CARRIER COMMUNICATION ON BUS-BARS AND CABLES

by

Peter T. Ashton

A Thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science

Winnipeg, Manitoba, 1989

ISBN 0-315-51632-1

CARRIER COMMUNICATION ON BUS-BARS AND CABLES

BY

PETER T. ASHTON

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis. I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Peter T. Ashton

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in whole or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Peter T. Ashton

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

The purpose of this thesis is to develop a device which allows relays employing zone selective instantaneous protection (ZSIP) to communicate over 600V busses and cables. What makes this application different from other carrier current systems is that the required signal must be sent at the exact moment that the electrical noise on the bus is at its worst. To facilitate development of this device, measurements are taken to determine the nature of the electrical noise present on 600V busses under different conditions. Using this information, a microprocessor based carrier current system is developed and two prototype units are constructed. Testing of these prototypes reveals that the system is very fast and has good noise rejection characteristics. Due to the success of these prototypes, I foresee little difficulty extending the ZSIP system to incorporate the carrier current concept.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor G. Swift for his valuable guidance throughout the development of this thesis. I would also like to thank Federal Pioneer Ltd. for suggesting this topic and providing the financial support which made it possible. Finally, the many helpful comments and suggestions provided by the department's electrical technologists are greatly appreciated, as they are what transformed this thesis from only an idea into a working system.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter I

INTRODUCTION

1.1 Purpose

The purpose of this research is to develop a device which will interface with Federal Pioneer's ZSIP[1] system of ground fault relays and allow them to communicate over 600V busses or cables. This device would be a useful enhancement to the ZSIP system and would make the product more attractive in a competitive marketplace.

1.2 Background

To fully understand the need for this research and how the carrier current unit would function, it is necessary to first know some of the fundamentals of protective relaying and some details of the ZSIP system itself.

1.2.1 Fundamentals of Relaying

Inside a typical factory, institution, or sub-station, power is distributed via a system of 600V busses and cables. The distribution network has a hierarchical structure with a primary distribution bus feeding several secondary busses,

[1]"Zone Selective Instantaneous Protection", a Federal Pioneer Limited registered trademark.

which in turn feed other busses, and so on. Figure 1.1 shows a simplified version of a distribution network. Near the junction of two busses there are circuit breakers (one for each phase) which are capable of interrupting the current being fed into the smaller bus. Connected to these breakers is a protective relay which senses the current on the bus through current transformers and causes the breakers to open if this current exceeds a preset threshold.

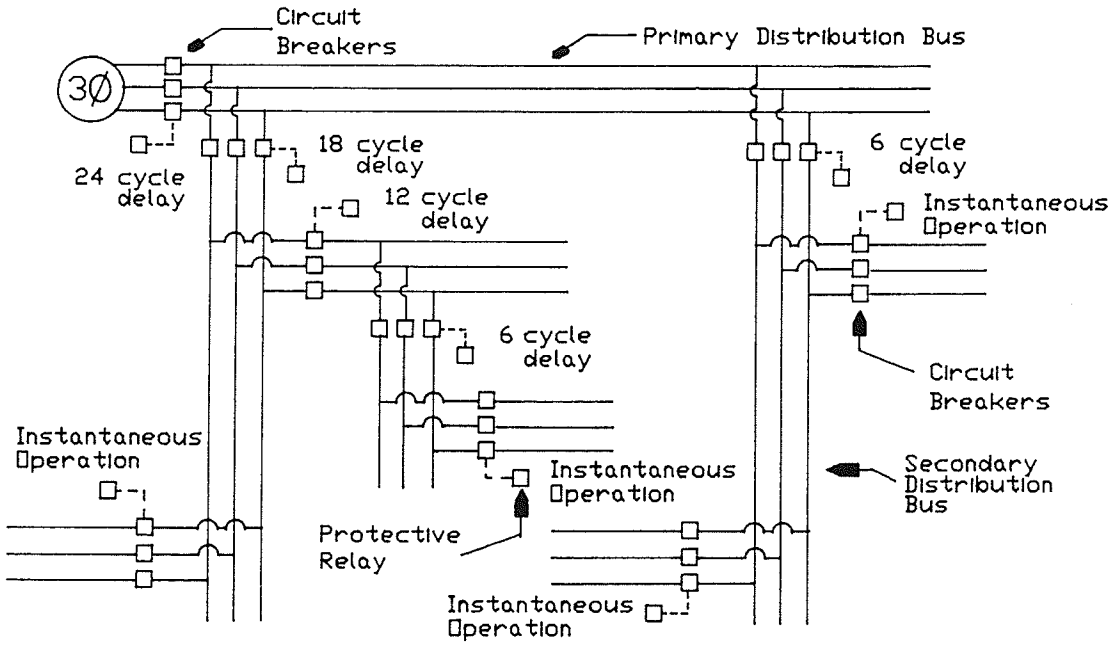


Figure 1.1: Time Coordinated Protection.

The relays and breakers described above are used to provide protection against faults, i.e. short circuits, which sometimes appear. These faults can burn and damage the busses because of the huge currents they draw. In addition, they are also dangerous because they can spread to other busses as the arc follows the path of the drifting ions. To remove a fault, the current feeding it must be interrupted by opening the breakers and then allowing the ions to dissipate.

Since it is desirable to cut the power to as few busses as possible, methods have been devised to make sure the breakers immediately upstream of the fault are the ones that are opened. The most common way of doing this is by time coordinating the upstream and downstream relays. Normally, the furthest downstream relay is in an instantaneous mode where it will open its breakers as soon as a fault is detected. Although the next upstream relay also "sees" the fault, it is in a timed mode in which it will wait for a preselected period of time (say, 6 cycles) after a fault is sensed before it will open its breakers. If the fault is located downstream of the furthest downstream relay, then the upstream relay will not operate since the fault will already be removed by the time the timed mode has elapsed. This system of protection can be extended to any number of distribution levels simply by adding successively longer delays to the upstream relays.

The problem with TCP (Time Coordinated Protection) is that for a fault occurring on the primary distribution bus

(near the ac source), there is a very long delay (say, 24 cycles) before the timed mode finally ends and the breakers are opened. During this time, considerable damage can be done to the bus-bars, and the arc may spread to other busses as well.

1.2.2 The ZSIP System

ZSIP avoids the problems associated with TCP by providing upstream and downstream relays with a method of communicating. Twisted wire pairs are run between each upstream/downstream relay pair, thereby providing the downstream relay with the ability to tell the upstream relay whether or not it sees a fault. Time delays are not necessary in this scheme so each relay is by default in instantaneous mode. When a relay sees a fault, it first sends a "block" signal to the relay just upstream from it. Then, if it has not received a block signal itself, operates its breakers.

Using this system, a fault on the primary distribution bus is removed almost instantly since there are no other relays to see the fault and block the relay. This results in far less bus-bar damage while maintaining the ability to cut off only the minimum number of busses.

1.3 Problem

Although the ZSIP system provides better protection than TCP, it is also more expensive. Since the upstream/downstream relay pairs can be far apart (say 300m), the cost of running wire and conduit is not inconsequential. The question is then "Why not use the bus-bars themselves as the communication media?". This type of communication is not new; carrier current devices have been around for years. However, what makes this application novel is that one is trying to send a signal while an arcing fault exists on the bus. In other words, the signal is being sent at the exact moment that the noise on the bus is at its worst.

1.4 Scope

To begin to solve the problem described above, it is necessary to have a feel for the kind of electrical noise found on a 600V distribution system. Only once the nature of the noise has been characterized can the design of the unit's hardware and software begin. In this thesis, Chapter II describes the types of measurements needed, how they were made, and what they revealed. Chapter III is devoted to the design of the hardware and software and describes why this design was chosen. A chapter explaining the testing conducted on the carrier current units comes next, and is followed by the final chapter containing conclusions and recommendations.

Chapter II

ELECTRICAL NOISE MEASUREMENTS

2.1 Required Measurements

Knowing the nature of the electrical noise present on a 600V distribution system is crucial to the design of the carrier current ZSIP system. There are really two distinct measurements which need to be made:

1) measurements under normal conditions (i.e. no fault present) to make sure steady state noise is not misinterpreted as a blocking signal.

2) measurements just after a fault occurs to make sure that arc induced noise is not interfering with the transmission of the blocking signal.

The first of the two measurements above is extremely simple to make. By clipping a spectrum analyzer onto the bus (through a HP filter) and scanning over a frequency range from 10 - 100 kHz, a good picture of the noise environment develops. On the other hand, measurement 2) is much more difficult to make as it requires that a few ac cycles of data

be "captured" immediately after the fault occurs. A calculating storage oscilloscope can do this, but its calculating function is usually limited to an FFT (Fast Fourier Transform) of only 512 points of data. To "see" frequencies up to 100kHz, the Nyquist criterion requires that the data be sampled at greater than 200kHz. If 5 ac cycles of data are wanted, then more than 16,667 samples need to be taken. A calculating oscilloscope is certainly not sufficient for this task. The next section describes the actual method used to accumulate the data for both types of measurements.

2.2 Measurement Equipment

The measurement system used in this thesis consists of a 12MHz IBM-PC AT computer, a custom made data acquisition board, a passive coupling circuit, and some machine code programs.

The data acquisition board used is the QUAD A/D1, made by MicroMarine Technology. It consists of four Harris 12μs, 12 bit analog to digital (A/D) converters, four sample and hold devices, a 16 channel MUX circuit, and the required hardware to interface it to the PC. The 12μs conversion time gives each one of the converters the ability to sample at 83.3kHz. However, by using 3 of these A/D converters at once, a sampling rate of 250kHz is theoretically possible. The 4th A/D channel is reserved for voltage sensing to initiate the sampling in the staged fault measurements.

The Quad A/D1 is connected to the 600V bus through a passive coupling circuit to protect its sensitive electronics from the high voltage. This coupling circuit is essentially just an RC network which forms a HP filter. High power zener diodes are used to clamp the output voltage at 5.2V so that high frequency noise spikes do not exceed the maximum input voltage of the sample and hold circuits. Figure 1.2 shows a schematic of the coupling circuit while an amplitude Bode plot is given in Fig. 1.3. It can be readily seen that this circuit provides a great deal of attenuation at 60Hz (>40dB) while providing almost no attenuation at frequencies upwards of 10kHz.

Due to the extreme speed required by this sampling task, the software to run the A/D equipment must be written in machine code. There are, in total, three versions of this software, each of which performs a slightly different task.

SAMPLE.EXE captures 6630 samples (can be varied) at 198.9kHz immediately after being run, and saves this data to the default disk drive using a filename supplied by the user.

TIME_SAM.EXE is used for automated data acquisition. When run, this program takes one set of data immediately (6630 samples at 198.9kHz) and saves it to disk using the user supplied name with a numerical suffix added.

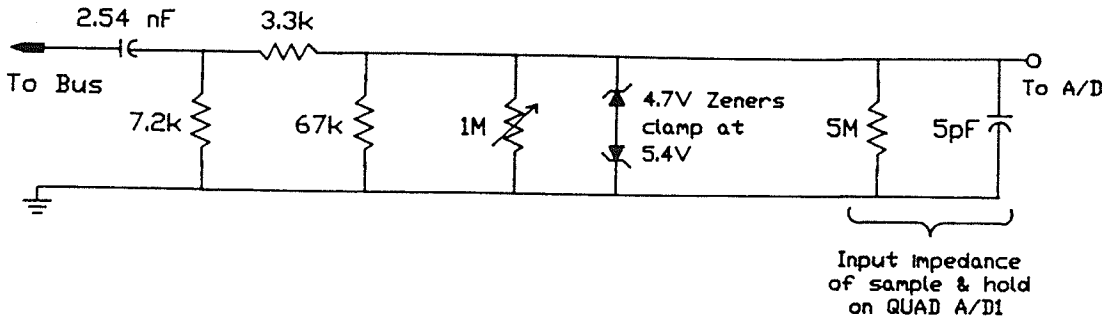


Figure 1.2: Schematic of Coupling Circuit.

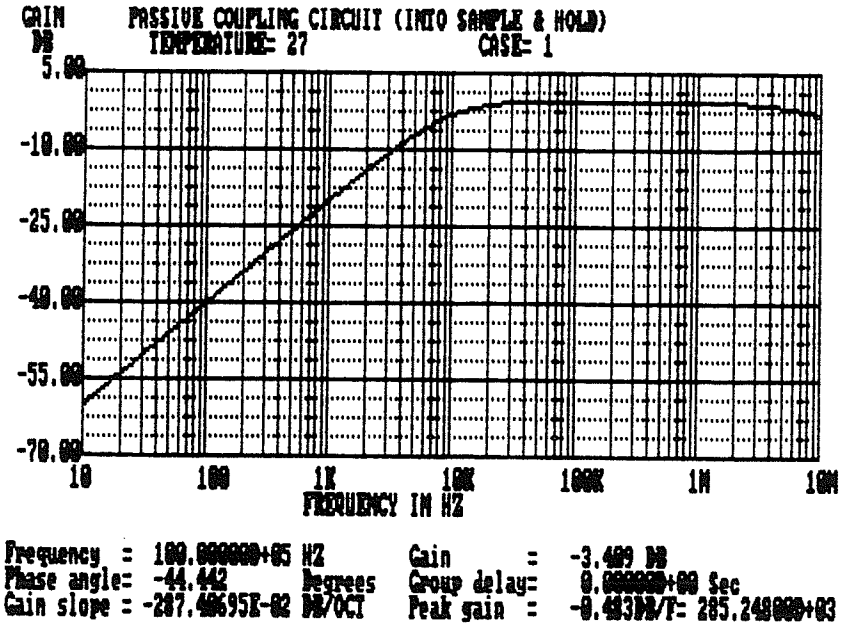


Figure 1.3: Amplitude Bode Plot of Coupling Circuit.

Once done, the program enters a wait mode where it continually polls the hardware timer until it determines that one hour has elapsed. At this time, another set of data is taken, the numerical suffix is updated, appended to the filename, and the file is saved. This procedure is repeated endlessly until stopped by the user.

CHK4FLT.EXE is used in the staged fault measurements. Here, the 4th A/D channel is used to sense the voltage on one phase through a voltage divider. Once any voltage exceeding a preset threshold is detected it means that both breakers have been closed and sampling can be initiated. This program takes 6630 samples at 198.9kHz and saves the data to disk using the name supplied by the user.

Since these three programs are so similar, only the listing for TIME_SAM.EXE is shown in Appendix A.

The final stage in the measurement process is the analysis of the data using Fourier transforms. Since this is a very time-consuming job, especially when the number of samples is large, the University's Amdahl mainframe is best suited to this task. There are many algorithms available which perform discrete Fourier transforms, each of which has advantages and disadvantages:

(1) Classical Discrete Fourier Transform (DFT) Method

This method is directly derived from the continuous Fourier transform equations [1].

The DFT , $X(k) = A(k) + jB(k)$

where, for N samples

$$A(k) = \frac{\left[\sum_{n=0}^{N-1} X(n)\cos(Qnk) + Y(n)\sin(Qnk) \right]}{N}$$

$$B(k) = \frac{\left[\sum_{n=0}^{N-1} Y(n)\cos(Qnk) - X(n)\sin(Qnk) \right]}{N}$$

where: $Q = \frac{2\pi}{N}$

$X(n)$ = real part of discrete signal

$Y(n)$ = imaginary part of discrete signal

k takes on values from 0 to N-1

Note: Physical signals have no imaginary part so the above simplify to:

$$A(k) = \frac{\left[\sum_{n=0}^{N-1} X(n)\cos(Qnk) \right]}{N}$$

$$B(k) = \frac{\left[\sum_{n=0}^{N-1} -X(n)\sin(Qnk) \right]}{N}$$

Fourier coefficients are given by:

$$C(k) = \left| A(k) + jB(k) \right|$$

This yields a two sided spectrum with terms for both positive and negative frequency. However, for this application, negative frequency is essentially the same thing as positive frequency. So, to simplify things, only the positive frequencies are shown. The Fourier coefficients have amplitudes that are the sums of their positive and negative components [2].

The fundamental drawback to the classical DFT is that it is so computationally intensive. For N samples, this algorithm requires $\frac{1}{2}N^2$ major operations (for example, a multiplication).

(2) Cooley Tukey FFT Algorithm

This is the method applied by many commercial FFT programs. It works by uncoupling row and column calculations, thus converting a single difficult problem into multiple easy ones [2]. This results in considerably less computation time than the classical DFT algorithm. For N samples there are only $\frac{1}{2}N log_2 N = \frac{1}{2}N \frac{log_{10} N}{log_{10} 2}$ (assuming a radix of 2) major operations required.

The problem with the FFT is that the allowable number of samples is intimately related to the radix of the FFT algorithm.

$N = R^M$

R = radix
N = number of samples
M = any integer

so, assuming a radix of 2 , $N = 2^M$

Since it is desirable to sample as fast as possible with the QUAD A/D1 AND take an integer number of ac cycles, this restriction on the number of samples is a problem.

(3) Goertzel's 2nd Order DFT Algorithm

Goertzel's algorithm [1] is a happy medium between the FFT and the classical DFT. This method requires only half as many real multiplications as the classical method ($\frac{1}{4}N^2$), and uses 2N trigonometric operations instead of $2N^2$. However, this algorithm still retains the advantage of the classical method

because it allows any even number of samples to be used.

A comparison of the previous three methods for a 4 096 point transform yields the following:

Table 2.1

Algorithm	Time (s)
Classical DFT	67.29
Cooley - Tukey	0.76
Goertzel's	12.42

The preceding table justifies the claim that Goertzel's algorithm is considerably faster than the classical method. Although this computational advantage is somewhat significant for 4 096 points of data, it takes on an even greater importance when the number of points exceeds 10 000. For this reason, and the fact that varying numbers of samples are wanted, Goertzel's method was chosen for analysis of all the sampled data. A listing of the FORTRAN program used is provided in the Appendix B.

The numerical data taken during normal and fault conditions is uploaded to the Amdahl via a 1200 baud telephone modem using Kermit[2]. This data is then processed and plotted on the Amdahl's high speed laser printers.

[2]Kermit is a telecommunications package which facilitates file transfers between 7 bit and 8 bit computer systems.

2.3 Results of Measurements

The next two sections describe the methodology of the electrical noise measurements and present some selected results. These results are explained and their impact on the carrier current ZSIP system is discussed.

2.3.1 Normal Conditions

Electrical noise measurements under normal conditions were conducted at Federal Pioneer's Rockman Ave. transformer plant in Winnipeg between May 11th and 18th, 1988. Measurements were made on three different electrical supplies at many different times throughout the day. These supplies were a 208V, 3-phase supply driving a variety of loads, a 600V supply near some seam welding machines, and finally, a 600V supply on a quiet side of the plant.

To make the above measurements, the sampling equipment was connected to one phase of the supply through the passive coupling circuit which was previously described. Since the sampling software automated the measurement process, the equipment could be left unattended for 24 hours at a time. This resulted in measurements being taken for a wide variety of load conditions through the day and night. This procedure was followed for the first two supplies but, due to time constraints, only three measurements could be made on the third supply.

Figure 2.1 shows a typical voltage waveform found on the 208V bus. This particular signal was recorded at 3:20 p.m. and certainly appears to suffer from a great deal of noise and distortion. However, this figure is, in some ways, deceptive. Since this waveform is viewed by the sampling hardware through a high-pass filter, the high frequency noise component appears very large in comparison with the 60Hz fundamental. If viewed on an oscilloscope which was directly coupled to the bus, there would not be 45 dB of attenuation at 60Hz as there is here, and therefore, the waveform would look more sinusoidal. Figure 2.2 shows the discrete spectrum obtained by performing a Fourier transform on the signal in Fig. 2.1. As can be clearly seen, there is some noise present at low frequencies (<4 kHz) but practically none up in the PLC (Power Line Carrier) frequency range (60kHz to 100kHz). Other measurements at different times reveal very similar results.

One observation worth noting with all of the spectra presented is the small frequency gap near zero Hz. This is created intentionally since these coefficients are large and tend to make everything else look comparatively tiny. Since these low frequencies are of no interest here anyway, they are set to zero before the plots are generated.

Figure 2.3 is a plot of the waveform found on the "noisy" 600V bus at 3:00 a.m. Observing this signal, it is quite obvious that this bus is subjected to more noise than is the 208V bus. When Fourier transforms are performed, there is

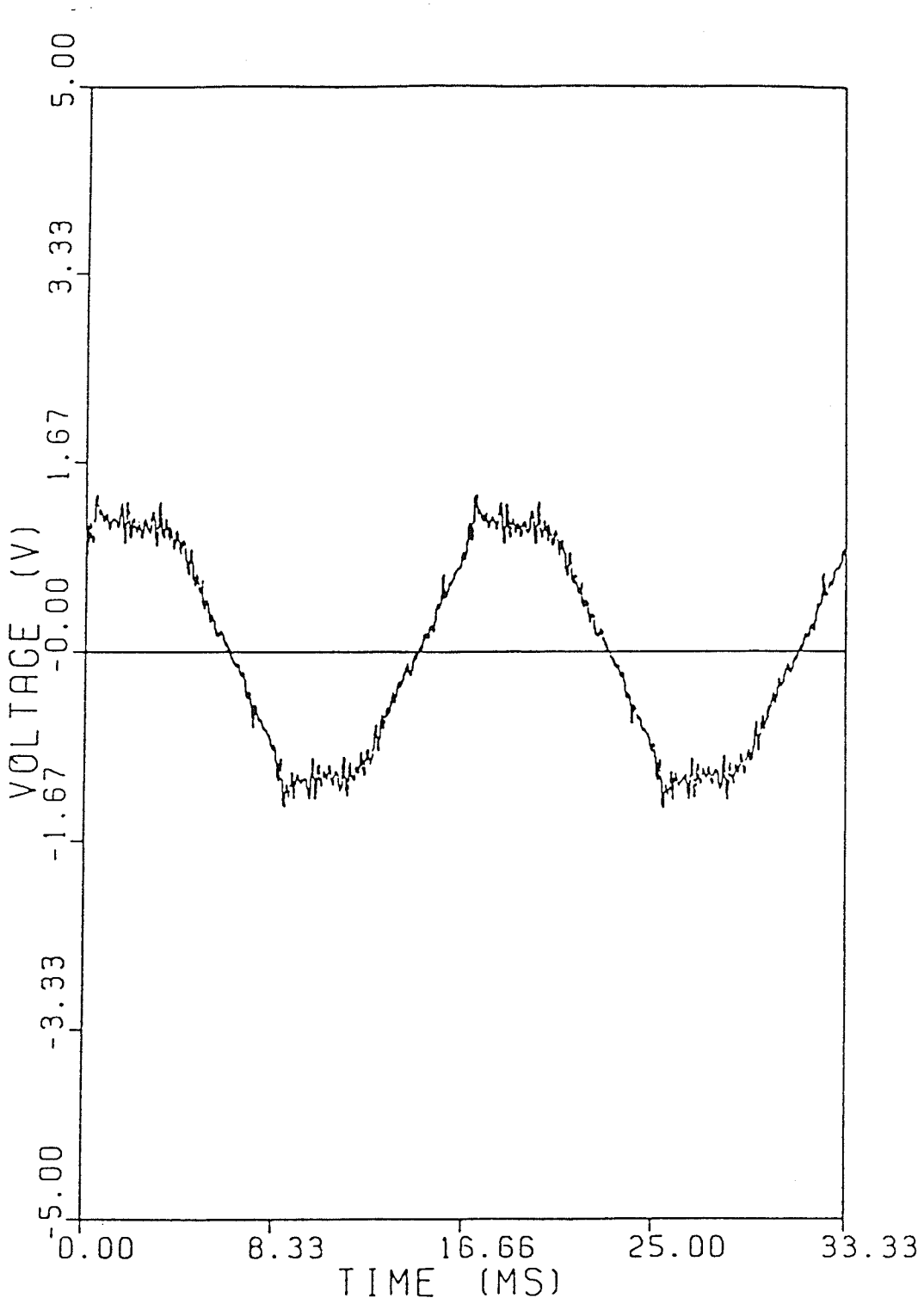


Figure 2.1: Time Domain Recording on 208V Bus @ 15:20.

