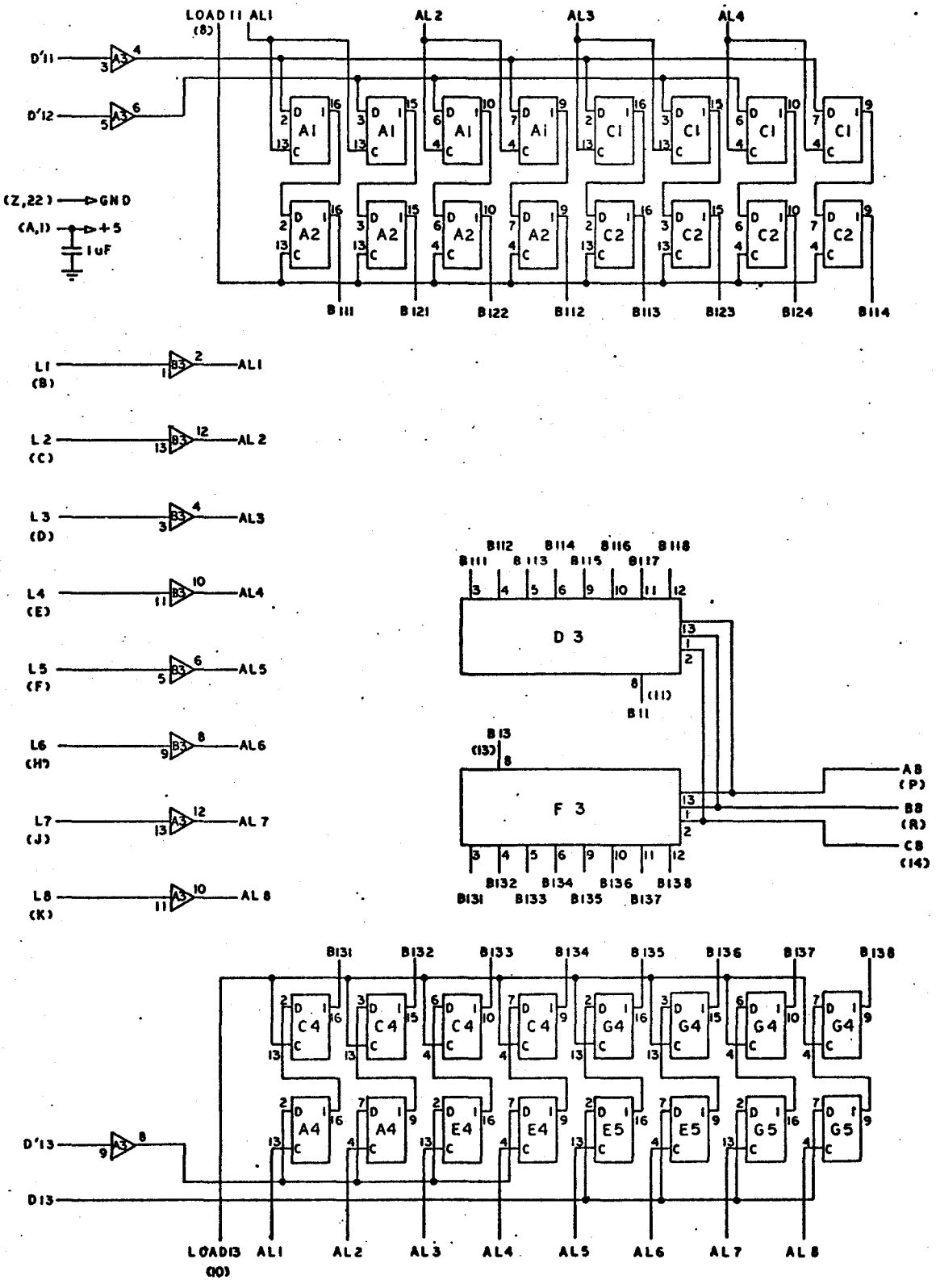
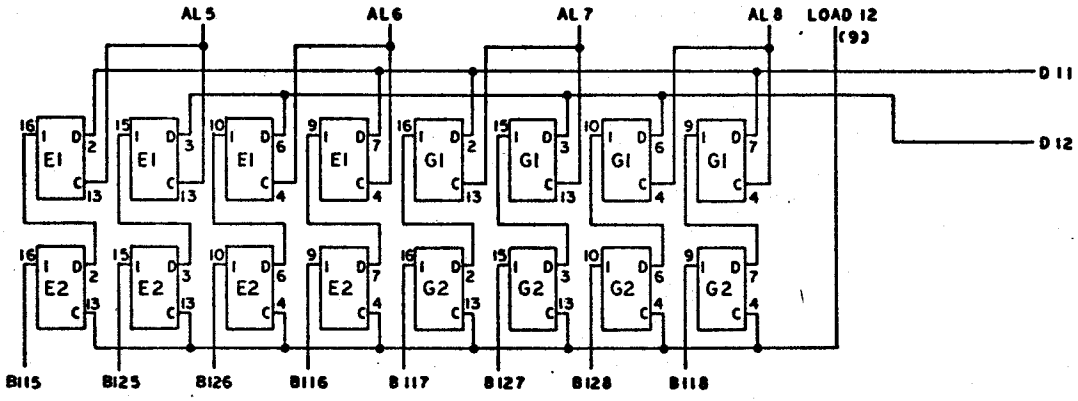
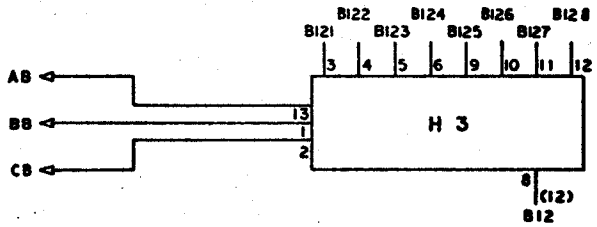