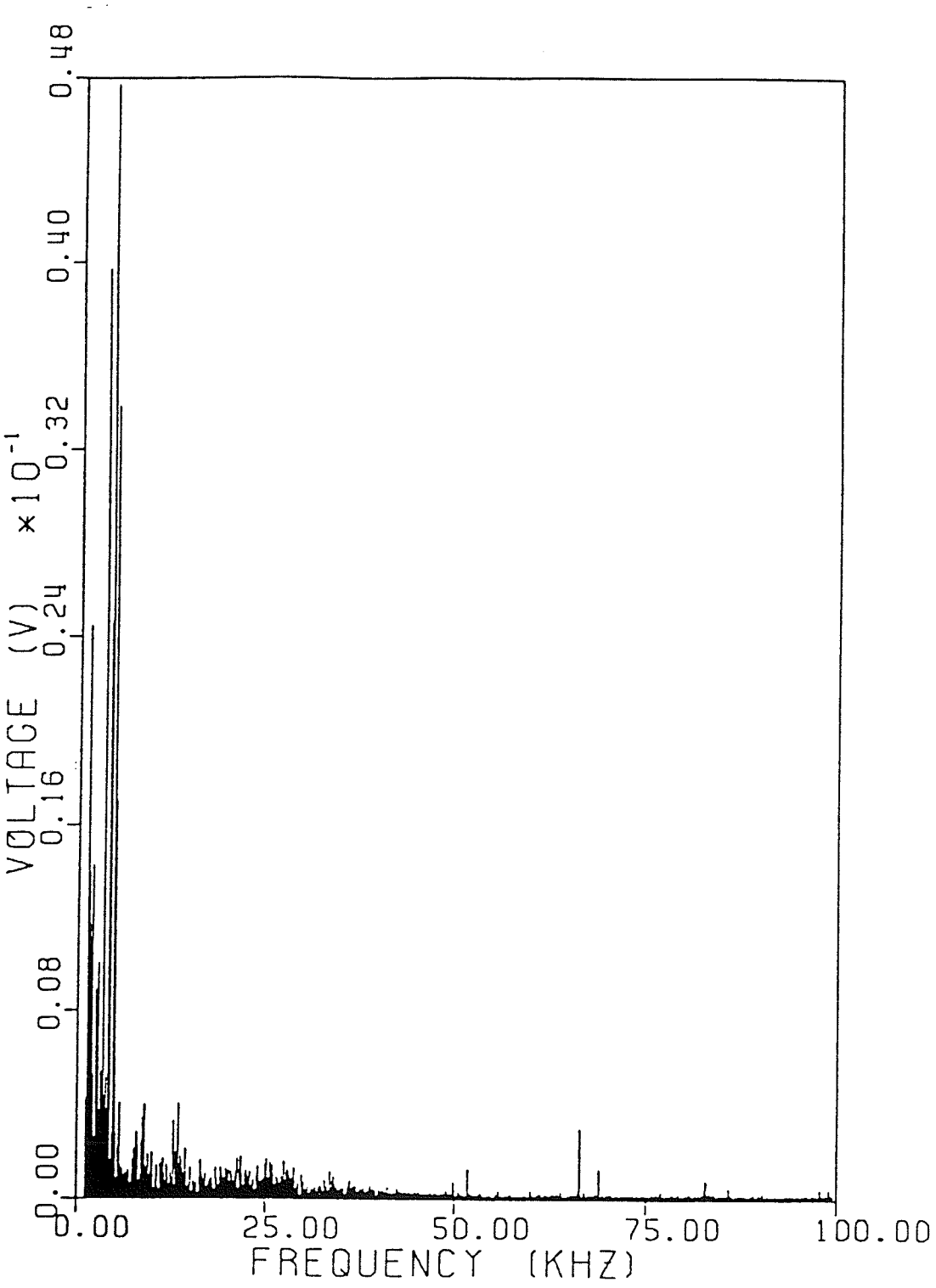


Figure 2.2: Fourier Spectrum of Waveform in Figure 2.1.

a very interesting result (see Fig. 2.4). Here, there is moderate noise below about 5kHz. However, upwards of 5kHz, the noise is almost insignificant with the exception of two huge noise spikes which appear to be centered at about 70kHz. On a later visit to FPL, we were able to determine with the aid of an oscilloscope that these signals were actually present at 130kHz rather than 70kHz. The 99.45kHz Nyquist frequency allowed these signals to be aliased since there was no anti-aliasing filter included in the coupling circuit. This oversight was later corrected, before the fault measurements were taken.

The source of these two "mystery signals" has never been identified. Since the frequency spacing between them is almost exactly 1kHz, it seems unlikely that they would result from some natural phenomenon. Below are several possible causes and a judgment on their likelihood:

1) Some resonant condition on FPL's busses - not likely since the signals are present on busses on opposite sides of the plant.

2) Some travelling wave phenomenon - no, since the wavelength is too long. The frequency of travelling waves is determined only by the length of the bus.

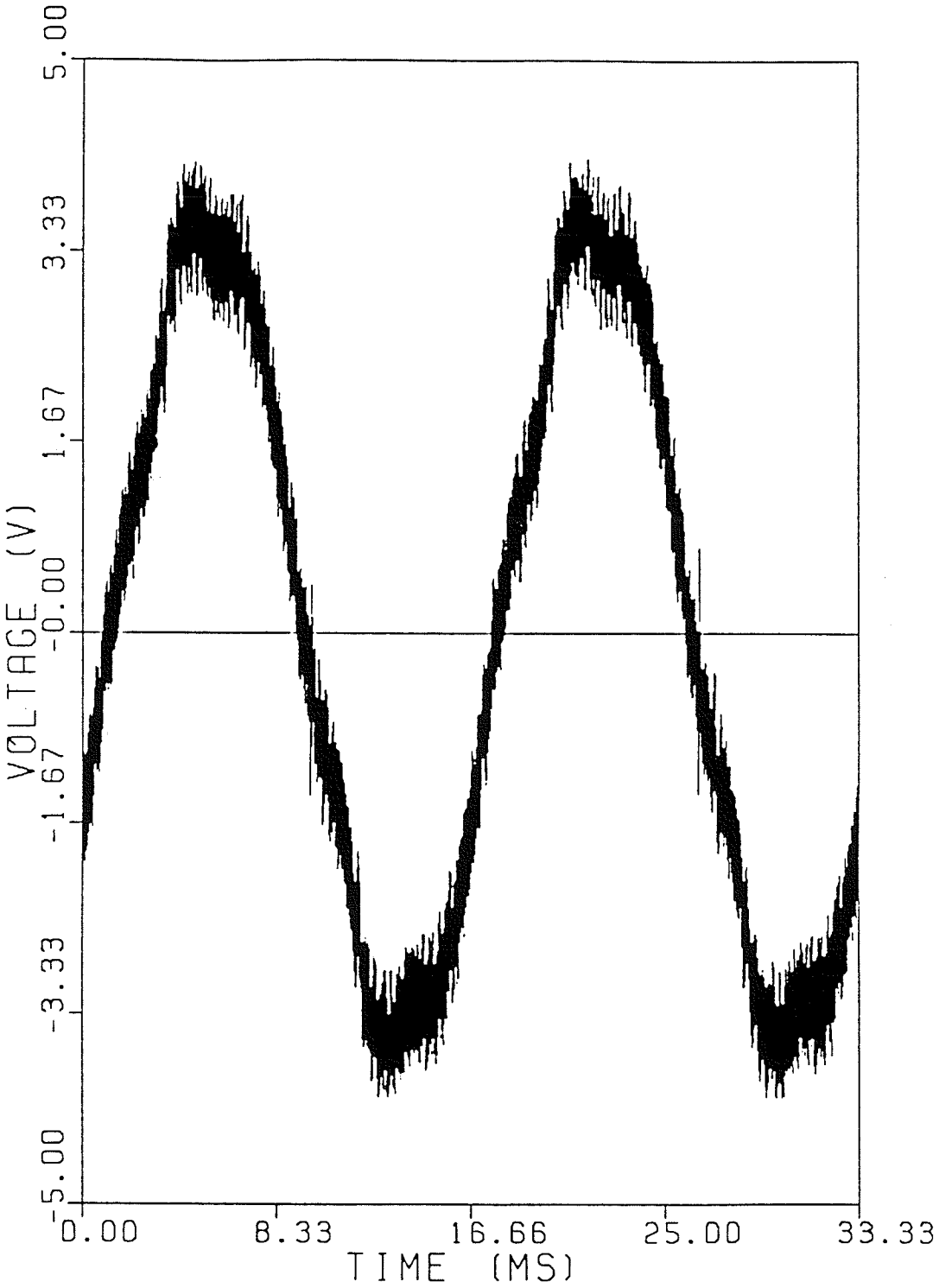


Figure 2.3: Time Domain Recording on 600V Bus @ 03:00.

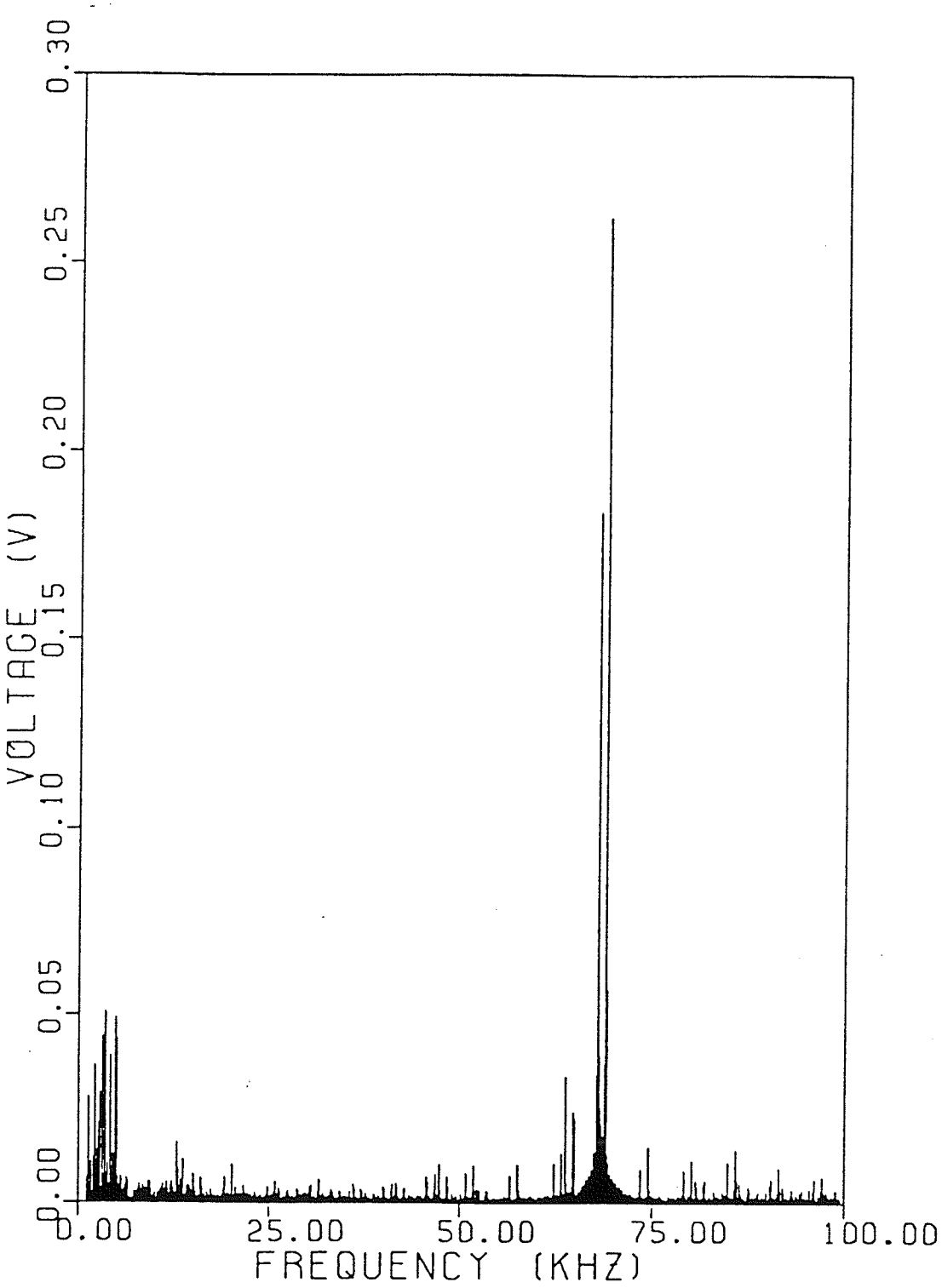


Figure 2.4: Fourier Spectrum of Waveform in Figure 2.3.

3) An air induced signal - might be possible. Apparently, some navigational beacons use extremely low frequencies like this, although it is unlikely that the transmitted power would be high enough to induce a voltage of 0.26V in FPL's busses.

4) Some carrier signal injected by Manitoba Hydro - Maybe, Hydro telecontrol confirms that a PLC channel from Dorsey to Brandon is operated in the 130kHz to 140kHz range. However, they claim that line traps prevent this signal from getting back into the distribution. In addition, this signal would have to pass through a large step-down transformer to get into FPL's plant. This would almost certainly attenuate the signal to a level far below 0.26V.

5) Some carrier current signal from one of FPL's industrial neighbors - impossible since there are only residential customers on FPL's feed.

6) A signal originating from inside FPL itself - most likely cause. Although FPL does have an energy management system, it uses lower frequencies ($\simeq$ 3kHz I think). The 1kHz frequency spacing suggests that the signals are due to some form of FSK communication. However, the exact source remains a mystery.

Since these "mystery signals" are far above the frequency range under investigation (60 - 100kHz), they are not of any real concern.

Proceeding to the next time domain measurement (Fig. 2.5), it appears that there is even more noise present on the 600V bus at 11:00 a.m. than at 3:00 a.m. This is expected since there is more equipment operating at this time. Figure 2.6 shows the corresponding Fourier spectrum for this waveform. Here again, the two "mystery signals" are present, although with smaller amplitudes. This is possibly due to capacitive loads on-line at this time which short some of the energy to ground. Also present is a large noise component below 5kHz and some seemingly random noise between about 10kHz and 60kHz. However, the PLC frequency range, which is of the most interest, seems basically noise free.

The next measurement (Fig. 2.7) was made on the other, quieter, 600V bus at 3:10 p.m. Notice that the low frequency noise component is somewhat smaller than on the other 600V bus. In addition, there is no appreciable noise component found in the mid-frequency band. The two mystery signals are present although with reduced amplitudes. Most importantly, the PLC frequency range is essentially free of noise which means that a carrier current system should not be adversely affected by steady state noise.

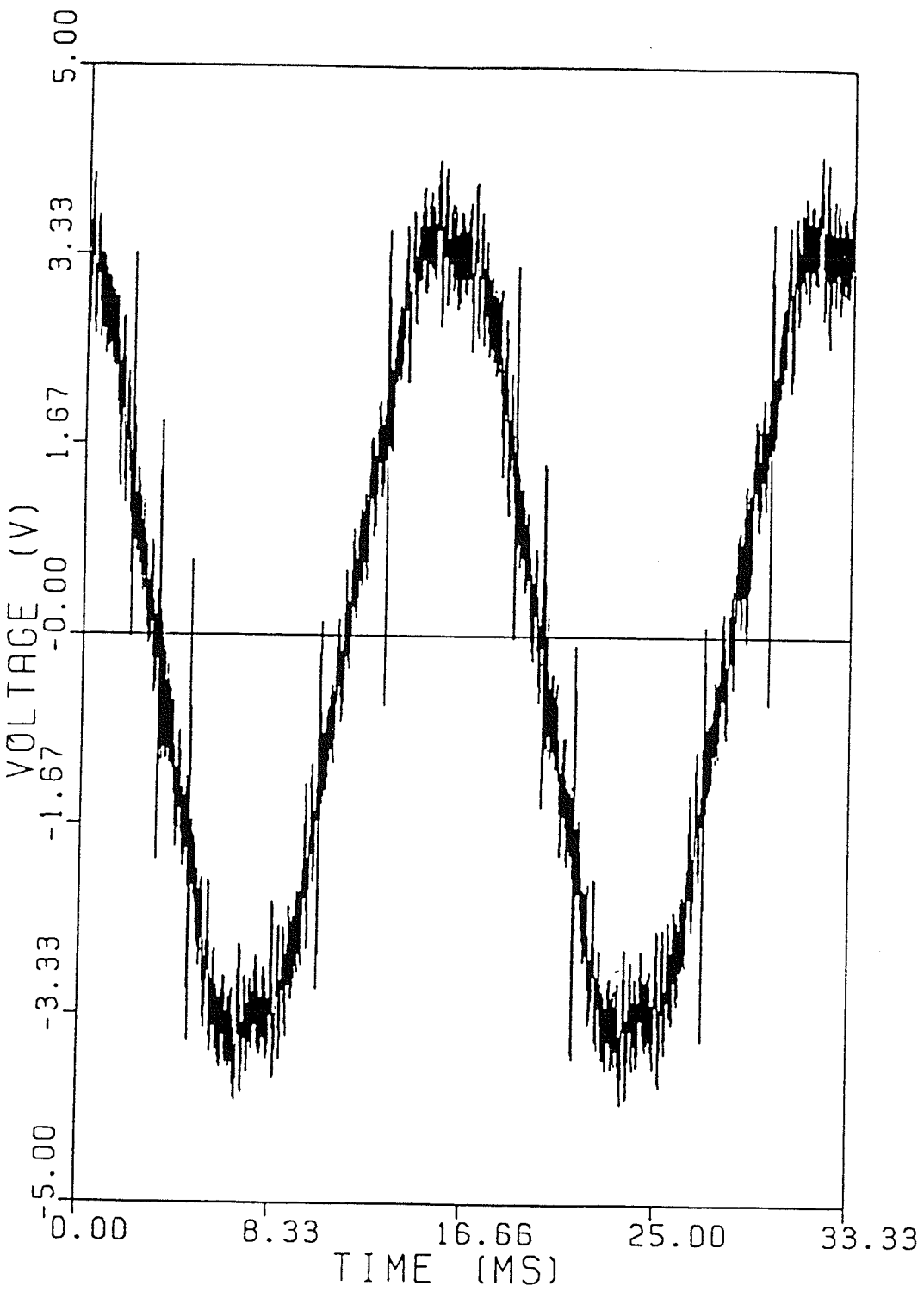


Figure 2.5: Time Domain Recording on 600V Bus @ 11:00.

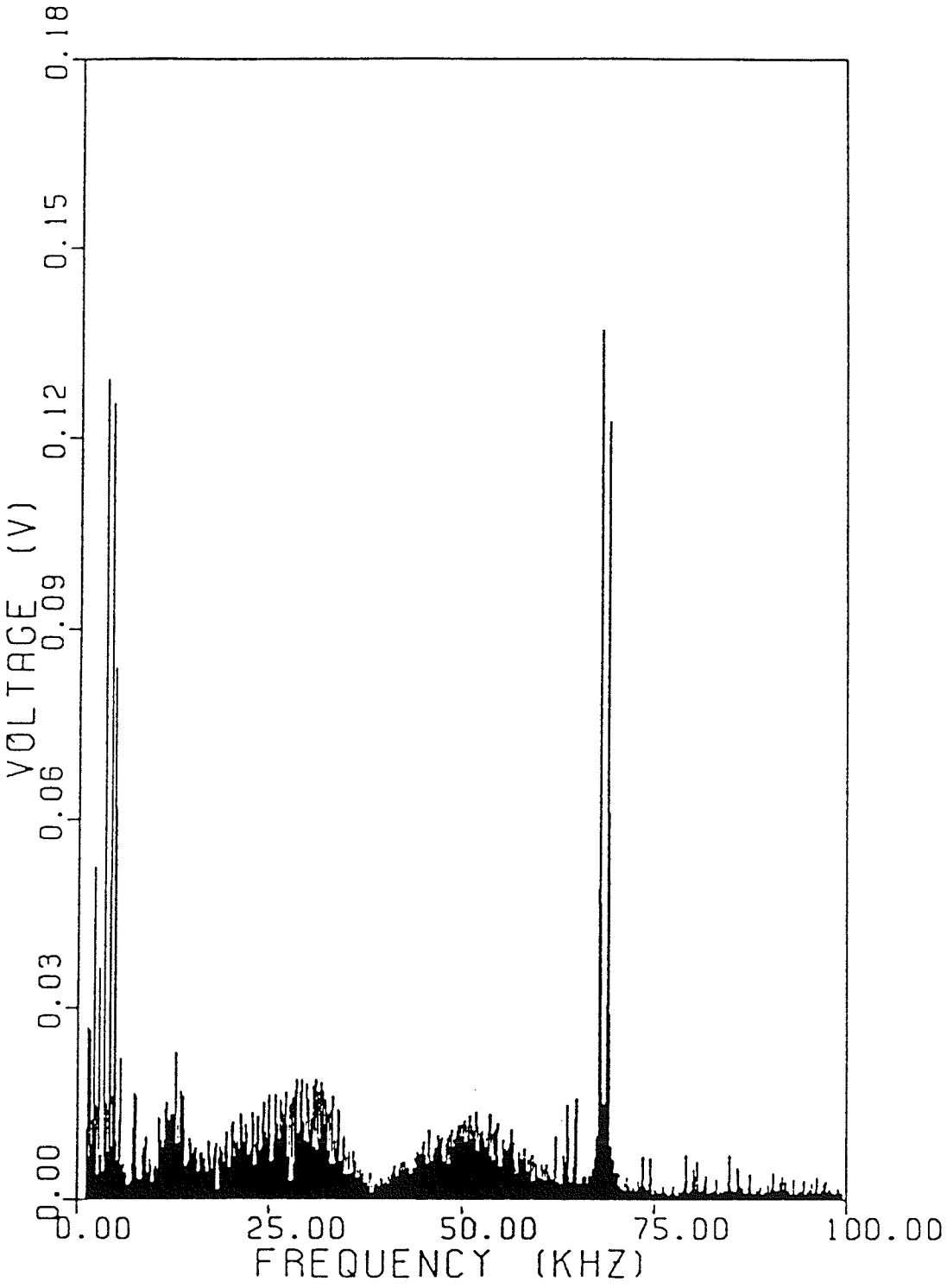


Figure 2.6: Fourier Spectrum of Waveform in Figure 2.5.

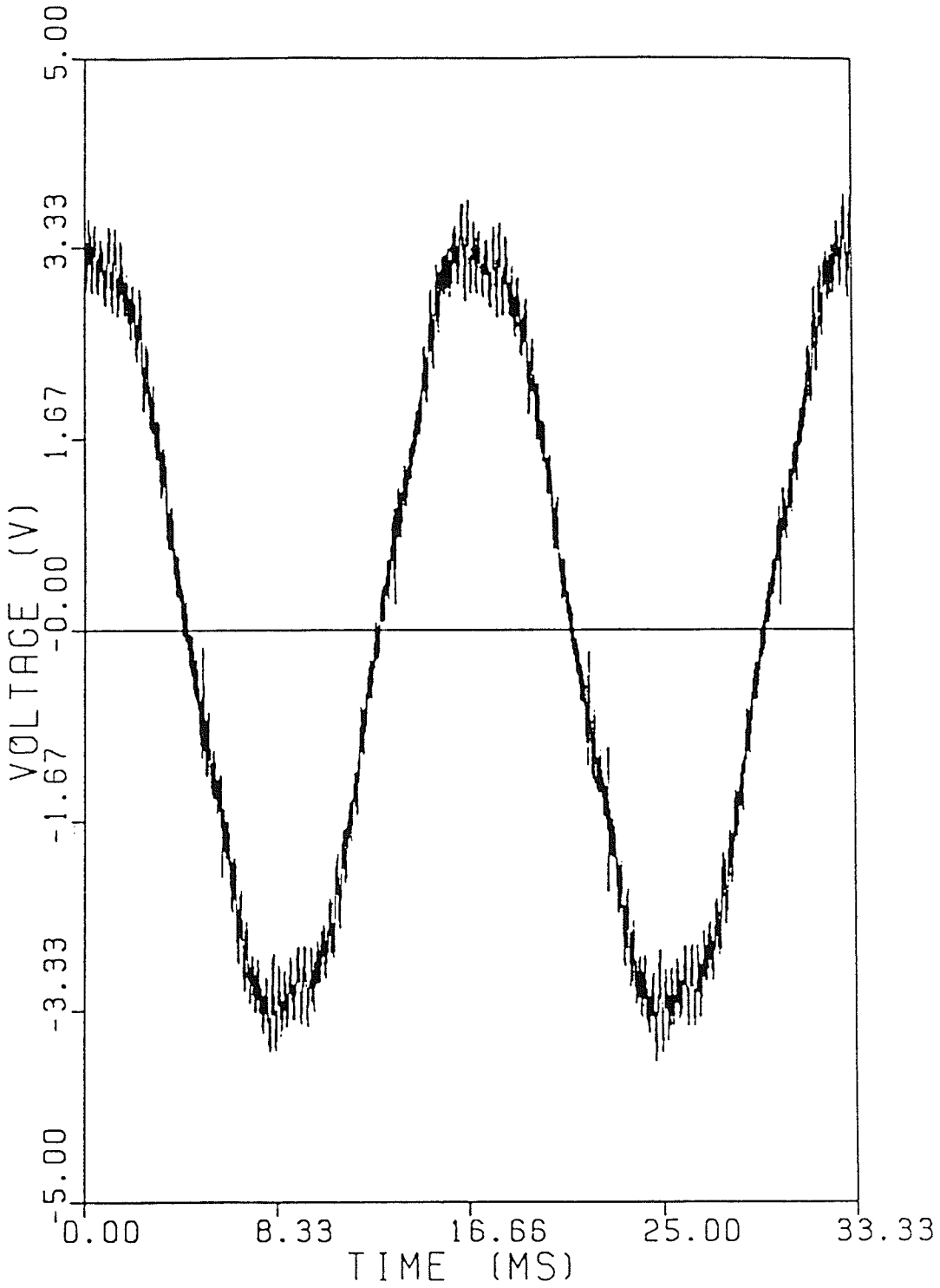


Figure 2.7: Time Domain Recording on "Quiet" 600V Bus @ 15:10.

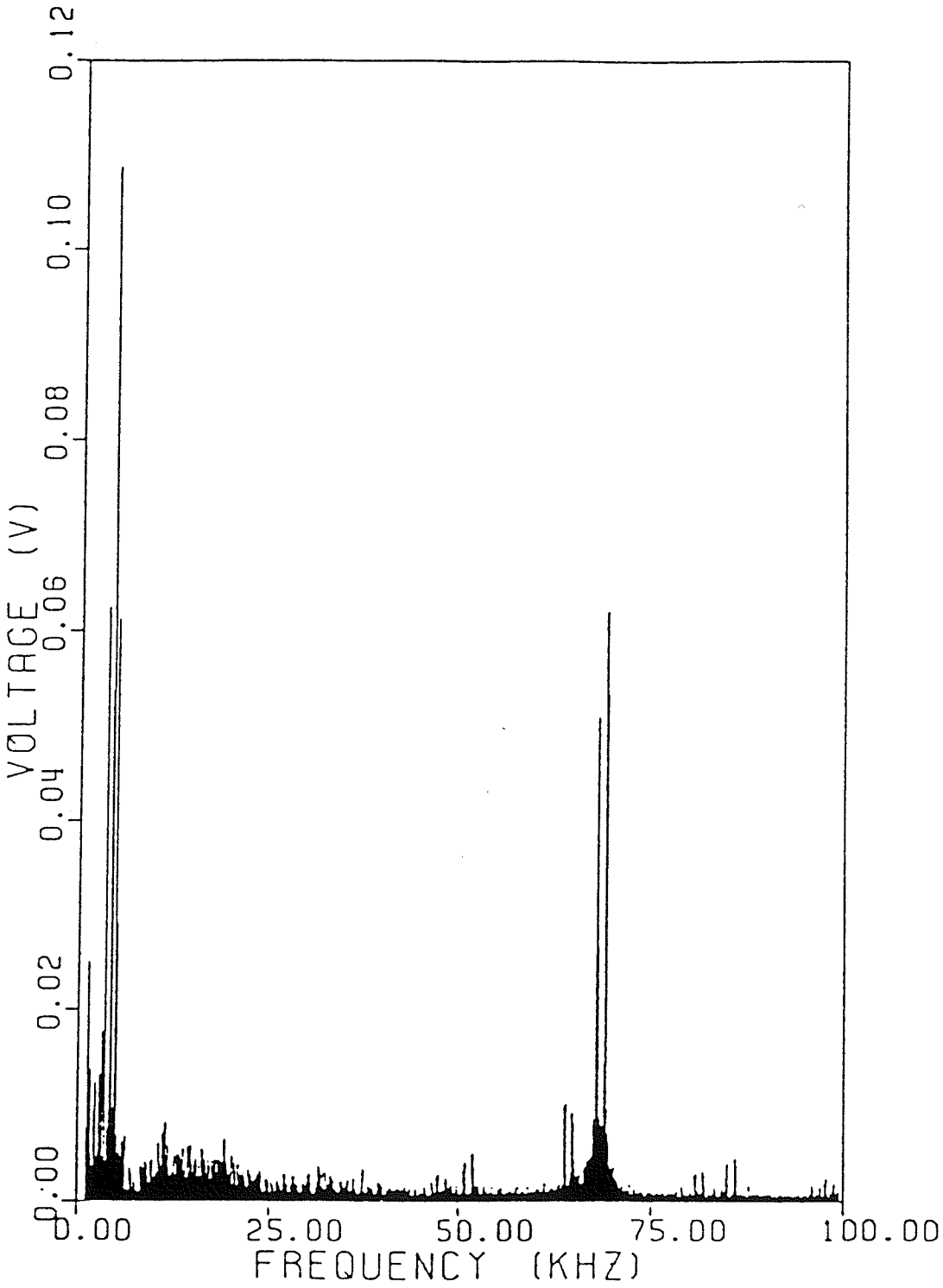


Figure 2.8: Fourier Spectrum of Waveform in Figure 2.7.

2.3.2 Fault Conditions

Electrical noise measurements under fault conditions were conducted in the University of Manitoba's machines lab between August 15th and 19th, 1988. Figure 2.9 is a simplified diagram which shows how the staged faults were conducted. Essentially, phases 'A' and 'B' of a 208V supply are shorted together through an arc gap, using four 7.5kW resistance boxes in parallel to limit the current. By properly controlling the current, a large amount of noise can be generated from the arcing fault while making sure that the circuit breaker does not operate. In the arc gap, a thin piece of nickel wire is used to connect phases 'A' and 'B'. When the breaker is closed, this wire evaporates almost instantly and provides the ions necessary to establish an arcing fault.

The measurement software (CHK4FLT.EXE) continuously polls A/D channel 4, looking for any significant deviation from zero volts on phase 'C'. If this occurs, it indicates to the program that the breaker has been closed and so sampling should be started. In Fig. 2.9, sampling is shown on phase 'A' only. Although this was true for most tests, some measured the noise on phase 'C', the unfaulted phase.

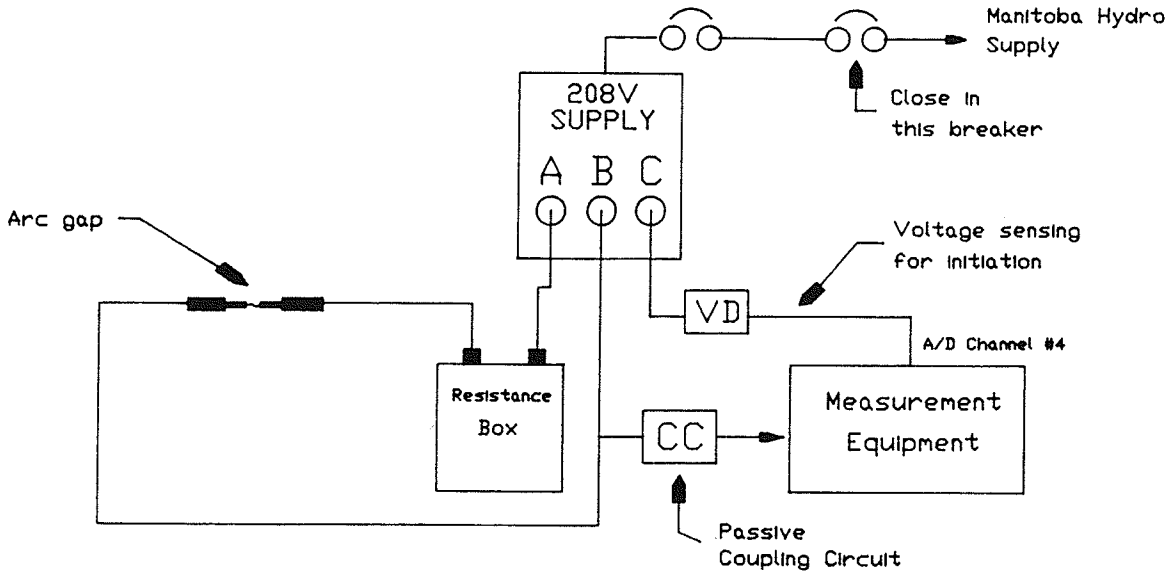


Figure 2.9: Staged Fault Equipment.

In Fig. 2.10, two cycles of data are shown immediately after an arcing fault is initiated on phases 'A' and 'B'. Here, the noise is worst immediately following the fault initiation as the nickel wire vaporizes and the arc becomes established. As time passes, the arc seems to quench itself and the waveform essentially returns to normal after about three ac cycles. Figure 2.11 shows the Fourier spectrum for this waveform. The electrical noise is very high at low frequencies (< 5kHz) but seems to decrease exponentially with frequency. In the PLC frequency range, the noise seems to be stable at about 5mV; far less than the 40mV needed to lock in a tone decoder.

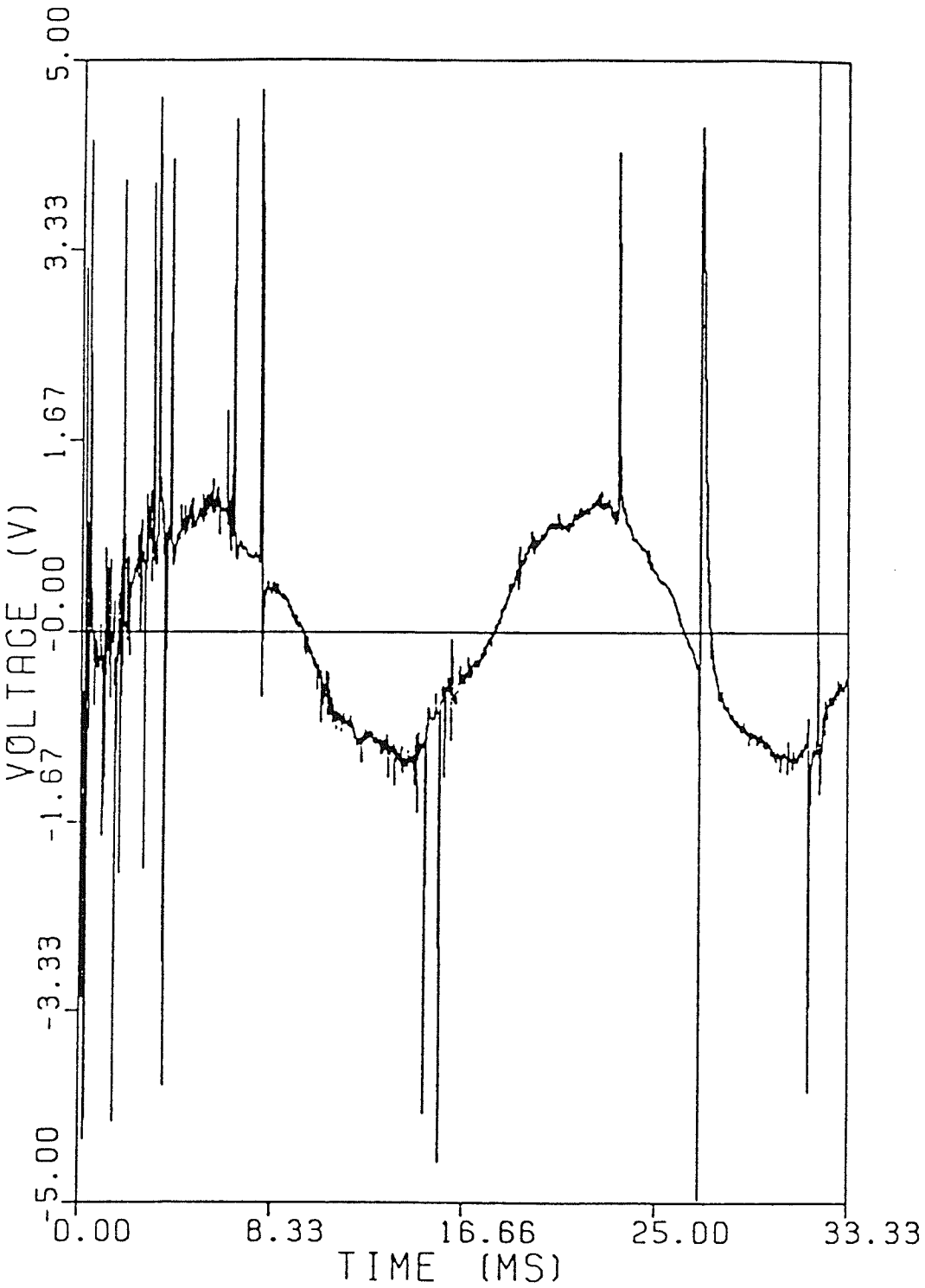


Figure 2.10: Fault Recording - Sampling on Phase 'A'.

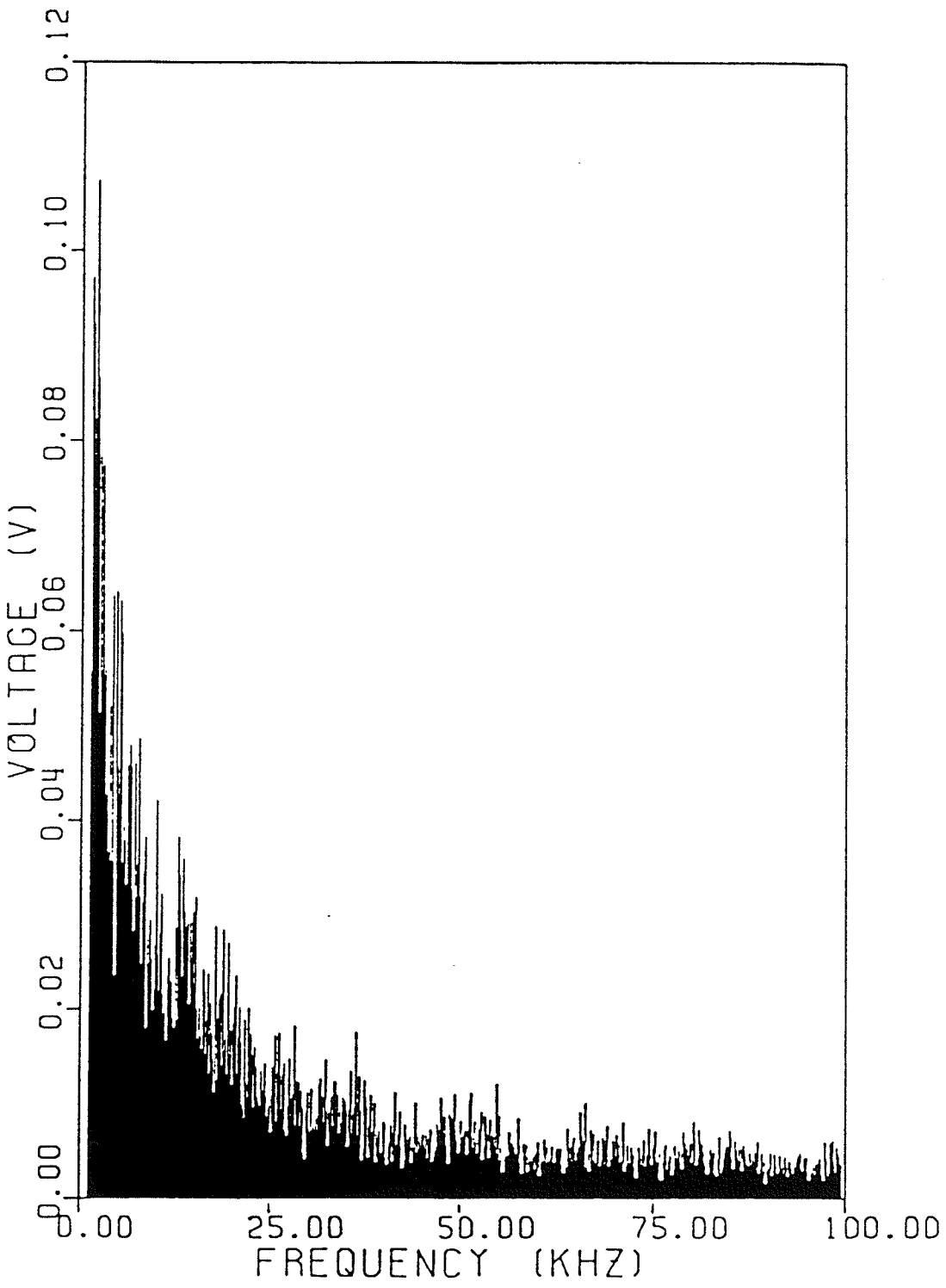


Figure 2.11: Fourier Spectrum of Waveform in Figure 2.10.

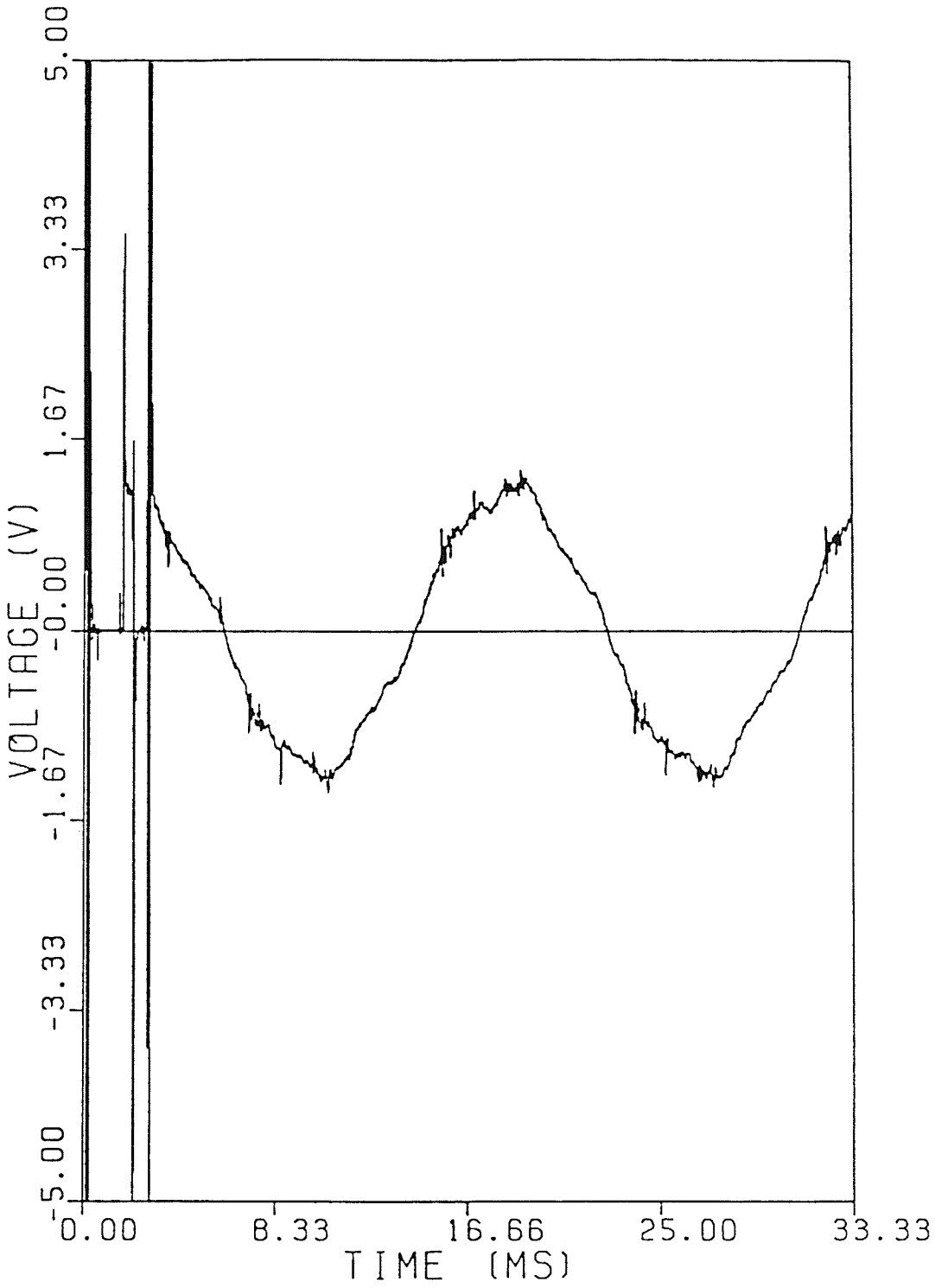


Figure 2.12: Fault Recording - Sampling on Phase 'C'.

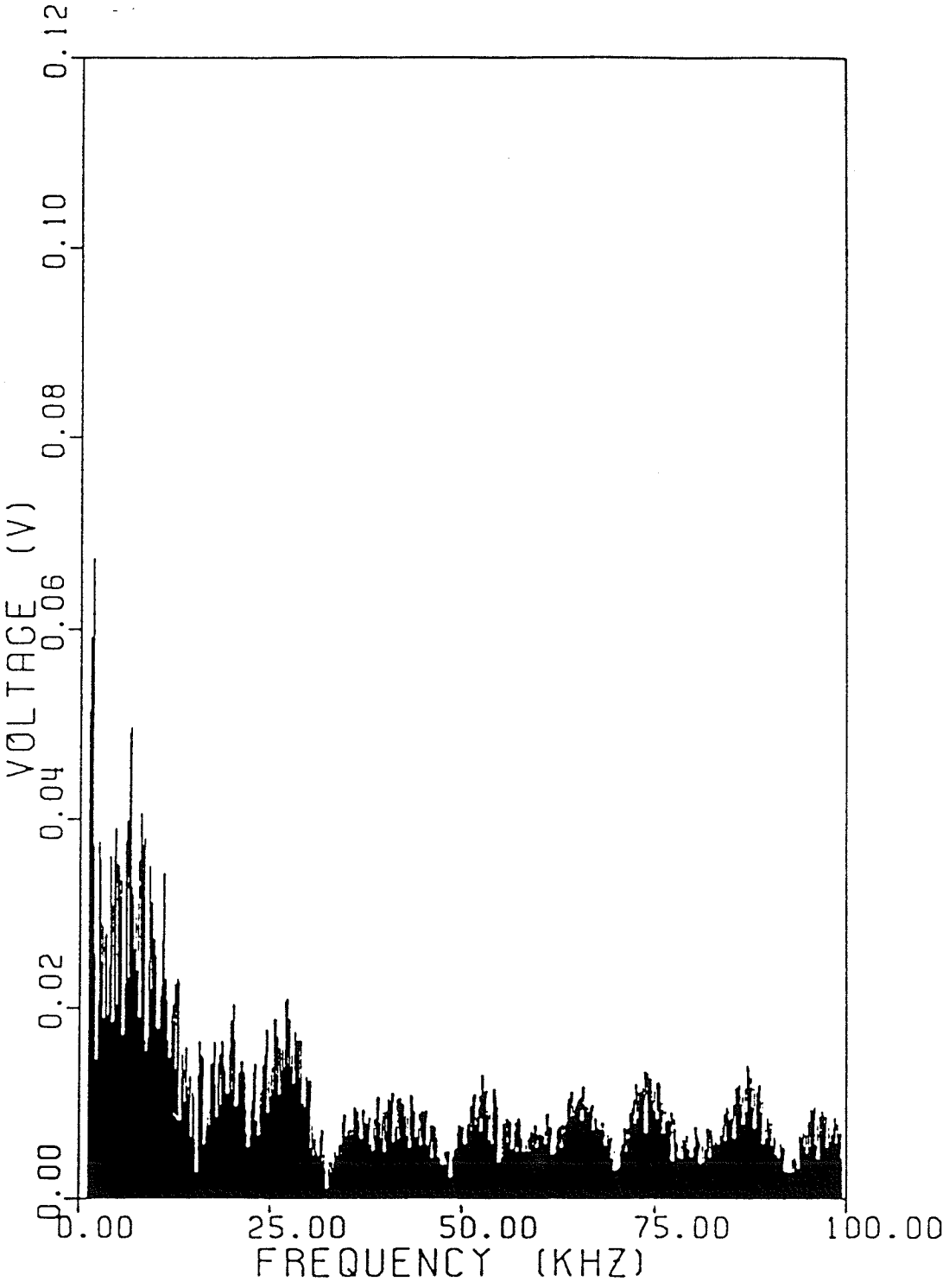


Figure 2.13: Fourier Spectrum of Waveform in Figure 2.11.

The next fault recording (Fig. 2.12) is for a situation very similar to the one above except that the sampling is performed on phase 'C' instead of phase 'A'. Since phase 'C' is unfaulted, almost all the noise found here will have been coupled in from phases 'A' and 'B'. The time domain recording shows a waveform which is extremely distorted during the first few milliseconds of the fault's existence. The distortion disappears very quickly and the waveform returns to normal. Fourier transforms reveal that there is somewhat less noise here than in the previous example, although the noise in the PLC range is still about 5 to 10mV.

Figure 2.14 shows the waveform recorded when phase 'A' is shorted to ground rather than to phase 'B'. The time domain plot reveals nothing out of the ordinary; the noise is severe at first but dies to nothing as the arc quenches itself. The frequency domain plot in Fig. 2.15 reveals a noise spectrum very similar to the ones found for phase to phase faults. Again, the noise in the PLC range is negligible.