Figure B-3. (cont'd)



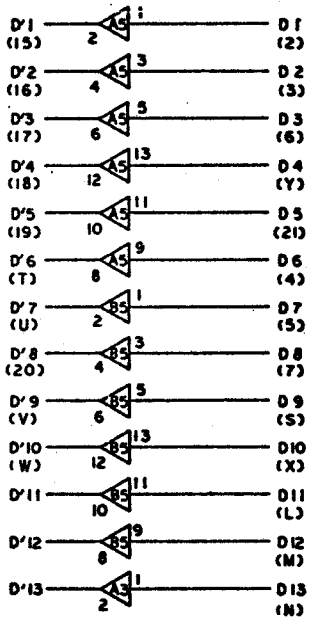



Figure B-4. Circuit Diagram - Board D

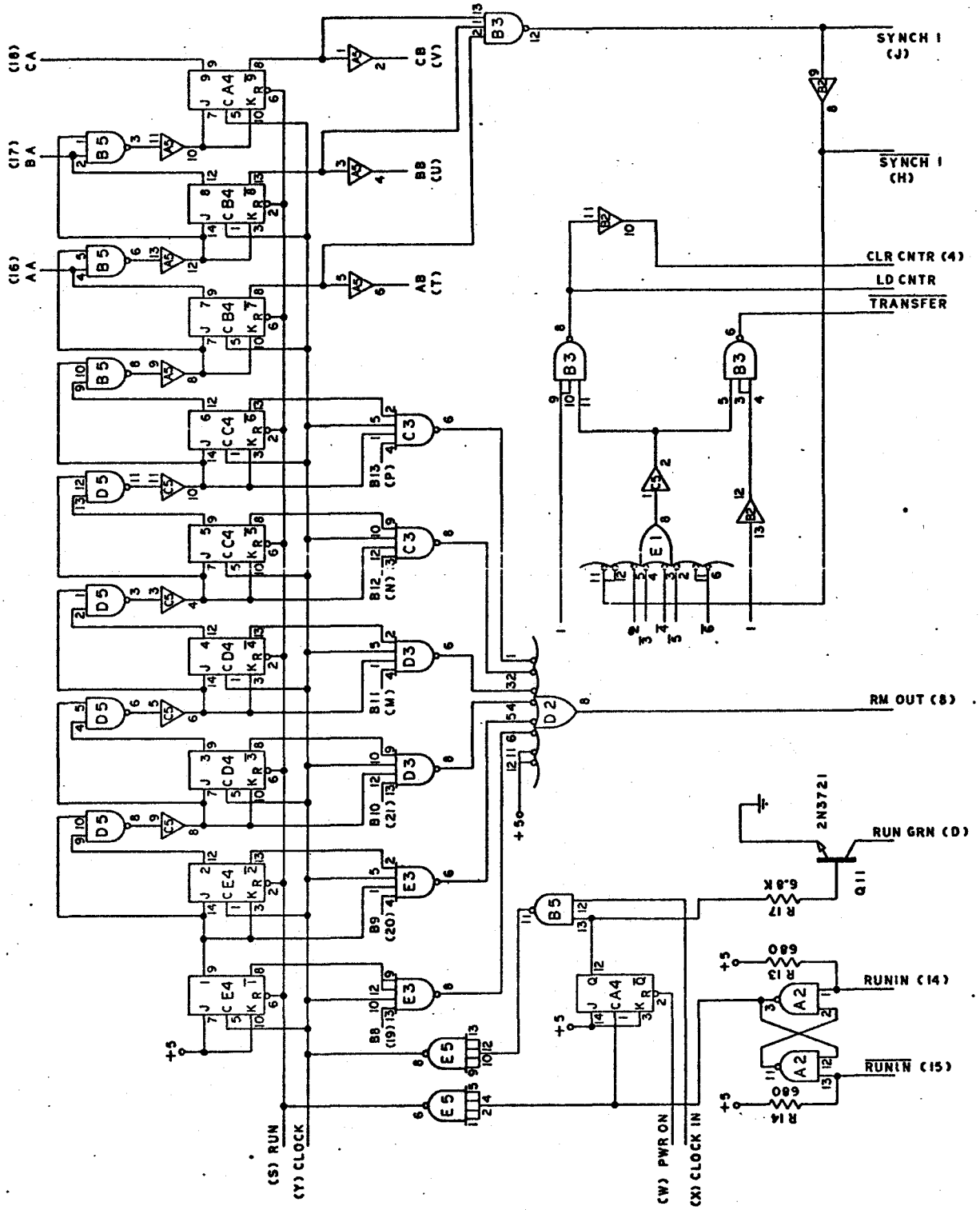


Figure B-4. (cont'd)

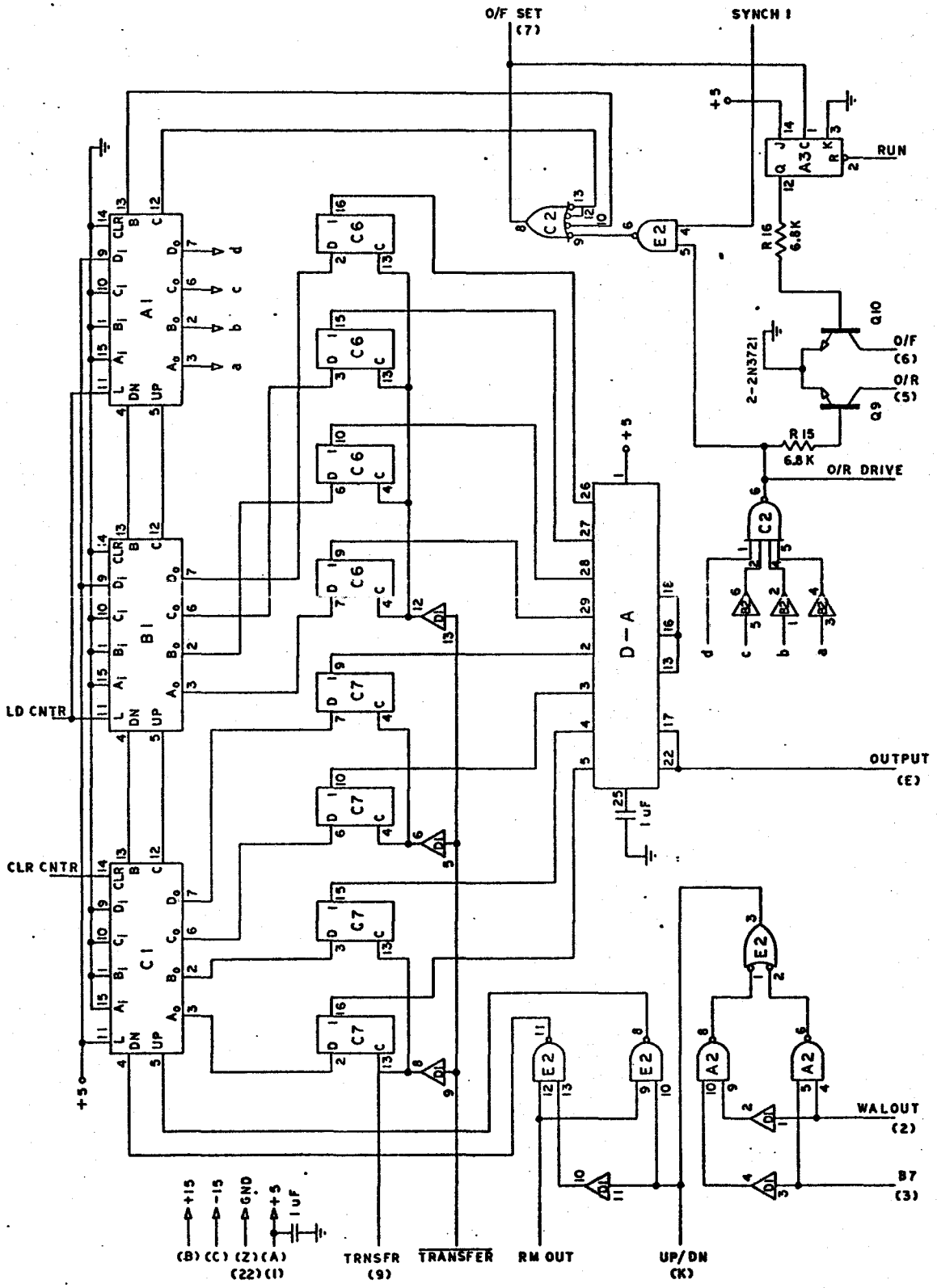


Figure B-5. Circuit Diagram - Board E