The next recording (see Fig. 2.16) shows the "worst case" noise for the staged fault measurements. Here, the resistance boxes are short circuited to see the results when the fault current is as high as possible. The electrical noise is quite severe at first as the wire evaporates, and even worse later when the circuit breaker operates. Figure 2.17 shows the noise spectrum for this measurement and, as expected, it is a real mess. As usual, the low frequency noise

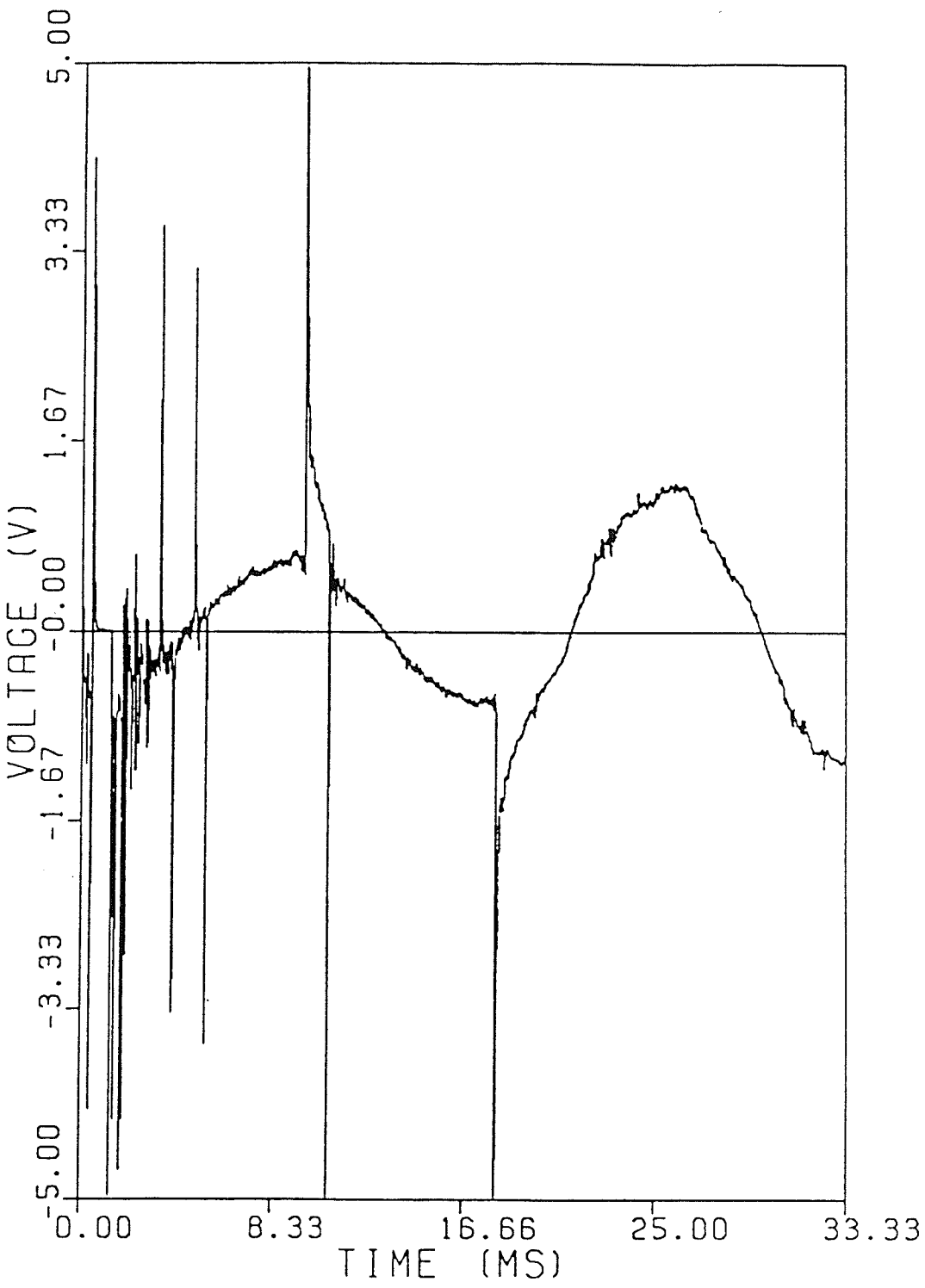


Figure 2.14: Fault Recording - Phase 'A' to Ground.

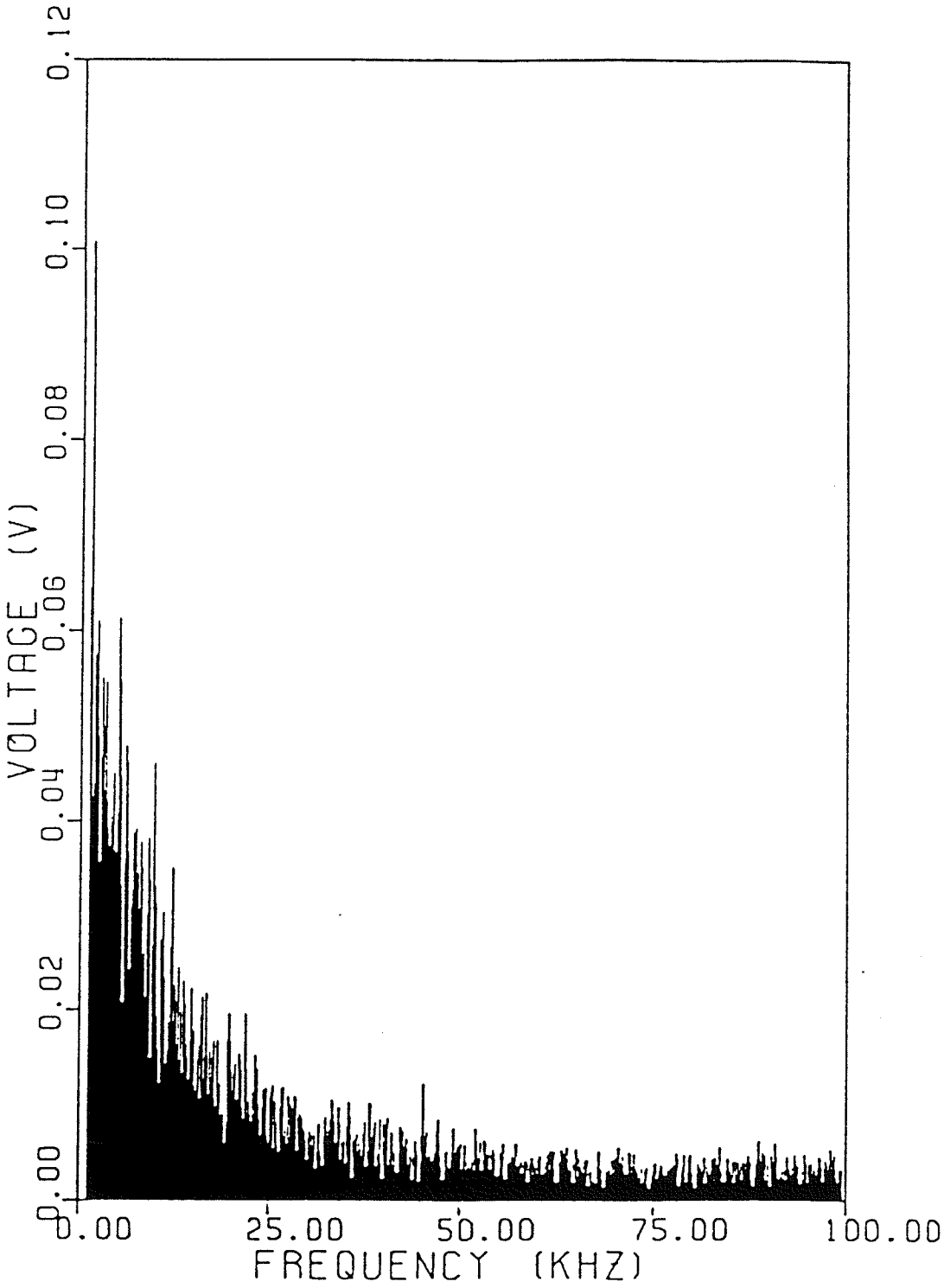


Figure 2.15: Fourier Spectrum of Waveform in Figure 2.14.

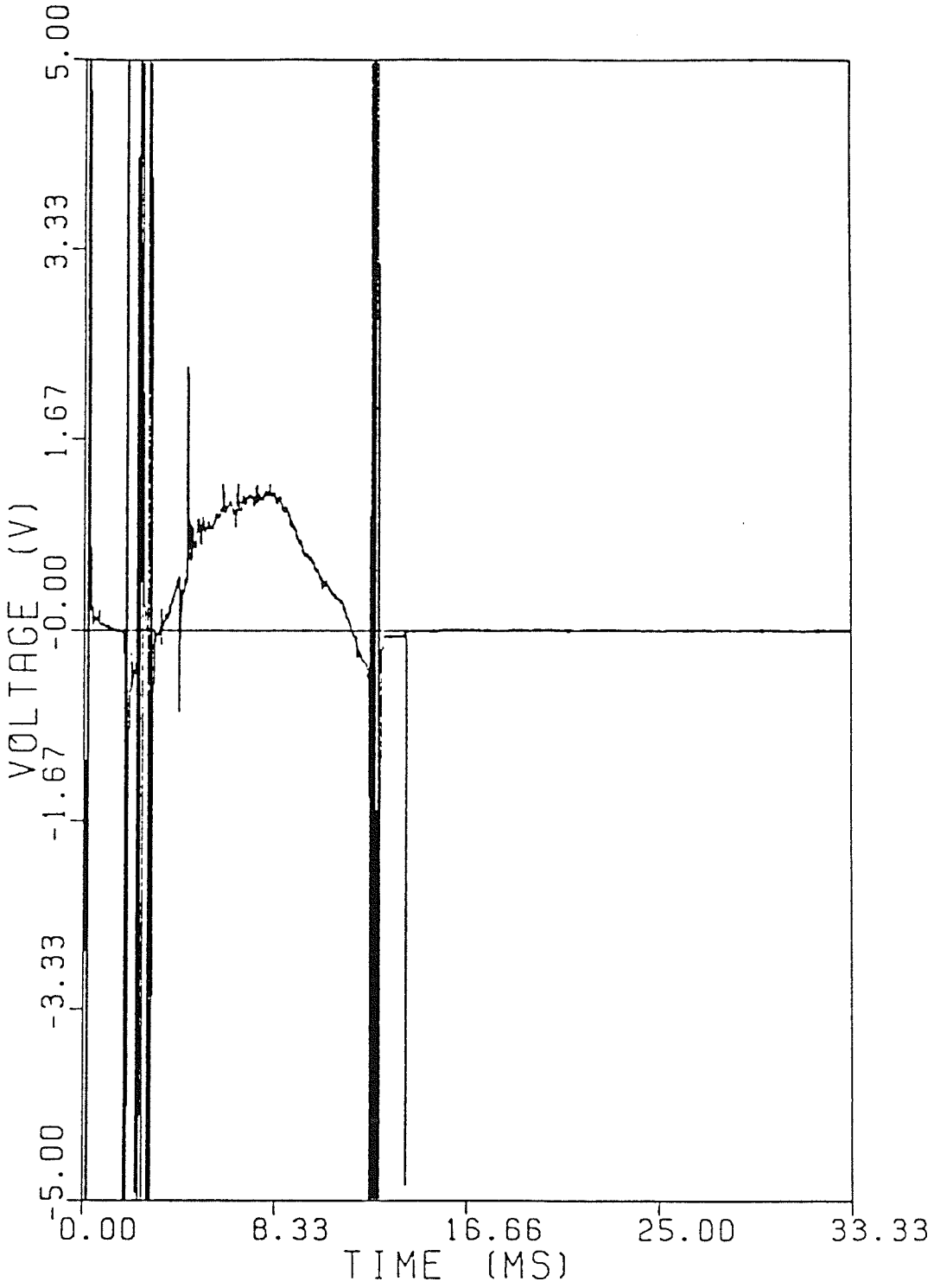


Figure 2.16: Fault Recording - Worst Case Noise.

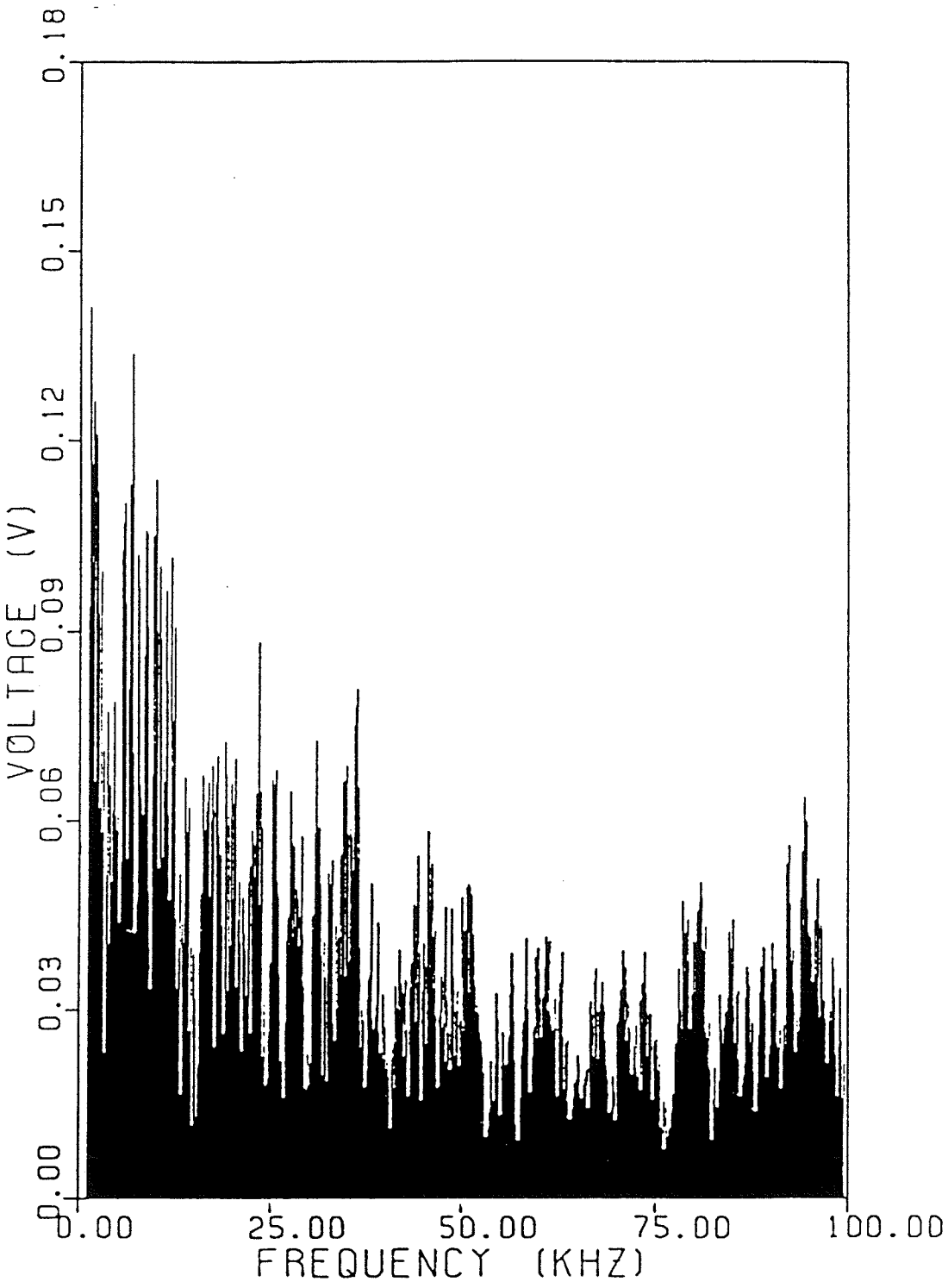


Figure 2.17: Fourier Spectrum of Waveform in Figure 2.16.

is the worst, but, in this case, there is also a lot of noise spread throughout the rest of the spectrum. Even up in the PLC range, the voltage varies from 10 to 65mV. It would be very difficult to make a carrier current system operate reliably under these conditions. Luckily, it turns out that most of the noise shown here is due to the operation of the breaker and not from the electrical arc itself. Since the blocking signals used in the ZSIP system are meant (by definition) to be sent before the breakers operate anyway, this condition does not pose any real problem for the carrier ZSIP system.

In concluding this chapter, it should be noted that a number of electrical noise measurements were taken to get an idea of the kinds of noise expected under normal and fault conditions. These measurements encompassed a variety of load conditions and fault scenarios which are a good representation of conditions found in the real world. None revealed enough noise in the PLC frequency range to jeopardize the operation of a carrier current system utilizing frequencies between 60 and 100kHz. Using the knowledge gained in this chapter, the design of the carrier ZSIP unit's hardware and software can now be started.

Chapter III

HARDWARE/SOFTWARE DESIGN

3.1 Design Philosophies

In designing the hardware and software for the carrier ZSIP system there are a few design philosophies to keep in mind:

1) Each unit should be universal. Ideally, there should be a "generic box" which can be taken off the shelf and installed at any point in the distribution system. This means that every unit must contain hardware for both transmitting and receiving of signals. In addition, there must be some simple method for the user to configure the unit to operate on a particular distribution level.

2) The system must be both reliable and secure. In protection, reliability means that a device WILL operate when it is supposed to. Security means it WON'T operate when it isn't supposed to. One hundred percent reliability and security is impossible to achieve, but, it is reasonable to expect that a

blocking signal will be received in less than 1 ac cycle and, that noise on the bus won't cause a false operation.

3) The system must be inexpensive. After all, if the cost of the carrier system is greater than the cost of running wire and conduit, then there is no reason to develop the carrier system in the first place.

3.2 Possible Blocking Signal Schemes

In the following sections, two possible schemes for transmitting blocking signals are discussed. The relative advantages and disadvantages of each are detailed and, from this, a conclusion is reached.

3.2.1 FSK System

Frequency shift keying (FSK) is widely used for low speed communications in devices such as 300 baud telephone modems. FSK works by transmitting one frequency to represent a digital zero, and another frequency to represent digital one (see Fig. 3.1). In an application such as a carrier current system, these two frequencies would be quite high so that they would be subject to as little low frequency electrical noise as possible [3,4]. A frequency spacing of up to 10% of the center frequency is typical.

Figure 3.2 shows how FSK could be applied to the ZSIP system. Communication between adjacent levels of distribution is accomplished by transmitting a particular sequence of bits. For example, the relay on the primary distribution bus receives signals from the two relays on the secondary distribution busses using SEQ1. The relays on the secondary busses receive signals from their corresponding downstream relays using SEQ2. Using this scheme, exactly n-1 unique sequences are required for n levels of distribution.

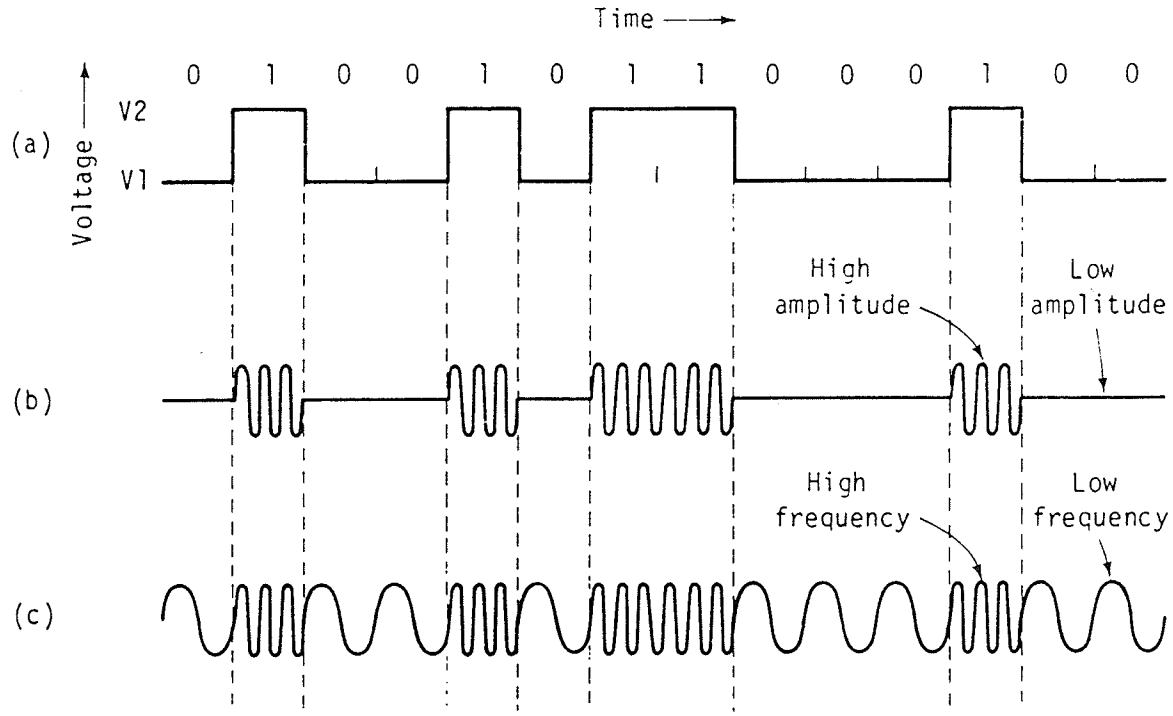


Figure 3.1: Modulation Schemes.
(a) Two level signal. (b) Amplitude modulation.
(c) Frequency modulation (FSK).

Source: Structured Computer Organization, Andrew S. Tannenbaum Prentice Hall Inc., 1984.

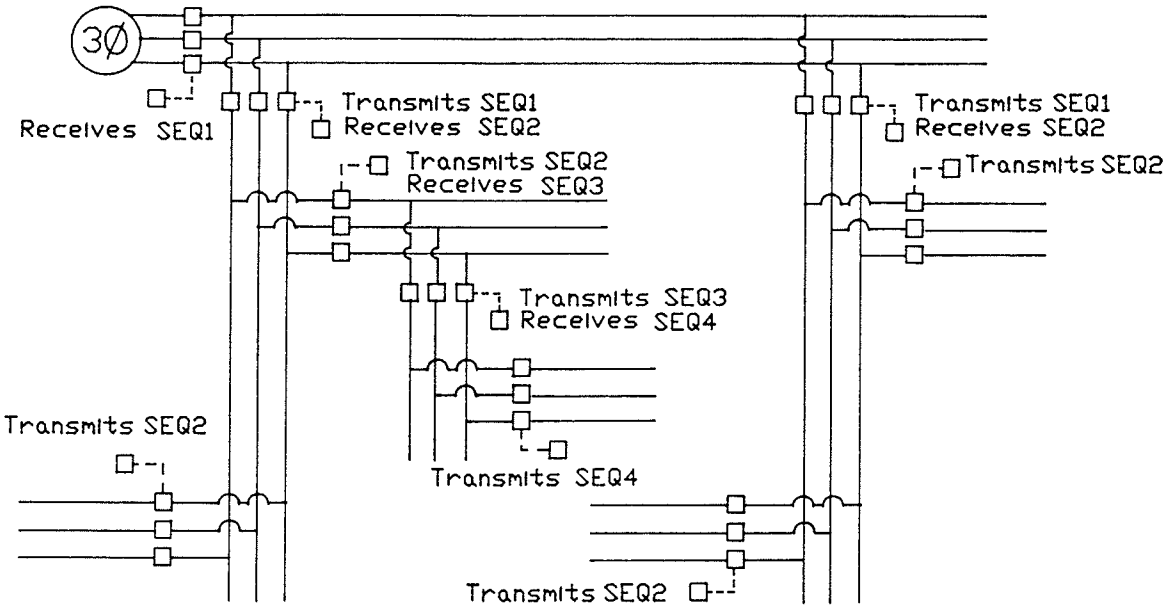


Figure 3.2: Application of FSK to the ZSIP System.

An FSK based carrier current system has two main advantages. First, because the transmissions are digital, it makes the system very flexible. To add new levels of distribution to the system, or to change some of the sequences, all that is required is a change in the software. The other advantage is the system's cost. Since the hardware is simple and readily available, each unit should be inexpensive to produce.

Unfortunately, the disadvantages of FSK in this application far outweigh its benefits:

1) Referring to Fig. 3.2 again, consider the situation for a fault occurring on the furthest downstream bus. The relay here is required to block its corresponding upstream relay using SEQ4. This relay is meanwhile trying to block its upstream relay using SEQ3, and so on up the distribution. It can be clearly seen that there is going to be a problem here. When more than one unit tries to transmit at once, interference results since each unit is trying to use the same two frequencies.

2) The way to get around problem (1) is to introduce delays. The furthest downstream unit gets the first chance to transmit, and, therefore, needs no delay. The upstream units do have delays which increase in duration as you move upstream (notice how this is starting to look like TCP again!). Using this technique eliminates the interference problem but causes another problem. Since the delays described above must be very small to maintain the advantages of the ZSIP system, the transmission must be very fast. To keep the total delay time below one ac cycle for five levels of distribution, a transmission speed of exceeding 2400 baud is required. In July and August of 1988, FSK prototypes using the EXAR 2207/2211 and National Semiconductor LM1893 were

tested using the University's electrical wiring as the transmission medium. It was found that the fastest reliable FSK transmission achievable was only about 600 baud. This is far too slow for the ZSIP system.

3) The final problem with FSK is its vulnerability to random noise. If one is sending a sequence consisting of 8 bits of data, and even one bit is corrupted by noise, then the entire transmission is garbage. To make an FSK system reliable and secure, a lot of redundancy and error checking would have to be incorporated into the transmission scheme.

3.2.2 Multi-frequency System

Presently, systems using amplitude modulated tones make up the majority of carrier current remote control systems [5]. In these devices, a high frequency signal is injected into the ac supply to indicate to a remote unit that a particular action is to be carried out. Different tones are used to select which of several actions is to be performed. In some devices, these tones occupy a relatively low frequency band, between say 3kHz and 10kHz. However, to avoid interference from low frequency noise associated with arcing faults, higher frequencies need to be chosen for this application (60 - 100kHz for example).

Figure 3.3 shows how tones could be applied to the carrier ZSIP system. The technique is similar to FSK except that adjacent relays now communicate by using a particular tone. Tone f_4 uses the lowest frequency, f_3 the second lowest, and so on. For a relay to be blocked, it must both "hear" the required tone, and "see" a fault on the bus. This prevents relays from other distribution branches from blocking relays accidentally. In this system, there are n-1 tones required for n levels of distribution.

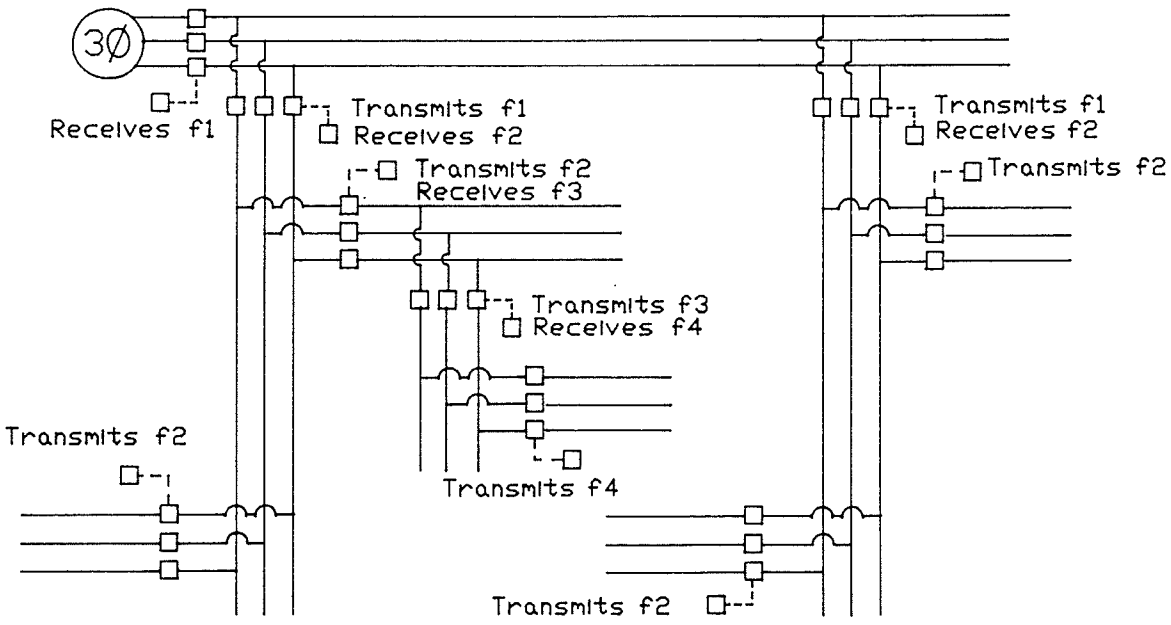


Figure 3.3: Application of Tones to the ZSIP System.

The advantages of a multi-frequency PLC system are numerous:

1) The system is very reliable since intermittent noise won't affect the receiver. Some type of voting system incorporated into the software would make the system resistant to random noise while maintaining a good deal of sensitivity [6].

2) The system is very fast since all the transmitters can operate at once, hence, there are no delays required. Manufacturer's data states that a tone decoder will typically lock onto a signal with frequency f_0 in 10 cycles of f_0. *i.e.* for f_0 = 80kHz, t = 0.125ms.

3) The cost to manufacture each unit is small since the hardware is cheap and readily available.

The only real drawback to this type of system is that it is not quite as flexible as FSK since it is the hardware, not software, that determines the number and frequency of the tones.

3.2.3 Blocking Signal Conclusion

Overall, it seems that a multi-frequency system is more suitable for this application than is an FSK system. FSK would be better suited to a use where only one transmitter is used at once, and where transmission speed is not a major concern. For example, FSK would be ideal for remotely interrogating devices, such as wattmeters, over the power lines.

3.3 Hardware Design

Since the protective relays used in the ZSIP system are microprocessor based, it only makes sense that the carrier current unit should be microprocessor based as well. This way, a common processor could be used to run both the relay and the carrier system, thus reducing the total cost of incorporating the carrier current concept into the ZSIP system. The microprocessor around which the prototype units were built was the Motorola 6802. This processor is basically an enhanced version of Motorola's popular 6800 with 128 bytes of internal RAM provided. The clock rate of 1MHz provides more than enough processor speed for this application. The 6802 is mounted on a PCB along with a Signetics 8732 4K EPROM, 2 Motorola 6821 PIAs for I/O control, and a socket for an additional 2K of RAM.

The remaining hardware is comprised of the carrier current transmitter and receiver. This part of the hardware is controlled by the microprocessor through the two PIAs described above. In the receiver, hardware is provided for the

reception of five different tones. As mentioned before, five tones will allow for up to six levels of distribution. Since three or four is the typical number of levels used in most installations, there should be no problem with this limitation.

The frequencies of the tones used are 95, 87, 79, 71, and 63kHz. A great deal of compromise was required when choosing these frequencies. First, since the measurements that were previously described extended only up to 100kHz, this is the maximum frequency that is known to be "safe". On the other hand, the frequencies should not be too low or there will be insufficient coupling of the signal between the distribution system's busses. One other consideration deals with the wavelength of the PLC compared to the length of the bus. To prevent standing waves, λ should be at least 10 times as long as the bus [5],

i.e. for a bus length of 300m, $\quad 10\times(300\text{m}) \leq \lambda$

$$3000\text{m} \leq \frac{3\times10^8}{f} \qquad f \leq 100\text{kHz}$$

This criterion further justifies the statement that the frequency range should not exceed 100kHz.

The frequency spacing of 8kHz is chosen basically arbitrarily to yield the frequency band listed above. However, trouble could arise if this spacing was made too small, since there is some frequency deviation in both the transmitter and receiver.

3.3.1 Carrier Current Receiver

A block diagram of the system's hardware is given in Fig. 3.4 and a schematic in Fig. 3.5. For a signal to be received by the carrier current hardware, it must first pass through a very sharp band-pass filter. This filter is formed using an A042 IF transformer in parallel with some capacitance [7]. This capacitance is chosen by the microprocessor, depending on the frequency of the signal trying to be detected. Dry reed relays are used to switch in the appropriate capacitance to achieve the desired center frequency. It should be noted that the user informs the microprocessor of the tones to be used by setting dip switches on the circuit board. These switches are read immediately after the PIAs are initialized and the settings are stored in RAM for future reference.

Next, the received signal passes through an amplitude limiter. This device is basically just two silicon diodes which clip any voltage above 0.7V. This attenuation is necessary because the capture bandwidth of the tone decoders increases as a function of the input voltage. By limiting the amplitude of the input signal, we are assured that the bandwidth will not rise above a known value.

The tone decoders used (EXAR 567s) were chosen on the basis of their cost, speed, and sensitivity to small signals. In this application, these devices have a lock-in time of less than 0.15ms. and, can detect as little as 40mV of signal

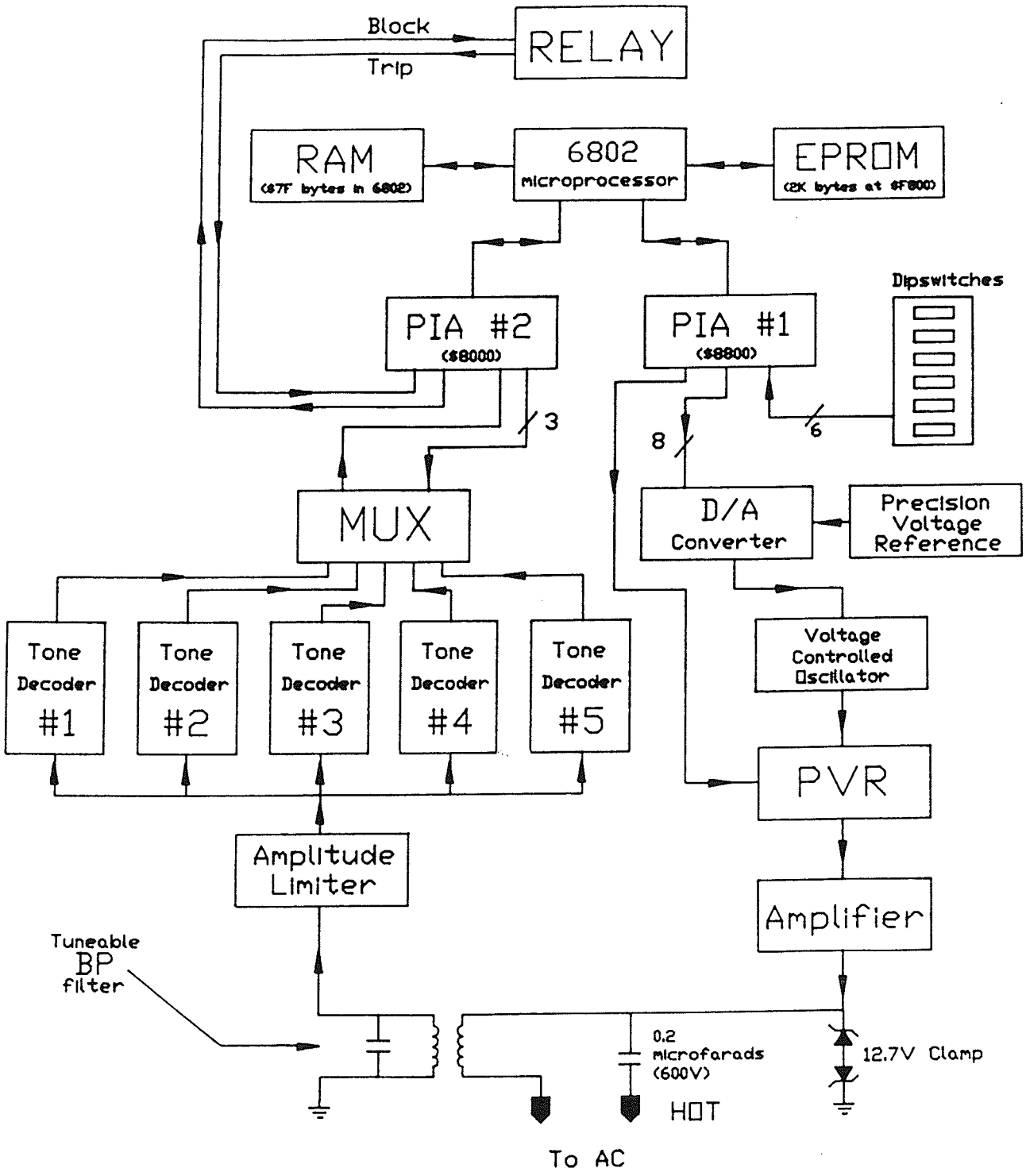


Figure 3.4: Block Diagram of Hardware.

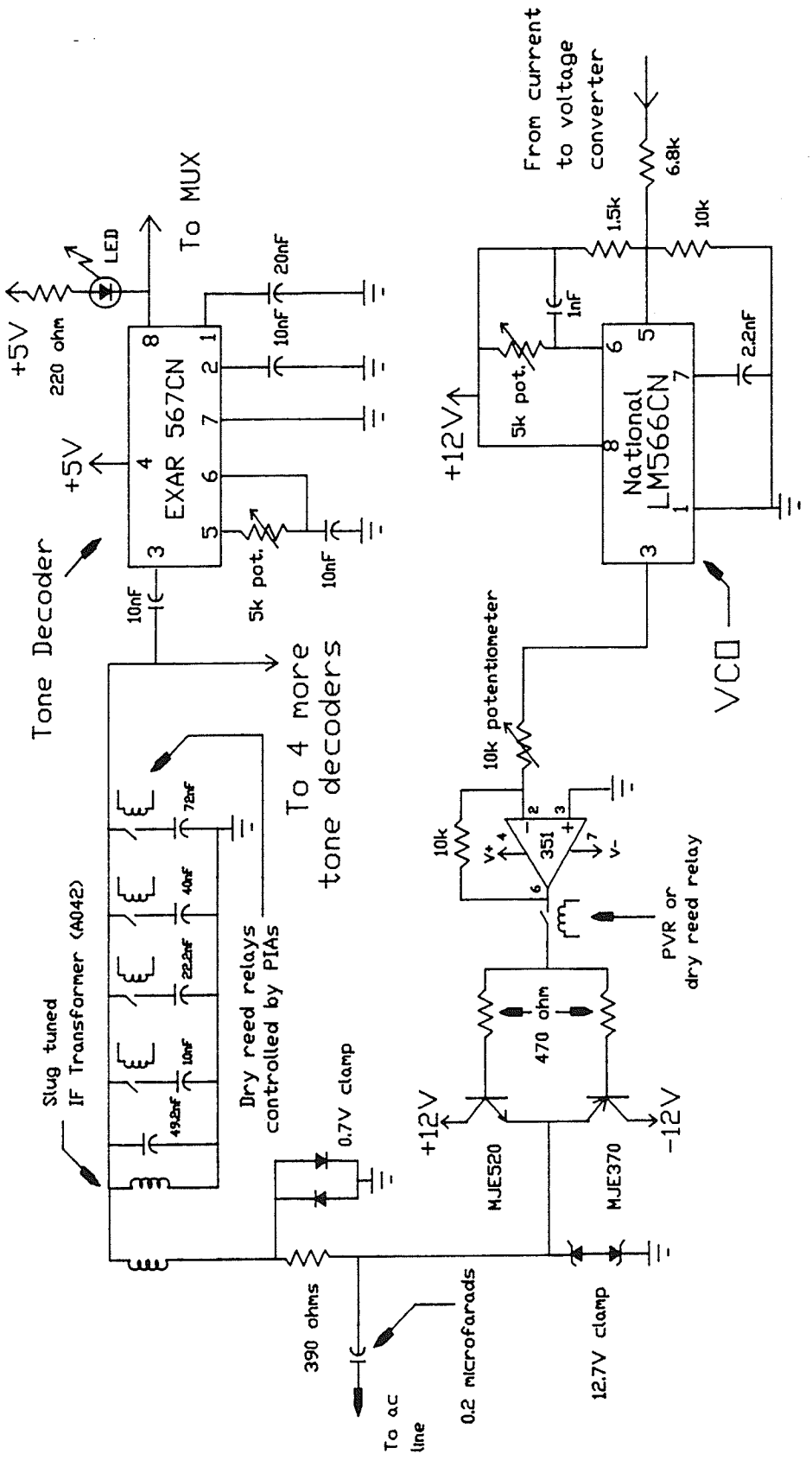


Figure 3.5: Part 1 of Simplified Hardware Schematic.

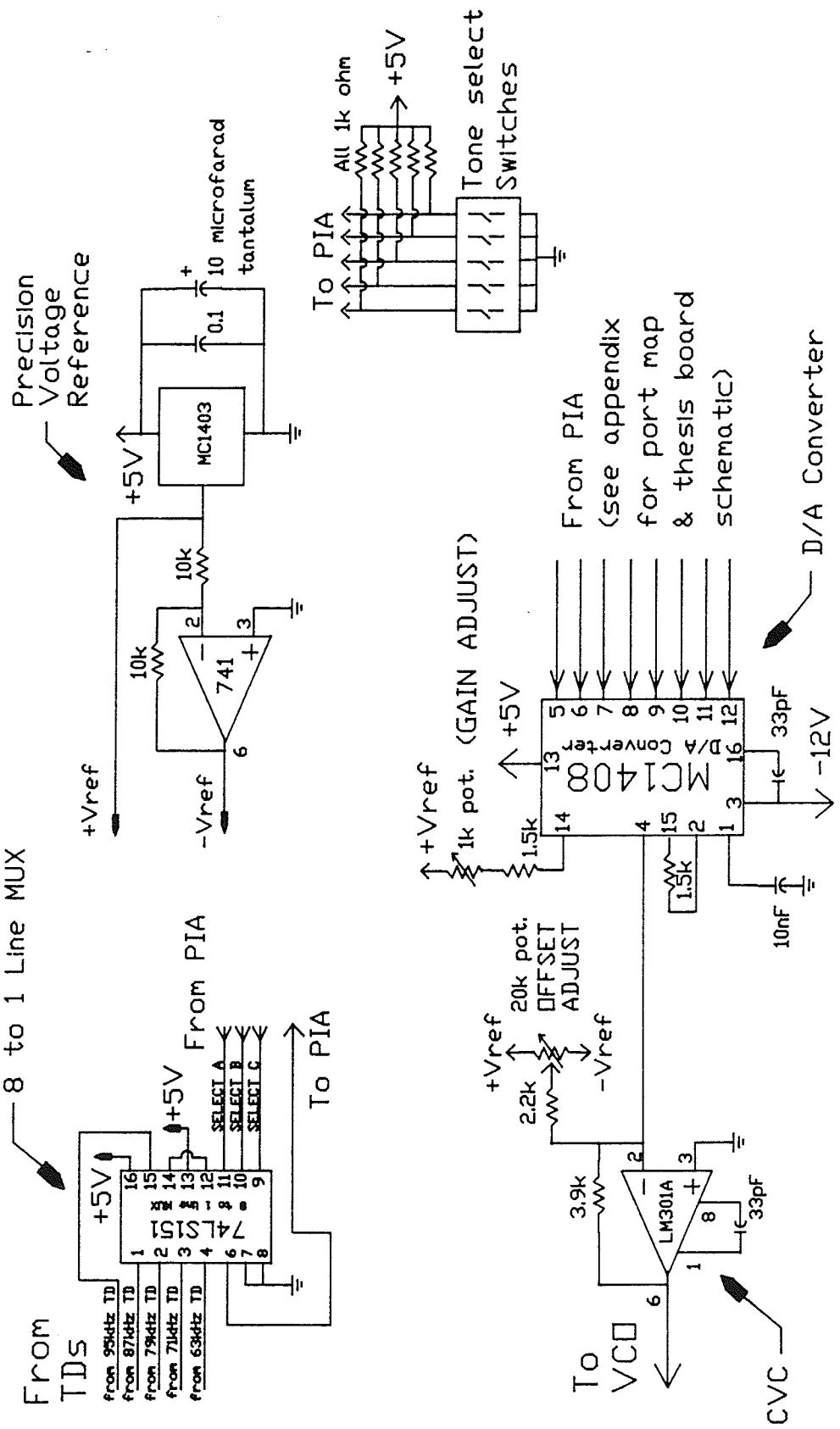


Figure 3.5: Part 2 of Simplified Hardware Schematic.

modulated with a great deal of Gaussian noise. The tone decoder center frequency can be very accurately controlled by adjusting a small ten turn potentiometer. Since the BW is so small, exact tuning is important for the unit to perform properly.

Although all five tone decoders are "listening" for a particular tone all the time, the output of only one of them makes it to the PIA. This is accomplished through the use of an 8 to 1 line multiplexer (MUX). This device receives three SELECT lines from the PIA which tell it which tone decoder output to route to the microprocessor. As an added advantage, the MUX has a pin with an inverted output which converts the active low outputs from the TDs to active high signals before being feeding them to the PIA. Note that the MUX's function could have been implemented in software rather than hardware. However, the cost of the MUX is small and, the software is both faster and simpler when the MUX is used. For this reason, the MUX is an integral part of the hardware design.

3.3.2 Carrier Current Transmitter

Control of the transmission hardware is accomplished through PIA #1. To select a particular tone, the PIA simply sends eight bits to the input of the digital to analog (D/A) converter. These eight bits are inversely proportional to the frequency desired with 11111111 being the lowest possible frequency and 00000000 being the highest possible frequency.

The correct binary numbers to yield the tones mentioned in the previous sections were determined experimentally and are now written into the software (see next section). The D/A converter translates the received bits into an analog signal which is passed onto the current to voltage converter.

Since it is important to make the D/A very stable, a special voltage reference is used. The MC1403 precision voltage reference provides a positive reference voltage of exactly +2.500V to the D/A converter. This $+V_{ref}$ is then inverted by a 741 op-amp to provide bipolar references of $\pm V_{ref}$ to the current to voltage converter (CVC).

The CVC is basically just an inverting op-amp with an adjustable offset voltage. Its purpose is to convert the D/A converter's output current to an analog voltage suitable for input to the voltage controlled oscillator (VCO) without loading the D/A's circuits. The offset voltage is adjusted with a small potentiometer and serves to vary the bias voltage to the VCO's input.

The high frequency tones to be transmitted are produced by the VCO. In the prototype units, National LM566s are used, although other suitable products are available from other manufacturers. The LM566 accepts an analog input voltage and produces a square or triangular wave output with a frequency proportional to this input. For this application, the square wave output is chosen because it results in a higher RMS voltage for equivalent supply voltages.

Following the VCO is a photovoltaic relay (PVR). This device acts like a very fast dry reed relay in that it can switch both positive and negative voltages with reasonably large amplitudes. Internally, the PVR uses a dielectrically isolated LED to energize a bidirectional MOSFET (BOSFET) switch. Since there are no moving parts, this device has an almost unlimited life span. The PVR is necessary due to the nature of the transmission scheme (see section on software). Since the transmitted signal appears huge at the tone decoder input (due to its proximity), the tone decoder is essentially blind to all small signals on the line as long as the transmitted signal is present. For this reason, the microprocessor must be able to halt the transmission from time to time so that the tone decoder gets a chance to operate. The PVR provides this capability. The problem with PVRs is that they are relatively new and, hence, fairly expensive. While waiting for the PVRs to arrive, dry reed relays were employed in their place and relatively good performance was still obtained. Thus, as an economy measure, dry reed relays could be substituted here at the expense of some performance.

The final transmitter stage is the push-pull amplifier which uses two large power transistors to amplify the input signal and drive the ac line through a 0.2μF capacitor. These transistors each handle about 6W and tend to get very hot when used for an extended period of time. However, since in actual operation the length of a transmission is short (say 10ms.)

this is not a major concern. As a safety precaution, a 12.7V clamp circuit is employed at the output of the amplifier to clip any high frequency noise spikes that may make their way into the transmitter from the ac line.

3.4 Software Design

Software to run the carrier current hardware is written in machine code (for speed) and stored in the 4K EPROM on the microprocessor board. This software takes care of two problems that exist with the previously described hardware.

Due to their proximity to the carrier current transmitter, the tone decoders are essentially blind to all small signals on the ac line while the transmitter is operating. To get around this problem, there must be some elaborate method of alternately transmitting and receiving in such a way as to guarantee a sufficient transmit/receive overlap between adjacent relays. Figure 3.6 shows this technique graphically. Since each unit may "see" a fault at a slightly different time, adjacent units must share an overlap time of at least 0.5ms. no matter when the individual transmit/receive cycles begin. In Fig. 3.6, the origin indicates the moment that the fault is detected by the relay. It can be clearly seen that no matter how the graphs are shifted, there is guaranteed to be an overlap of at least 0.5ms. within the 10ms. window. Practically speaking, the difference between the times that individual relays sense the

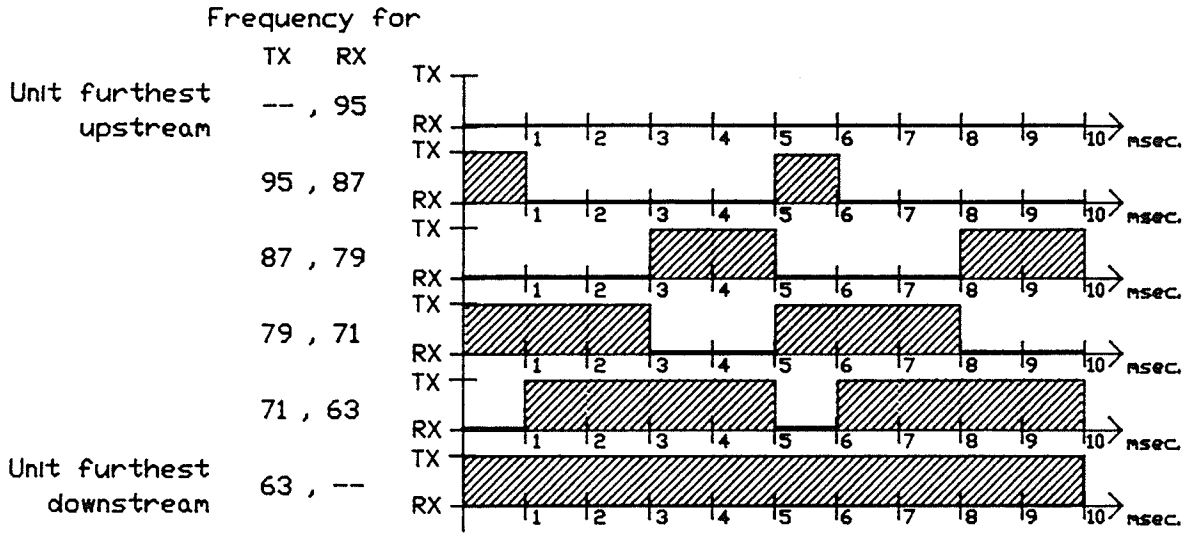


Figure 3.6: Transmit/Receive Timing Diagram.

fault is small and so the actual TX/RX timing would look very similar to that shown in Fig. 3.6. Here, the overlap between adjacent units has been intentionally maximized for optimum performance.

The second problem taken care of by the software is the susceptibility to spurious noise. Since this noise is of a random nature, the best way to distinguish it from a real signal is to only recognize signals with a duty cycle greater than some threshold value. This is accomplished in the μP based system by polling the MUX output every 0.2ms. (the tone decoder lock-in/ring time) and blocking the relay only if at least two of the last four samples indicate that the required

tone is present. Figure 3.7 details the sampling and voting that occurs within a 2.2ms. window.

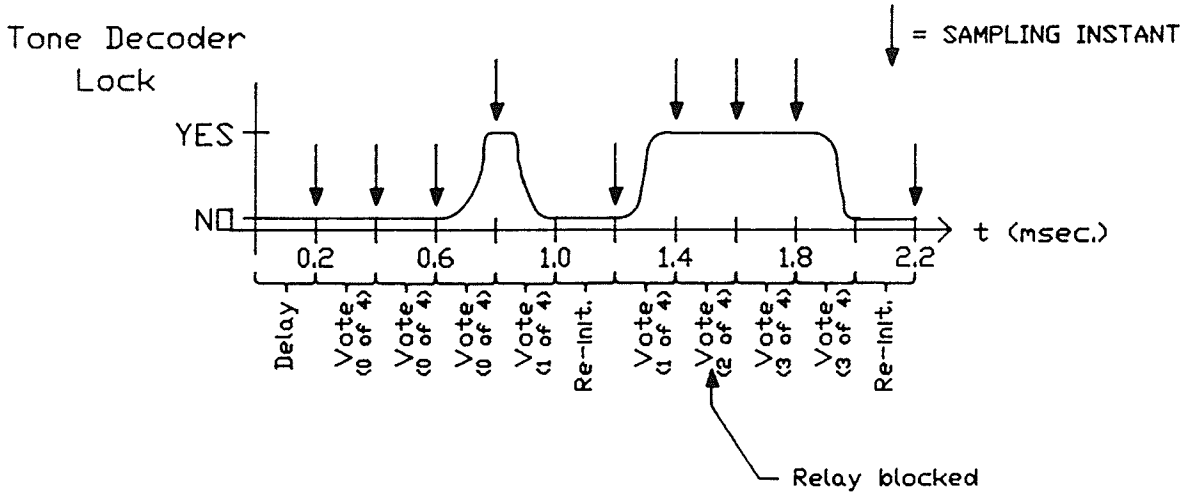


Figure 3.7: Receiver Sampling Scheme.

The above voting scheme is only one of many possible schemes. In the section on testing, some results using this method under extremely noisy conditions are presented, along with the results using a slightly modified algorithm.

Figure 3.8 is a flowchart which shows the salient features of the software. Additionally, a complete listing of the program is given in Appendix C. Some noteworthy information not found in the flowchart follows:

1) Dip switches are read immediately after PIA initialization and, if no errors are found, their settings are stored in RAM for future reference. This means that the only way to make the unit recognize a change in dip switch settings is to turn the unit off, then back on. If an error is found in the dip switch information, the programming error light is turned on and program execution halts. Either of the following constitutes a programming error:

   (i) more than one switch set (ON)

   (ii) a switch with number greater than 6 set.

2) Each dip switch represents a particular pair of transmit and receive tones. The low numbered switches correspond to the high frequencies while the high numbered switches correspond to lower frequencies. When configuring two units for communication, it is essential that the downstream relay use the higher numbered switch. In addition, if units are to communicate, they must use consecutively numbered dip switches (e.g. 1 and 2, or 4 and 5, etc.). This implies that the furthest downstream relay in the distribution system must use the highest numbered dip switch. The second most downstream relay must use the second highest dip switch, and so on.

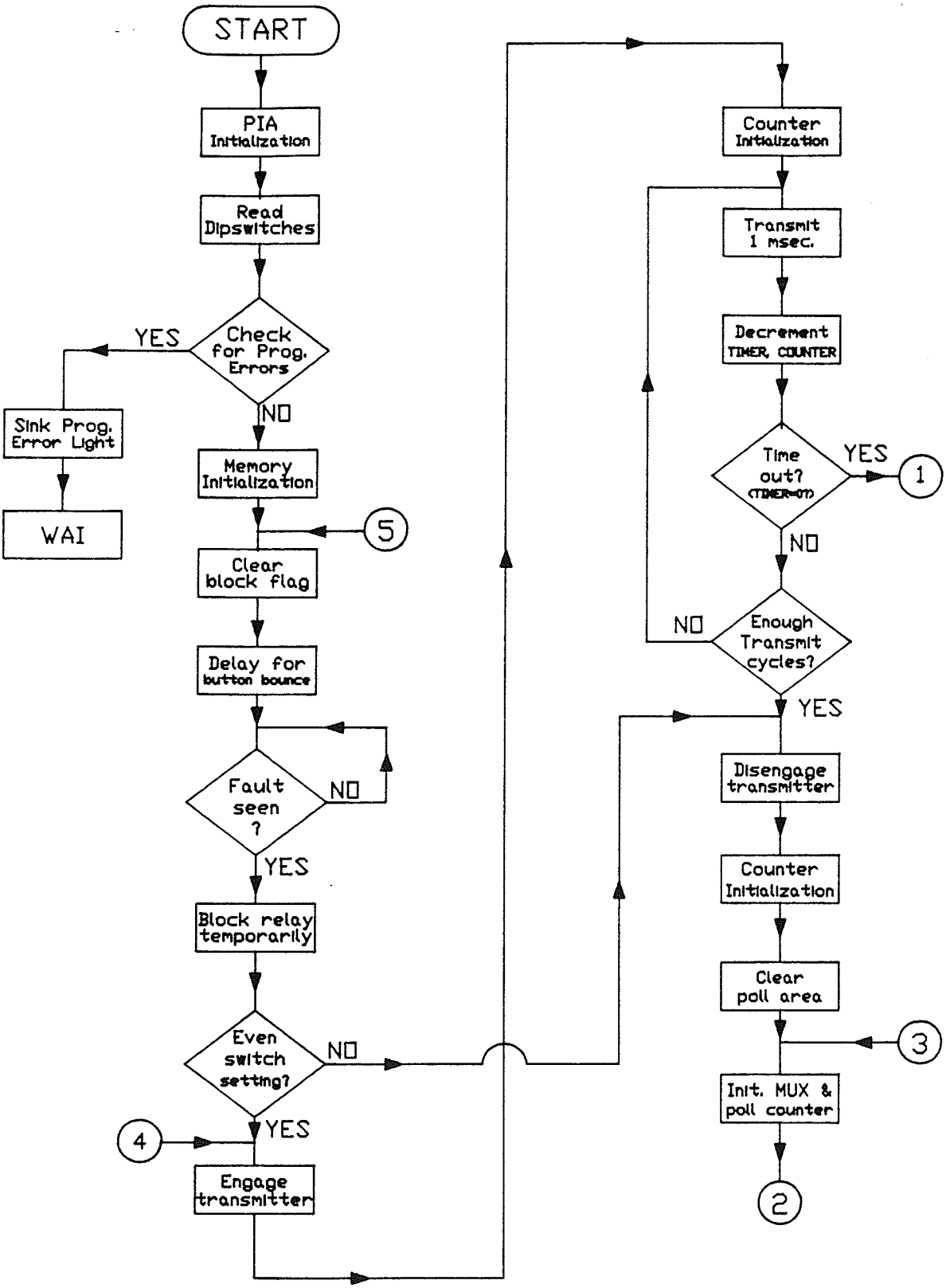


Figure 3.8: Flowchart of Carrier Current Software - Part 1.

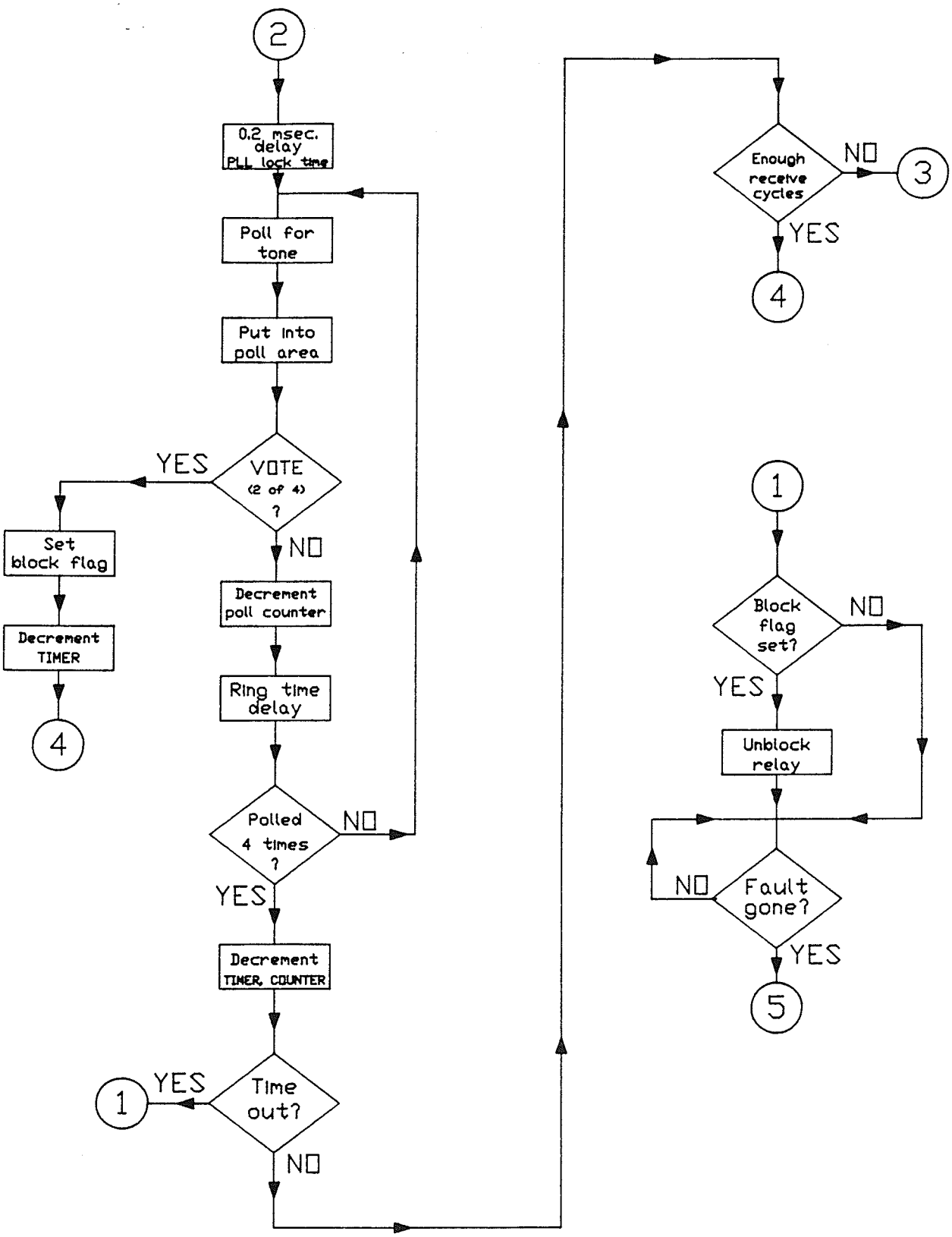


Figure 3.8: Flowchart of Carrier Current Software - Part 2.

3) A red button on each unit serves to provide a TRIP signal to the unit for testing purposes. Since for a transmission to occur both units must "see" a fault, a connector is furnished at the back of each unit. When the wires from each unit are connected, the result is that when either test button is pressed, both units "see" a fault at the same time.

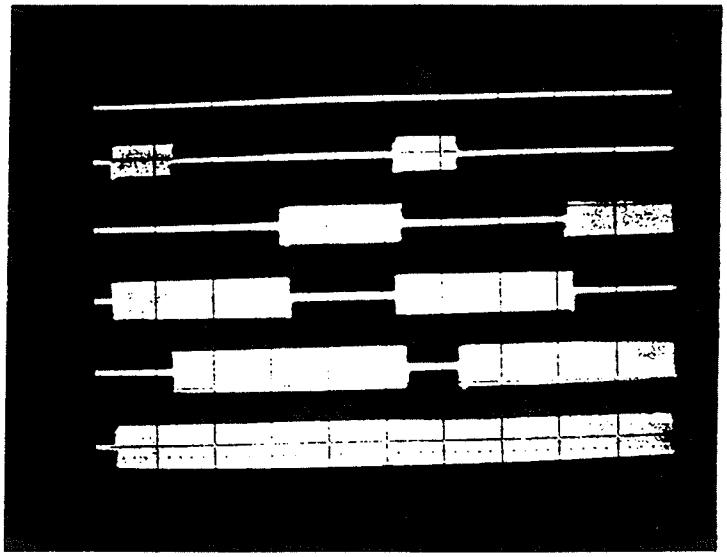
4) In the software, a default communication window of 10ms. is used. Each relay will be temporarily blocked at the start of this interval and will remain blocked until the 10ms. expires. In this time, the unit will have tried to detect the required blocking signal being sent from the downstream relay. If this signal was detected, then the relay will remain blocked until the fault disappears (or, in these test prototypes, for approximately three seconds). If a block signal was not detected in the 10ms. window, then the relay will be unblocked and allowed to go into its timed mode. If the 10ms. window is not satisfactory, it can be changed by altering the value of the variable TIMER and reassembling the code.

Chapter IV

TESTING

4.1 Quantitative Data

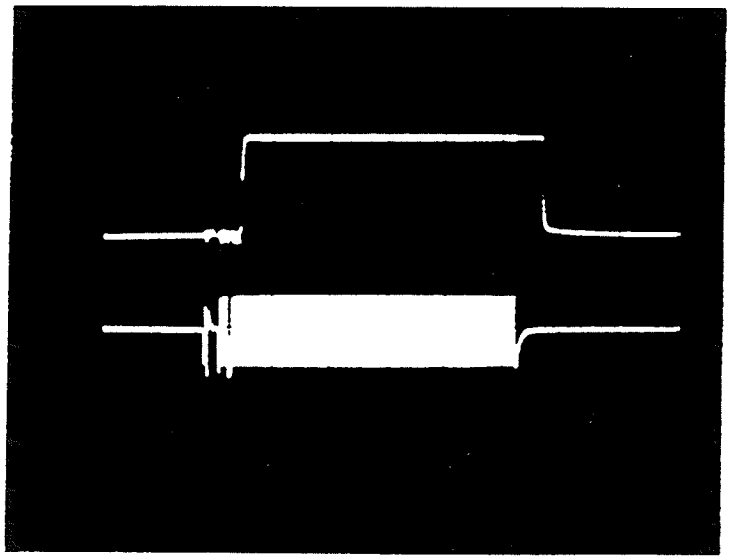
To determine if the carrier current hardware is performing as it should, several tests need to be performed. The first test is to find out if the actual transmit/receive scheme is the same as the planned scheme, shown in Fig. 3.6. Here, an analog capture scope is used to view the carrier output using a time scale of 1 ms/div. By using multiple triggers, a picture can be built up showing the unit's output for each possible dip switch setting (see Fig. 4.1). The leftmost side of this picture corresponds to the instant when the unit's test button is pressed. Notice that Fig. 4.1 is nearly identical to the desired output shown in Fig. 3.6. The only discrepancy is that in the actual output there is some overlap of the transmit cycles. For example, the 3rd and 4th traces have a transmit overlap of about 0.1ms. This imperfection is caused by the slow operation of the dry reed relay which is used to engage and disengage the transmitter. In the software, this relay is opened about 0.2ms. before the beginning of the receive cycle, although it seems to take about 0.3ms. to actually cease transmitting. Utilizing a PVR to engage the transmitter would eliminate this delay, since



20V/div

1ms/div

Figure 4.1: Photo of Actual Transmit/Receive Timing.



TOP TRACE
CHANNEL A
2V/div

BOTTOM TRACE
CHANNEL B
10V/div

0.2ms/div

Figure 4.2: Photo of Tone Decoder Lock-in Response.

it operates about twice as fast as a dry reed relay.

Figure 4.2 shows the scope trace resulting from a test to measure the actual tone decoder lock-in time. The bottom trace is a 95kHz input signal to one of the unit's five tone decoders. The top trace is the signal emerging from the inverting output of the MUX. Notice that the tone decoder has both a finite lock-in time and a finite ring time. It appears that the time to achieve a lock is about 0.125 - 0.15 ms. Most of this delay is due to the tone decoder's phase locked loop (PLL), although some is due to the propagation delay in the MUX. The tone decoder ring time seems to be only about 0.1ms in duration. Finally, note the distortion at the beginning of the transmit cycle. The exact cause of this is unknown, although it is suspected that the dry reed relay is at least part of the problem.

4.2 Use of Pseudo Bus

One requirement of the carrier current system is that a signal from it should be able to couple into neighboring phases of the distribution system through the small parallel capacitance. This is necessary so that a blocking signal can skirt around a fault on one phase of the distribution system. In a typical installation, the bus-bars might measure 3" wide, have 6" spacing, and run parallel for 1500'. Therefore, a good guess for this parallel capacitance is:

$$C = \frac{\varepsilon A}{d} \qquad \text{where: } A = \text{area}$$

$$= 3(\text{in}) \times 0.0254(\text{m/in}) \times 1500(\text{ft})$$

$$\times 0.3048(\text{m/ft}) = 34.8\ \text{m}^2$$

$$d = \text{spacing} = 6(\text{in}) \times 0.0254(\text{m/in}) = 0.1524\ \text{m}$$

$$\varepsilon = \text{permittivity} \simeq \varepsilon_0 = \text{permittivity of free space}$$

$$= 8.85\times10^{-12}\ \text{F/m}$$

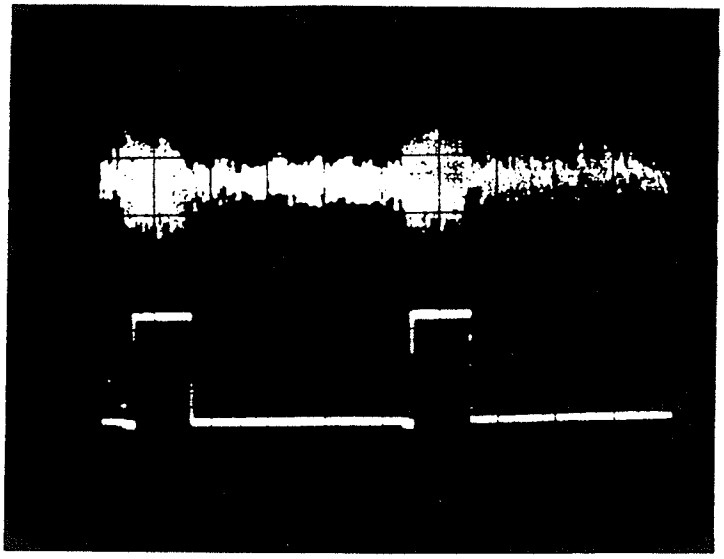
$$\Rightarrow C = 2.02\times10^{-9}\ \text{F} = 2.02\ \text{nF}$$

Since the third phase is twice as far away (12"), the capacitance will be roughly half of this, or 1nF.

To test the carrier system's ability to couple between phases of a distribution system, a piece of pseudo-bus is constructed. This pseudo-bus was made from three, 2m pieces of 1.5" copper bus-bar with a tiny spacing between the phases equal to the thickness of a piece of black electrical tape. A capacitance meter shows that this bus has a capacitance of 2.2nF between the center phase and either outer phase, and about 1.5nF between the two outside phases. This is very similar the the capacitances expected from the long three phase distribution bus described above. The two carrier current prototypes are found to communicate flawlessly between the phases of this pseudo-bus. To determine the minimum phase-to-phase capacitance for the system to work, the pseudo-bus is shortened to 1m and the three phases are spaced with packing foam to decrease the capacitance. Using this configuration, the capacitance from the inside conductor to one of the two outside conductors was about 250pF. Capacitance

between the outside conductors was about 150pF. Even with this tiny capacitance, the carrier system works perfectly at all but the very lowest tone. For this reason, there should be no difficulty coupling a blocking signal into nearby phases in an actual distribution system.

To test the carrier system's susceptibility to electrical noise, a Wavetek pseudo-random noise generator is used. This device has a 100kHz bandwidth and, therefore, produces noise very similar to that which can be expected during an arcing ground fault. In this test, an attempt was made to transmit between the two outermost phases of the pseudo-bus. The noise generator is coupled, through a resistance box, into the same phase as the carrier current receiver. The series resistance serves to provide many discrete amplitude levels so that the noise susceptibility can be evaluated over a wide range of conditions. It soon becomes clear that the receiver has no trouble picking out the desired signals even when there is a tremendous amount of noise present. Figures 4.3 and 4.4 are taken with an analog storage scope for different levels of injected noise. In each of these figures, the top trace is the noise present on the receiving bus-bar and the bottom trace is the output from the receiver MUX. Figure 4.4 is particularly impressive because the blocking tone is indistinguishable from the rest of the noise on the bus. Although the duration of the 'ON' signal from the MUX output is shorter than in Fig. 4.3, it is still more than



TOP TRACE
CHANNEL A
0.5V/div

BOTTOM TRACE
CHANNEL B
2V/div

1ms/div

Figure 4.3: TD Response with Small Amount of Added Noise.

TOP TRACE
CHANNEL A
0.5V/div

BOTTOM TRACE
CHANNEL B
2V/div

1ms/div

Figure 4.4: TD Response with Large Amount of Added Noise.

long enough to cause the relay to block.

Next, tests need to be performed to determine the absolute maximum amount of noise that the system can tolerate. There are actually two noise limits to be determined:

1) The noise limit for reliable operation.
2) The noise limit for security.

However, as it turns out, the system becomes insecure before it becomes unreliable, so, the latter of these two provides the overall noise limit. To compare the amount of noise present here with the amount present during faults, the QUAD A/D1 is used again. To be consistent with the earlier fault measurements, the samples are taken with the same measurement equipment, including the HP filter. Figure 4.5 is a time domain plot which shows the experimentally determined maximum allowable noise when using the "2 of last 4" voting scheme. The Fourier spectrum of this noise is presented in Fig. 4.6. Notice that the noise seems to be distributed quite evenly across the entire frequency spectrum. In the PLC range, the noise varies from about 20 to 85mV, with an average of about 55mV.

To try to improve the carrier system's security, the voting system is changed to "3 of last 4" scheme. Using this algorithm, a much higher noise limit is found (see Fig. 4.7 and 4.8). The frequency spectrum indicates that the PLC noise

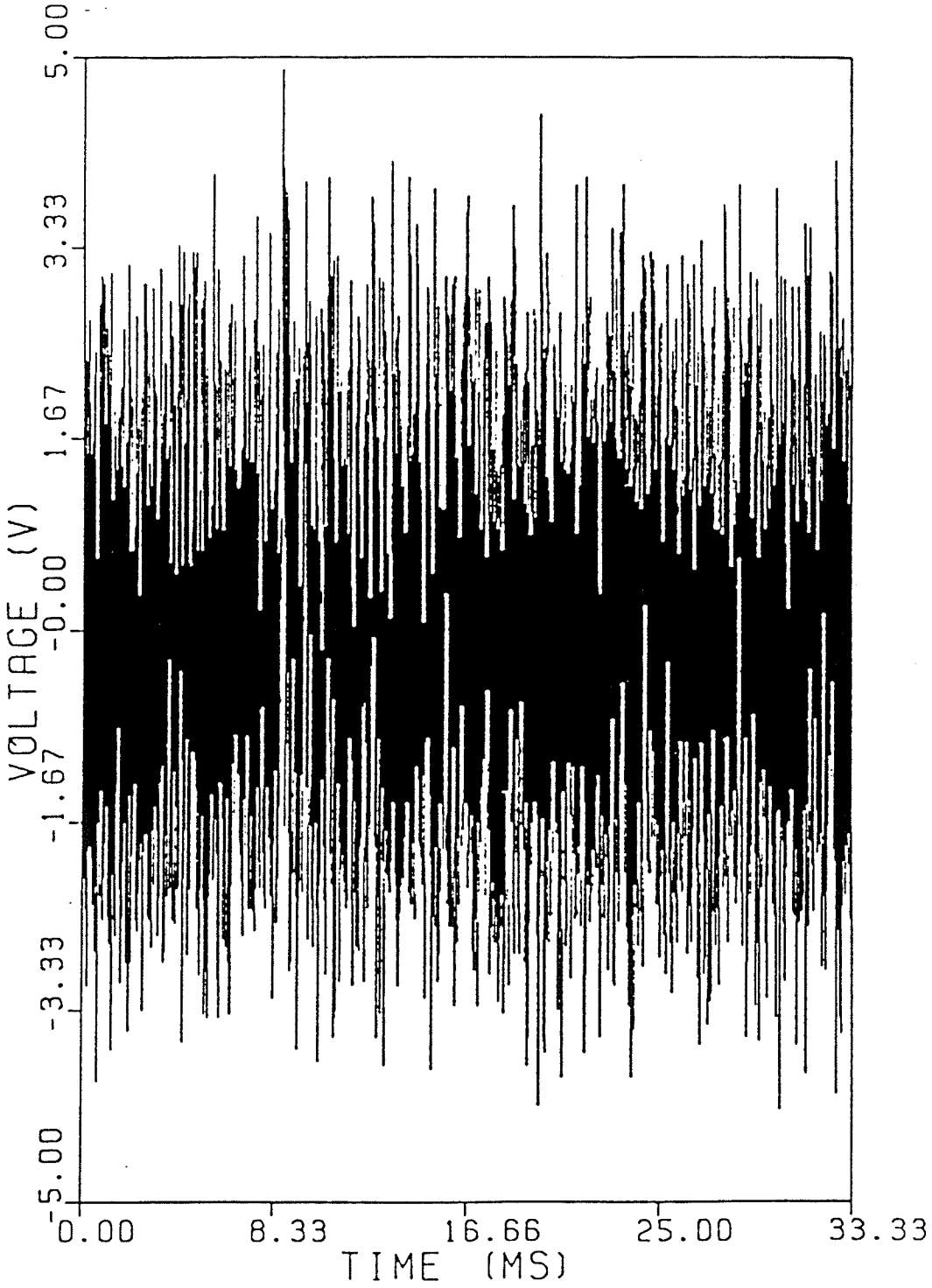


Figure 4.5: Recording of Maximum Allowable Noise when using "2 of Last 4" Voting Scheme.

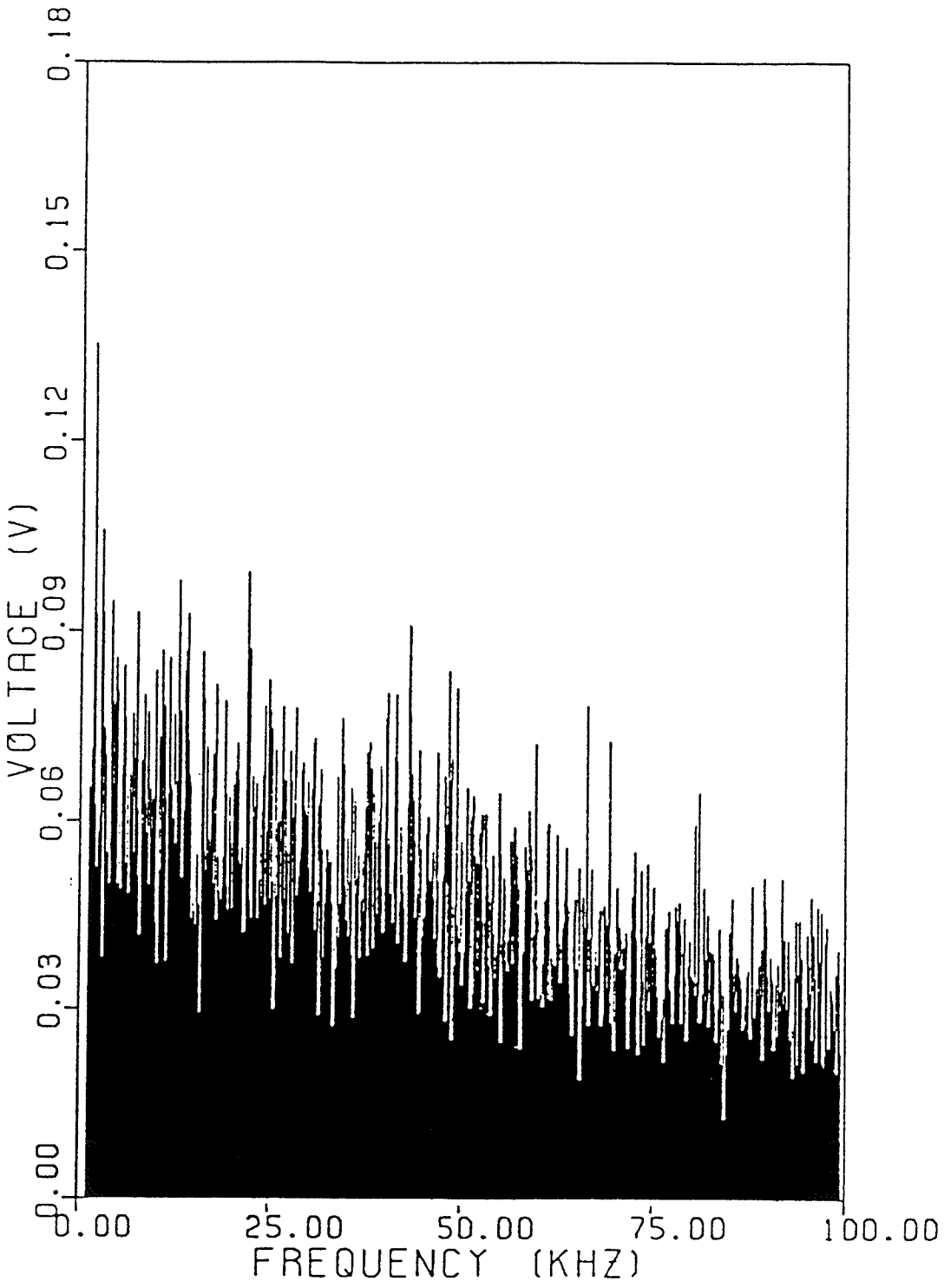


Figure 4.6: Fourier Spectrum of Waveform in Figure 4.5.

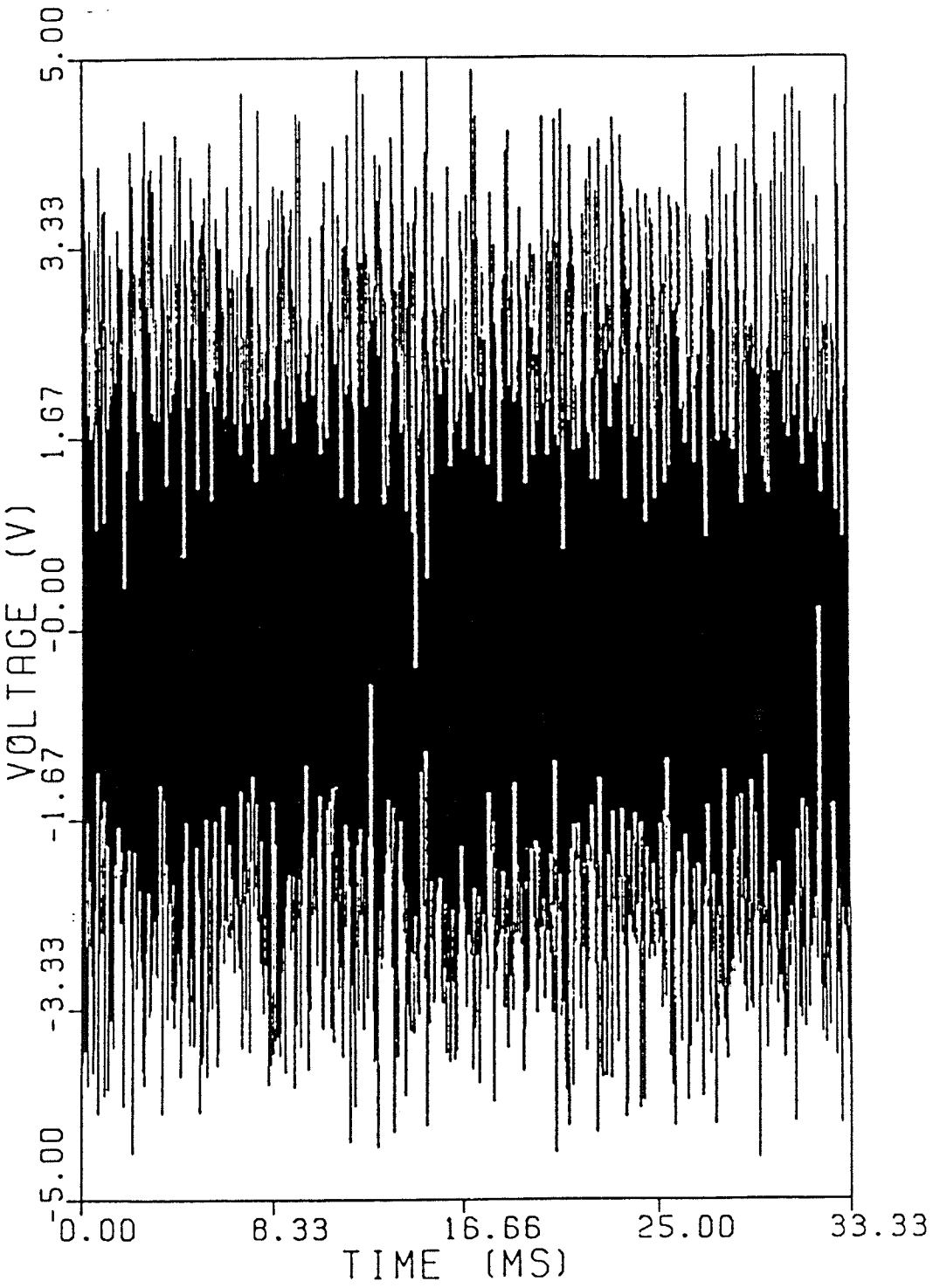


Figure 4.7: Recording of Maximum Allowable Noise when using "3 of Last 4" Voting Scheme.

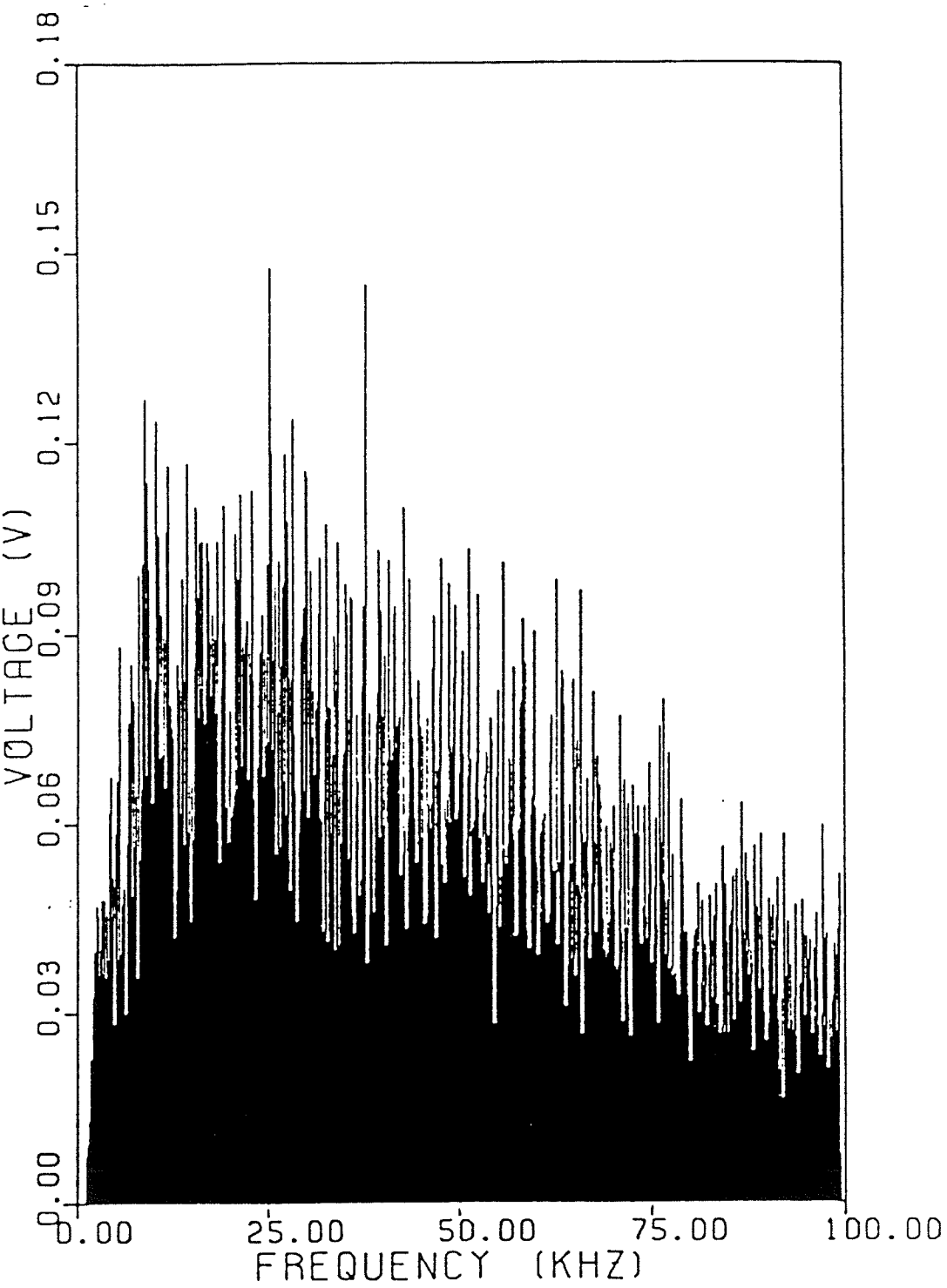


Figure 4.8: Fourier Spectrum of Waveform in Figure 4.7.

ranges from roughly 25 to 80mV, with a mean of about 65mV. This is about seven times as much noise as was found in the 208V fault tests. Even when the fault test data is extrapolated to 600V, the noise immunity is still acceptable. Keep in mind though, these results are only for two possible voting schemes. Given time, many other even better algorithms could probably be found ,and the noise limit could be extended still higher.

To make sure that the carrier current system can operate while low impedance, ground connected loads are present, another test using the pseudo-bus was set-up. Here, the transmit lead was coupled onto phase 'A' and the receive lead was placed on phase 'C' along with a resistance box, one side of which was connected to ground. Even when the resistance to ground was reduced to zero ohms, a blocking signal was still easily received on phase 'C'. However, when the resistance box was moved to phase 'A' (same phase as transmitter), no transmission was possible until the series resistance was raised to 6.8Ω. Shorting phase 'B' (the middle conductor) to ground did not affect the operation in any way. These results suggest that if very low impedance loads are found in an installation, the output impedance and power of the transmitter circuit may need to be increased.

4.3 Tests Using AC Wiring

Transmitting a carrier current signal over conventional 120V wiring is a useful test because it experiences some of the same noise that can be expected on a 600V supply. However, results from this type of test are not conclusive because of the differences between ac wiring and 600V busses and cables. Some of these differences are:

1) AC wiring provides less phase-to-phase coupling because the wiring paths are short and do not necessarily run parallel to each other.

2) The signals emitted by the carrier current transmitter eventually return to the unit through ground. Since the neutral wire in conventional wiring is always connected to ground at some point, each and every device connected to the ac line will short some of the RF energy to ground. This effect causes a tremendous attenuation of carrier current signals. In a factory where a 600V supply is used, the loads are usually connected between phases and rarely have a ground connection.

3) Many devices such as computers and stereos actually have RF filters in their power supply circuits. These filters usually consist of a capacitor between the

hot and neutral wires, and sometimes include an inductor in series with the hot wire, on the device side of the capacitor [5]. Although these devices keep RF energy out, they also attenuate high frequency signals on the line by shorting them to ground. Industrial installations generally have fewer devices which use this sort of power line conditioning, and, hence, experience less attenuation of high frequency signals.

Power line tests conducted over long distances pose somewhat of a problem since no relays are available for testing. Therefore, the only way to force both units to "see" an imaginary fault at the same time is to connect their test buttons together with a wire. Since the trip signal created by the button is only 5V, there is a limit to how long this wire can be. In the tests conducted, successful transmission of all but the lowest tone was possible between two different labs in the Engineering building. These labs were physically about 30m apart and utilized different phases of the ac supply. The fact that only the lowest tone failed to operate properly is indicative of insufficient coupling capacitance, rather than attenuation or noise.

Overall, the test results indicate that the carrier current system works quite well, although some minor changes in the hardware and/or software may be required in the future.

Chapter V

CONCLUSIONS & RECOMMENDATIONS

The ZSIP system provides the best available protection against ground faults occurring on 600V distribution systems. To make the ZSIP system a more saleable commodity, a system to allow the relays to communicate over 600V busses and cables has been developed. The following are conclusions reached in this thesis research with accompanying recommendations where appropriate:

1) Electrical noise measurements have been taken during steady state and fault conditions. Using Goertzel's 2nd order algorithm to analyze the data reveals that the noise present in the PLC frequency range is reasonably small, and therefore, there are no foreseeable interference problems.

2) The relative benefits of FSK and multi-frequency systems were weighed. It was decided that a multi-frequency system is better suited to this application due to its inherent speed and reliability.

3) Two prototype carrier current units have been constructed and tested. These units are microprocessor based and use five tones to provide communication between six levels of distribution. Testing these units revealed the following:

(i) A blocking signal from either unit can easily be transmitted between different phases of the distribution system through the tiny parallel capacitance (as small as 150pF) to be received by the other unit. This ability allows the blocking signal to skirt around a fault on one phase of the supply.

(ii) Each unit's noise rejection is acceptable, but could be improved by exploring other blocking signal verification schemes (i.e. voting).

(iii) If very low impedance ground connected loads are expected ($<6.8\Omega$), the carrier current transmitter's output impedance and power may need to be increased.

4) Due to the success of the prototype units, I foresee little difficulty extending the ZSIP system to incorporate the carrier current concept.

REFERENCES

[1] Burrus, C.S., and Parks, T.W., DFT/FFT and Convolution Algorithms: Theory and Implementation, Wiley Interscience Publications, 1985

[2] Ramirez, R. W., The FFT, Fundamentals and Concepts, Prentice Hall Inc., 1985

[3] Ciarcia, S., Build a Power-line Carrier Current Modem, Byte Magazine, August 1983

[4] Summary of an IEEE Guide to Power Line Carrier Applications, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-99, No. 6, Nov./Dec. 1980

[5] Cunnningham, J.E., and Horn, D.T., Handbook of Remote Control and Automation Techniques, TAB Books Inc., 1984

[6] Ciarcia, S., Tune in and Turn on: A Computerized AC Remote Control System, Byte Magazine, April/May 1978

[7] National Semiconductor, 1984 Linear Supplement, National Semiconductor Corp., 1984

[8] Jung, W.G., IC Op-amp Cookbook, 2nd Edition, Howard W. Sams & Co. Inc., 1980

[9] Sedra, A., and Smith, K., Microelectronic Circuits, CBS College Publishing, 1982

[10] Piety, R.A., Intrabuilding Data Transmission Using Power-line Wiring, Hewlett Packard Journal, May 1987

[11] EXAR Telecommunications Databook, EXAR Corp., 1985

Appendix A

MEASUREMENT SOFTWARE

```
                                ;
                                ; T I M E _ S A M - 3 A/D PORTS USED AT ONCE
                                ;                   12 MHz - 80286 Version
                                ;
0000                            stack   segment para stack 'stack'
0000  0064[                             db 100 dup(0)
        00
                              ]

0064                            stack   ends
0000                            data    segment para public 'data'
0000  0024[                     fcb     db 36 dup(0)
        00
                              ]

0024  00                        dta     db 0
0025  00                        namnum1 db 0
0026  01                        namnum2 db 1
0027  00                        fcharn  db 0
0028  42 65 67 69 6E 6E 69      spmsg   db 'Beginning sampling...make sure clo
ck speed is set at maximum.'
      6E 67 20 73 61 6D 70
      6C 69 6E 67 2E 2E 2E
      6D 61 6B 65 20 73 75
      72 65 20 63 6C 6F 63
      6B 20 73 70 65 65 64
      20 69 73 20 73 65 74
      20 61 74 20 6D 61 78
      69 6D 75 6D 2E
0065  57 61 69 74 69 6E 67      wtmsg   db 'Waiting...'
      2E 2E 2E
006F  50 72 6F 67 72 61 6D      abmsg   db 'Program halted by request.'
      20 68 61 6C 74 65 64
      20 62 79 20 72 65 71
      75 65 73 74 2E
0089  46 49 4C 45 20 45 52      errmsg  db 'FILE ERROR!'
      52 4F 52 21
0094  57 72 69 74 69 6E 67      flmsg   db 'Writing data for sample set #'
      20 64 61 74 61 20 66
      6F 72 20 73 61 6D 70
      6C 65 20 73 65 74 20
      23
00B1  00                        ssnum1  db 0
00B2  00                        ssnum2  db 0
00B3  19E6[                     storage dw 6630 dup(0)
        0000
                              ]

347F  48 24 12 56               lookup1 db 48h, 24h, 12h, 56h
3483  28 64 32 16                       db 28h, 64h, 32h, 16h
3487  08 04 02 01                       db 8h, 4h, 2h, 1h
348B  20 10 05 02               lookup2 db 20h, 10h, 5h, 2h
348F  01 00 00 00                       db 1h, 0h, 0h, 0h
3493  00 00 00 00                       db 0h, 0h, 0h, 0h
```

```
3497  0004[                   decimal  db 4 dup(0)
           00
                        ]

349B  00                      temph    db 0
349C  00                      templ    db 0
349D  19E6                    numsamp  dw 6630
349F  08A2                    numloop  dw 2210
34A1                          data     ends
0000                          code     segment para public 'code'
0000                          main     proc far
                              ;
                              ; Std. preamble except retain DS as PTR to PSP
                              ;
                                       assume cs:code
0000  1E                               push ds
0001  B8 0000                          mov ax,0
0004  50                               push ax
0005  B8 ---- R                        mov ax,data
0008  8E C0                            mov es,ax
                                       assume es:data
                              ;
                              ; Move FCB parameter from PSP to DS
                              ;
000A  BE 005C                          mov si,5ch          ;string's source
000D  BF 0000 R                        mov di,offset fcb   ;string's dest.
0010  B9 000C                          mov cx,12           ;string's length
0013  FC                               cld                 ;fwd. transfer
0014  F3/ A4                          rep movsb           ;move the string
                              ;
                              ; Establish data segment addressability
                              ;
0016  8E D8                            mov ds,ax
                                       assume ds:data
                              ;
                              ; Routine to clear the display
                              ;
0018  B9 0019                          mov cx,25           ;set counter
001B  B0 0A                            mov al,0ah          ;LF character
001D  E8 0250 R               jback:   call dispchar       ;output char.
0020  E2 FB                            loop jback          ;continue...
0022  C6 06 0027 R 3D                  mov fcharn,61       ;# of char. in
                                                           ;upcoming msg
0027  BB 0028 R                        mov bx,offset spmsg ;point to data
002A  E8 0237 R                        call display        ;print msg
002D  B0 0A                            mov al,0ah
002F  E8 0250 R                        call dispchar       ;line feed
0032  EB 2D 90                         jmp sample
                              ;
                              ; Read the T.O.D. clock
                              ;
0035  B4 00                   clock:   mov ah,0            ;clear AH
0037  CD 1A                            int 1ah             ;read T.O.D.
0039  8B DA                            mov bx,dx           ;save LSW
```

```
003B  CD 1A                   wait:   int 1ah             ;read T.O.D.
003D  3B DA                           cmp bx,dx           ;same as before?
003F  74 FA                           je wait             ;if yes, go back
0041  B4 01                   frstpt: mov ah,1            ;set lowest bit
0043  CD 16                           int 16h             ;read keyboard
0045  74 12                           jz fine2            ;if no key has
                                                          ;been pressed,
                                                          ;jump ahead
0047  B4 00                           mov ah,0            ;clear AH
0049  CD 16                           int 16h             ;get key
004B  3C 03                           cmp al,3            ;is it Ctrl-C?
004D  75 03                           jne fine1           ;no, jump ahead
004F  E9 0218 R                       jmp abort           ;abort otherwise
0052  3C 1B                   fine1:  cmp al,27           ;is it ESC?
0054  75 03                           jne fine2           ;no, jump ahead
0056  E9 0218 R                       jmp abort           ;abort otherwise
0059  B4 00                   fine2:  mov ah,0
005B  CD 1A                           int 1ah             ;read T.O.D. clock

005D  3B D3                           cmp dx,bx           ;time to sample?
005F  75 E0                           jne frstpt          ;no keep looking
                              ;
                              ; Routine which performs the sampling
                              ;
0061  BA 030A                 sample: mov dx,30ah         ;point DX to the
                                                          ;MUX channel
0064  B0 00                           mov al,0            ;clear AL
0066  EE                             out dx,al           ;set no MUX op.
0067  8B 0E 349F R                    mov cx,numloop      ;get # of samples
006B  BB 00B3 R                       mov bx,offset storage
006E  BE 0300                         mov si,300h         ;conv. 0 address
0071  BF 0302                         mov di,302h         ;conv. 1 address
0074  BD 0304                         mov bp,304h         ;conv. 2 address
0077  8B D6                           mov dx,si           ;point to conv 0
0079  EE                             out dx,al           ;start conv. 0
007A  FA                              cli                 ;3 clock cycles
007B  FA                              cli
007C  FA                              cli
007D  FA                              cli
007E  FA                              cli
007F  FA                              cli
0080  FA                              cli
0081  FA                              cli
0082  FA                              cli
0083  FA                              cli
0084  FA                              cli
0085  FA                              cli
0086  FA                              cli
0087  FA                              cli
0088  FA                              cli
0089  FA                              cli
008A  FA                              cli
008B  FA                              cli
008C  FA                              cli                 ;57 clocks total
```

```
008D  8B D7                     mov dx,di           ;point to conv 1
008F  EE                        out dx,al           ;start conv. 1
0090  FA                        cli                 ;3 clock cycles
0091  FA                        cli
0092  FA                        cli
0093  FA                        cli
0094  FA                        cli
0095  FA                        cli
0096  FA                        cli
0097  FA                        cli
0098  FA                        cli
0099  FA                        cli
009A  FA                        cli                 ;33 clocks total
009B  8B D5                     mov dx,bp           ;point to conv 2
009D  EE                        out dx,al           ;start conv. 2
009E  FA                        cli                 ;3 clock cycles
009F  FA                        cli
00A0  FA                        cli
00A1  FA                        cli
00A2  FA                        cli
00A3  FA                        cli
00A4  FA                        cli
00A5  FA                        cli
00A6  FA                        cli                 ;27 clocks total
00A7  8B D6             rdnstrt: mov dx,si          ;point to conv 0
00A9  ED                        in ax,dx            ;read the data
00AA  EE                        out dx,al           ;restart conv 0
00AB  89 07                     mov [bx],ax         ;store the data
00AD  43                        inc bx              ;point to next
00AE  43                        inc bx              ;storage location
00AF  FA                        cli                 ;3 clock cycles
00B0  FA                        cli                 ;3 more
00B1  F8                        clc                 ;2 clock cycles
00B2  F8                        clc                 ;2 more
00B3  8B D7                     mov dx,di           ;point to conv 1
00B5  ED                        in ax,dx            ;read the data
00B6  EE                        out dx,al           ;restart conv 1
00B7  89 07                     mov [bx],ax         ;store the data
00B9  43                        inc bx              ;point to next
00BA  43                        inc bx              ;storage location
00BB  FA                        cli                 ;3 clock cycles
00BC  FA                        cli                 ;3 more
00BD  F8                        clc                 ;2 clock cycles
00BE  F8                        clc                 ;2 more
00BF  8B D5                     mov dx,bp           ;point to conv 2
00C1  ED                        in ax,dx            ;read the data
00C2  EE                        out dx,al           ;restart conv 2
00C3  89 07                     mov [bx],ax         ;store the data
00C5  43                        inc bx              ;point to next
00C6  43                        inc bx              ;storage location
00C7  E2 DE                     loop rdnstrt        ;continue
00C9  FB                        sti
                        ;
                        ; Set DTA and create file
```

```
                                ;
00CA  BA 0024 R                         mov dx,offset dta
00CD  B4 1A                             mov ah,1ah          ;'set DTA'
00CF  CD 21                             int 21h             ;invoke DOS fcn.
00D1  8A 26 0025 R                      mov ah,namnum1      ;get filespec HB
00D5  80 C4 30                          add ah,30h          ;convert to ASCII
00D8  88 26 00B1 R                      mov ssnum1,ah       ;store it
00DC  8A 26 0026 R                      mov ah,namnum2      ;get filespec LB
00E0  80 C4 30                          add ah,30h          ;convert to ASCII
00E3  88 26 00B2 R                      mov ssnum2,ah       ;store it
00E7  B0 0A                             mov al,0ah          ;linefeed char.
00E9  E8 0250 R                         call dispchar       ;print it
00EC  C6 06 0027 R 1F                   mov fcharn,31       ;store # of chars
00F1  BB 0094 R                         mov bx,offset flmsg ;point to data
00F4  E8 0237 R                         call display        ;print message

00F7  BB 0000 R                         mov bx,offset fcb   ;point to fcb data

00FA  83 C3 08                          add bx,08h          ;posn to filename
00FD  8A 26 0026 R                      mov ah,namnum2      ;get HB
0101  80 C4 30                          add ah,30h          ;convert to ASCII
0104  88 27                             mov [bx],ah         ;move LB to AH
0106  8A 26 0025 R                      mov ah,namnum1      ;move HB to AH
010A  80 FC 00                          cmp ah,0h
010D  74 0C                             je zero             ;jump if HB is
                                                            ;non-existent
010F  83 EB 01                          sub bx,1h           ;posn 7 in flname
0112  8A 26 0025 R                      mov ah,namnum1      ;get HB of flspec
0116  80 C4 30                          add ah,30h          ;convert to ASCII
0119  88 27                             mov [bx],ah         ;mov HB to FCB
011B  8A 26 0026 R              zero:   mov ah,namnum2      ;put LB in AH
011F  FE C4                             inc ah              ;increment it
0121  80 FC 09                          cmp ah,09h          ;compare to 9
0124  7E 06                             jng restore         ;branch if < or =
0126  B4 00                             mov ah,0            ;clear AH
0128  FE 06 0025 R                      inc namnum1         ;increment HB
012C  88 26 0026 R              restore: mov namnum2,ah     ;restore LB
0130  4B                        backfil: dec bx             ;backup 1 char
0131  8A 27                             mov ah,[bx]         ;get char from
                                                            ;fcb in filename
0133  80 FC 20                          cmp ah,20h          ;compare to space
0136  75 05                             jne full            ;if no space jump
0138  C6 07 30                          mov byte ptr [bx],30h ;else put zero
                                                            ;in to replace it
013B  EB F3                             jmp backfil         ;repeat
013D  BA 0000 R                 full:   mov dx,offset fcb
0140  B4 16                             mov ah,16h          ;'create file'
0142  CD 21                             int 21h             ;invoke dos fcn.
0144  3C 00                             cmp al,0            ;create work OK?
0146  74 03                             jz init             ;yes, branch
0148  E9 0230 R                 noname: jmp error           ;no, quit
014B  C7 06 000C R 0000          init:   mov word ptr fcb+0ch,0 ;block=0
0151  C7 06 000E R 0001                  mov word ptr fcb+0eh,1 ;record size=1
0157  C6 06 0020 R 00                    mov fcb+20h,0       ;record=0
```

```
                                ;
                                ; Hex to decimal routine
                                ;
015C  8B 0E 349D R                      mov cx,numsamp      ;# of data items
0160  BB 00B3 R                         mov bx,offset storage
0163                            out     proc near
0163  8B 17                             mov dx,[bx]
0165  86 F2                             xchg dh,dl          ;switch
0167  C6 06 349B R 00                   mov temph,0         ;clear the temp.
016C  C6 06 349C R 00                   mov templ,0         ;data holder
0171  53                                push bx             ;save BX
0172  51                                push cx             ;save count posn
0173  B9 000C                           mov cx,12           ;12 shifts req'd.
0176  BE 0000                           mov si,0
0179  D1 D2                     here:   rcl dx,1            ;rotate left
017B  73 18                             jnc ahead           ;branch if carry
                                                            ;flag is clear
017D  BB 347F R                         mov bx,offset lookup1
0180  A0 349C R                         mov al,templ        ;get old low byte
0183  02 00                             add al,[bx+si]      ;add increment
0185  27                                daa                 ;decimal adjust
0186  A2 349C R                         mov templ,al        ;store
0189  BB 348B R                         mov bx,offset lookup2
018C  A0 349B R                         mov al,temph        ;get old HB
018F  12 00                             adc al,[bx+si]      ;add increment
0191  27                                daa                 ;decimal adjust
0192  A2 349B R                         mov temph,al        ;store
0195  46                        ahead:  inc si              ;next posn. in
                                                            ;lookup tables
0196  E2 E1                             loop here           ;continue til 12
                                                            ;shifts are done
                                out     endp
0198  A0 349C R                         mov al,templ        ;get low byte
019B  B4 00                             mov ah,0            ;clear AH
019D  D1 D0                             rcl ax,1            ;put high nibble
019F  D1 D0                             rcl ax,1            ;into AH
01A1  D1 D0                             rcl ax,1
01A3  D1 D0                             rcl ax,1
01A5  88 26 3499 R                      mov decimal+2,ah    ;store
01A9  B4 00                             mov ah,0            ;clear AH
01AB  D1 D0                             rcl ax,1            ;put low nibble
01AD  D1 D0                             rcl ax,1            ;in AH
01AF  D1 D0                             rcl ax,1
01B1  D1 D0                             rcl ax,1
01B3  88 26 349A R                      mov decimal+3,ah    ;store
01B7  A0 349B R                         mov al,temph        ;get high byte
01BA  B4 00                             mov ah,0            ;clear AH
01BC  D1 D0                             rcl ax,1            ;high nib. to AH
01BE  D1 D0                             rcl ax,1
01C0  D1 D0                             rcl ax,1
01C2  D1 D0                             rcl ax,1
01C4  88 26 3497 R                      mov decimal,ah      ;store
01C8  B4 00                             mov ah,0            ;clear AH
01CA  D1 D0                             rcl ax,1            ;low nib. to AH
```

```
01CC  D1 D0                          rcl ax,1
01CE  D1 D0                          rcl ax,1
01D0  D1 D0                          rcl ax,1
01D2  88 26 3498 R                   mov decimal+1,ah    ;store
                                     ;
                                     ; Write data to the disk
                                     ;
01D6  B9 0004                        mov cx,4            ;set counter
01D9  BB 3497 R                      mov bx,offset decimal
01DC  8A 07                  send:   mov al,[bx]         ;get data
01DE  04 30                          add al,30h          ;convert to ASCII
01E0  A2 0024 R                      mov dta,al          ;store to DTA
01E3  E8 0224 R                      call write          ;output to disk
01E6  43                             inc bx              ;point to next
                                                         ;item of data
01E7  E2 F3                          loop send           ;4 bytes sent yet
01E9  C6 06 0024 R 0D                mov dta,0dh         ;CR
01EE  E8 0224 R                      call write          ;output to disk
01F1  C6 06 0024 R 0A                mov dta,0ah         ;LF
01F6  E8 0224 R                      call write          ;send to disk
01F9  59                             pop cx              ;restore CX
01FA  5B                             pop bx              ;restore BX
01FB  43                             inc bx              ;adv. pointer to
01FC  43                             inc bx              ;next data item
01FD  49                             dec cx              ;decr. counter
01FE  74 03                          jz up
0200  E9 0163 R                      jmp out             ;continue
                                     ;
                                     ; Close file and repeat in 1 hour
                                     ;
0203  BA 0000 R              up:     mov dx,offset fcb
0206  B4 10                          mov ah,10h          ;'close file'
0208  CD 21                          int 21h             ;invoke DOS fcn.
020A  C6 06 0027 R 0A                mov fcharn,10       ;store # of chars
020F  BB 0065 R                      mov bx,offset wtmsg ;point to data
0212  E8 0237 R                      call display        ;print it
0215  E9 0035 R                      jmp clock           ;wait until 1
                                                         ;hour has passed
                                                         ;then repeat
                                     ;
                                     ; 'Abort' Routine
                                     ;
0218  BB 006F R              abort:  mov bx,offset abmsg ;point to data
021B  C6 06 0027 R 1A                mov fcharn,26       ;store # of chars
0220  E8 0237 R                      call display        ;output message
0223  CB                             ret                 ;return to dos
                                     ;
                                     ; 'Write' subroutine
                                     ;
0224                         write   proc near
0224  BA 0000 R                      mov dx,offset fcb
0227  B4 15                          mov ah,15h          ;'write seq. file'
0229  CD 21                          int 21h             ;invoke DOS fcn.
```

```
022B  3C 00                     cmp al,0            ;was write OK?
022D  75 01                     jnz error           ;branch if not
022F  C3                        ret                 ;return from subr.

                        write   endp
                        ;
                        ; 'Error' subroutine
                        ;
0230                    error   proc near
0230  BB 0089 R                 mov bx,offset errmsg
0233  E8 0237 R                 call display        ;display message
0236  C3                        ret                 ;return to DOS
                        error   endp
                        ;
                        ; 'Display' subroutine
                        ;
0237                    display proc near
0237  8A 0E 0027 R              mov cl,fcharn       ;# of characters
                                                    ;to display
023B  B5 00                     mov ch,0            ;clear HB of reg
023D  8A 07             displ:  mov al,[bx]         ;get next char.
023F  E8 0250 R                 call dispchar
0242  43                        inc bx
0243  E2 F8                     loop displ          ;do it til done
0245  B0 0D                     mov al,0dh          ;CR
0247  E8 0250 R                 call dispchar
024A  B0 0A                     mov al,0ah          ;LF
024C  E8 0250 R                 call dispchar
024F  C3                        ret                 ;return to caller
                        display endp
                        ;
                        ; 'Dispchar' subroutine
                        ;
0250                    dispchar proc near
0250  53                        push bx             ;save BX
0251  BB 0000                   mov bx,0
0254  B4 0E                     mov ah,14           ;write to display
0256  CD 10                     int 10h             ;invoke DOS fcn.
0258  5B                        pop bx              ;restore BX
0259  C3                        ret                 ;return to caller
                        dispchar endp
                        ;
                        ; Wrap things up
                        ;
                        main    endp
025A                    code    ends
                                end main
```

Segments and Groups:

N a m e	Size	Align	Combine	Class
CODE	025A	PARA	PUBLIC	'CODE'
DATA	34A1	PARA	PUBLIC	'DATA'
STACK	0064	PARA	STACK	'STACK'

Symbols:

N a m e	Type	Value	Attr
ABMSG	L	BYTE	006F DATA
ABORT	L	NEAR	0218 CODE
AHEAD	L	NEAR	0195 CODE
BACKFIL	L	NEAR	0130 CODE
CLOCK	L	NEAR	0035 CODE
DECIMAL	L	BYTE	3497 DATA Length = 0004
DISP1	L	NEAR	023D CODE
DISPCHAR	N	PROC	0250 CODE Length = 000A
DISPLAY	N	PROC	0237 CODE Length = 0019
DTA	L	BYTE	0024 DATA
ERRMSG	L	BYTE	0089 DATA
ERROR	N	PROC	0230 CODE Length = 0007
FCB	L	BYTE	0000 DATA Length = 0024
FCHARN	L	BYTE	0027 DATA
FINE1	L	NEAR	0052 CODE
FINE2	L	NEAR	0059 CODE
FLMSG	L	BYTE	0094 DATA
FRSTPT	L	NEAR	0041 CODE
FULL	L	NEAR	013D CODE
HERE	L	NEAR	0179 CODE
INIT	L	NEAR	014B CODE
JBACK	L	NEAR	001D CODE
LOOKUP1	L	BYTE	347F DATA
LOOKUP2	L	BYTE	348B DATA
MAIN	F	PROC	0000 CODE Length = 025A
NAMNUM1	L	BYTE	0025 DATA
NAMNUM2	L	BYTE	0026 DATA
NONAME	L	NEAR	0148 CODE
NUMLOOP	L	WORD	349F DATA
NUMSAMP	L	WORD	349D DATA

```
OUT  . . . . . .  N PROC0163CODELength = 0035

RDNSTRT  . . . . . .  L NEAR00A7CODE
RESTORE  . . . . . .  L NEAR012CCODE

SAMPLE . . . . . .  L NEAR0061CODE
SEND . . . . . .  L NEAR01DCCODE
SPMSG  . . . . . .  L BYTE 0028DATA
SSNUM1 . . . . . .  L BYTE 00B1DATA
SSNUM2 . . . . . .  L BYTE 00B2DATA
STORAGE  . . . . . .  L WORD 00B3DATALength = 19E6

TEMPH  . . . . . .  L BYTE 349BDATA
TEMPL  . . . . . .  L BYTE 349CDATA

UP . . . . . .  L NEAR0203CODE

WAIT . . . . . .  L NEAR003BCODE
WRITE  . . . . . .  N PROC0224CODELength = 000C
WTMSG  . . . . . .  L BYTE 0065DATA

ZERO . . . . . .  L NEAR011BCODE


    390 Source  Lines
    390 Total   Lines
     70 Symbols

  48070 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```

Appendix B

DFT SOFTWARE

```
//  JOB ',,,T=60,I=20,L=30'
// EXEC FORTXCLG,PARM='NOXREF,NOLIST,NOMAP',
//              USERLIB='SYS4.DRIVER.DYLOAD'
//SYSIN DD *
C
C     THE MAIN PROGRAM BLOCK WHICH CREATES THE DATA, CALLS
C     THE DFT PROGRAM AND OUTPUTS THE RESULTS TO THE PRINTER.
C
C     MODIFY DATA AT LINE 220 TO GET TD OR FD PLOT.
C
      REAL X(20000), A(10000), B(10000), CN(10000), FREQ(20000),
     *     TIME(20000) ,DELT, SFREQ, DIG
      INTEGER LOOP, NPTS, NPLOTS, P, OFFSET, TEMP
      LOGICAL WINDOW, TD, FILTER
      READ *, NPTS, WINDOW, TD, FILTER
      DELT = 5.0276520865E-6
      DO 10 LOOP = 1,NPTS
         READ *, DIG
         X(LOOP) = (DIG - 2048.0) / 411
         TIME(LOOP) = DELT * (LOOP - 1)
 10   CONTINUE
      IF (.NOT. WINDOW) GO TO 40
      DO 30 LOOP = 1,NPTS
         X(LOOP) = X(LOOP) * HAMM(LOOP, NPTS)
 30   CONTINUE
 40   NPLOTS = 0
      CALL PLOTS(0,0,0)
      CALL AREA(8.25, 10.75)
      CALL PLOTS(IBUF, 1)
      IF (TD) GO TO 35
      CALL DFT(X,A,B,FREQ,CN,DELT,NPTS)
      P = NPTS / 2 + 1
      PRINT 90
 90   FORMAT('1',2X,'FREQUENCY',4X,'AMPLITUDE',6X,
     *   'REAL PART OF X(K)',X,'IMAG. PART OF X(K)')
      PRINT 99
 99   FORMAT (' ',' ----------------------------------------',
     * T41,'--------------------------')
      DO 20 LOOP = 1,P
         PRINT 95, FREQ(LOOP), CN(LOOP), A(LOOP), B(LOOP)
 95      FORMAT (' ',2X,F8.1,5X,F11.6,8X,F9.4,9X,F9.4)
 20   CONTINUE
      SFREQ = 1.0 / DELT
      PRINT 98, SFREQ
 98   FORMAT ('-',9X,'SAMPLING FREQUENCY = ',F8.1,' HERTZ')
      GO TO 36
 35   PRINT 66
 66   FORMAT ('1',3X,'NUMBER',9X,'TIME (SEC.)',11X,'AMPL. (VOLTS)')
      PRINT 67
 67   FORMAT (' ',' ----------------------------'
     *       ,T32,'----------------------')
      DO 13 LOOP = 1,NPTS
         PRINT 96, LOOP, TIME(LOOP), X(LOOP)
 13   CONTINUE
 96   FORMAT (' ',2X,I5,10X,F12.6,10X,F12.6)
      GO TO 15
 36   IF (.NOT. FILTER) GO TO 34
      OFFSET = 20
      NPTS = NPTS - OFFSET
      DO 37 LOOP = 1,NPTS
         TEMP = LOOP + OFFSET
         CN(LOOP) = CN(TEMP)
         FREQ(LOOP) = FREQ(TEMP)
 37   CONTINUE
```

```
   34  CALL PLOT3(FREQ,CN,P,'     FOURIER COEFFICIENTS',
      *  'FREQUENCY','AMPLITUDE',24,9,9,.TRUE.,NPLOTS,1)
       GO TO 16
   15  CALL PLOT3(TIME,X,NPTS,'          WAVEFORM','TIME',
      *  'AMPLITUDE',17,4,9,.FALSE.,NPLOTS,1)
   16  CALL PLOT(0.0,0.0,9999)
       STOP
       END
C
C      THIS SUBPROGRAM FOR A 2ND ORDER GOERTZEL DFT ALGORITHM
C
       SUBROUTINE DFT(X, A, B, FREQ, CN, DELT, N)
       REAL X(20000), A(10000), B(10000), FREQ(20000), CN(10000),
      *     Q, C, S, CC, A1, A2, B1, B2, T, DELT
       INTEGER N, J, I, P
       COMPLEX HOLD
       P = N / 2 + 1
       Q = 6.283185307 / N
       DO 20 J = 1,P
          C = COS(Q * (J - 1))
          S = SIN(Q * (J - 1))
          CC = 2.0 * C
          A2 = 0.0
          B2 = 0.0
          A1 = X(1)
          B1 = 0.0
          DO 30 I = 2,N
             T = A1
             A1 = CC * A1 - A2 + X(I)
             A2 = T
             T = B1
             B1 = CC * B1 - B2
             B2 = T
   30     CONTINUE
          A(J) = 2.0 * (C * A1 - A2 - S * B1) / N
          B(J) = 2.0 * (C * B1 - B2 + S * A1) / N
          FREQ(J) = (1.0 / (N * DELT)) * (J - 1)
          HOLD = CMPLX(A(J), B(J))
          CN(J) = CABS(HOLD)
   20  CONTINUE
       RETURN
       END
C
C      FUNCTION TO EVALUATE THE HAMMING WINDOW FOR SAMPLING
C
       REAL FUNCTION HAMM(COUNT, N)
       INTEGER COUNT, N
       REAL Q
       Q = 6.28318507 * COUNT / N
       HAMM = 0.08 + 0.46 * (1.0 - COS(Q))
       RETURN
       END
C
C*************************************************************************
C                                                                        *
C PURPOSE: PERFORM PLOTTING OF X VS. Y USING CALCOMP, EITHER AS         *
C          VERTICAL LINES OR AS A CONTINUOUS CURVE.                     *
C VERSION: 85 MAR 11,  STARTED: 85 MAR 11                               *
C FILE: FPLOT3                                                          *
C                                                                        *
C*************************************************************************
       SUBROUTINE PLOT3(X,Y,NPTS,TITLE,XLABEL,YLABEL,NTITLE,XNCHAR,
      &YNCHAR,DSCPLT,NPLOTS,LOOP)
       INTEGER XNCHAR,YNCHAR,IBUF(1),LOOP
```

```
      INTEGER TITLE(1),XLABEL(1),YLABEL(1)
      LOGICAL DSCPLT
      COMMON  /SCAL/  XNP1,XNP2,YNP1,YNP2
      REAL       XNP1,XNP2,YNP1,YNP2
      REAL X(20000),Y(20000)
      XAXLEN=4.0
      YAXLEN=6.0
      ORG=0
      XPAGE=2.0
      YPAGE=2.0
      IF(LOOP.GT.1)GOTO 40
      CALL SCALE1(X,XAXLEN,NPTS)
      CALL SCALE1(Y,YAXLEN,NPTS)
      XNP1=X(NPTS+1)
      XNP2=X(NPTS+2)
      YNP1=Y(NPTS+1)
      YNP2=Y(NPTS+2)
      GOTO 50
40    CONTINUE
       X(NPTS+1)=XNP1
       X(NPTS+2)=XNP2
       Y(NPTS+1)=YNP1
       Y(NPTS+2)=YNP2
50     CONTINUE
      XNCHAR=-IABS(XNCHAR)
      IF(.NOT.DSCPLT)GO TO 20
      YXORG= 0.0
      CALL PLOT(XPAGE,YPAGE,-3)
      CALL LINES(X,Y,NPTS)
      GO TO 30
20    YXORG=0
      IF (LOOP.EQ.2) GO TO 70
      CALL PLOT(XPAGE,YPAGE,-3)
      CALL LINE(X,Y,NPTS,1,0,0)
30    CALL AXIS(ORG,ORG,XLABEL,XNCHAR,XAXLEN,0,X(NPTS+1),X(NPTS+2))
      CALL AXIS(YXORG,ORG,YLABEL,YNCHAR,YAXLEN,90.0,Y(NPTS+1),
     &Y(NPTS+2))
      CALL SYMBOL(0.5,YAXLEN+0.5,0.14,TITLE,0.,NTITLE)
C     CALL SYMBOL(1.75,-1.0,0.14,'FIG.-',0.,5)
      GO TO 71
 70   CALL LINE(X,Y,NPTS,1,1,3)
 71   CONTINUE
      RETURN
      END
C
C*************************************************************************
C PURPOSE: S/R TO PERFORM SPECTRAL LINE PLOTTING USING CALCOMP          *
C VERSION: 85 MAR 11,  STARTED: 85 MAR 11                               *
C FILE: FLINES                                                          *
C                                                                       *
C*************************************************************************
      SUBROUTINE LINES(X,Y,NPTS)
      DATA ILABX/'X'/,ILABY/'Y'/
      REAL X(20000),Y(20000)
      X0=X(NPTS+1)
      DX=X(NPTS+2)
      Y0=Y(NPTS+1)
      DY=Y(NPTS+2)
      IF(DX.EQ.0) WRITE(6,20) ILABX, ILABX
      IF(DY.EQ.0) WRITE(6,20) ILABY, ILABY
      DO 10 I=1,NPTS
      XI=(X(I)-X0)/DX
      YI=(Y(I)-Y0)/DY
      CALL PLOT(XI,0,3)
```

```
10     CALL PLOT(XI,YI,2)
       RETURN
20     FORMAT('0S/R LINES: D',A1,' IS 0, ',A1,' VECTOR MAY BE CONSTANT')
       END
C
C*************************************************************************
C                                                                        *
C PURPOSE: S/R TO SCALE A VECTOR TO FIT A GIVEN AXIS LENGTH.            *
C          SCALING DATA IS PLACED IN LOCATIONS NPTS+1 AND NPTS+2        *
C          OF THE X VECTOR                                              *
C VERSION: 85 MAR 11,  STARTED: 85 MAR 11                               *
C FILE: FSCALE1                                                         *
C                                                                        *
C*************************************************************************
       SUBROUTINE SCALE1(X,AXLEN,NPTS)
       REAL X(20000)
       XMAX=0
       XMIN=0
       DO 10 I=1,NPTS
       XMAX=AMAX1(XMAX,X(I))
10     XMIN=AMIN1(XMIN,X(I))
       X(NPTS+1)=XMIN
       X(NPTS+2)=(XMAX-XMIN)/AXLEN
       RETURN
       END
//GO.SYSIN DD *
C
6630,.FALSE.,.FALSE.,.TRUE.
C  6630    = # of data points (up to 20000 allowed)
C .false. = Hamming window wanted? (else rectangular)
C .false. = Time domain plot? (else freq. domain)
C .true.  = HP filter wanted (first 25 coeffs. will be set to 0)
C
/*
//GO.FT22F001 DD SYSOUT=A
//GO.FT23F001 DD DSN=SYS4.EPIC.PARMS,DISP=SHR,LABEL=(,,,IN)
//GO.FT24F001 DD DSN=&&FT01F001,SPACE=(6136,300),
// DCB=(BLKSIZE=6136,DSORG=DA,OPTCD=C,RECFM=F),
// DISP=(,PASS),UNIT=SYSDA
//GO.PLOTLIB  DD DSN=SYS4.DRIVER.XEROX,DISP=SHR
//GO.FT02F001 DD SYSOUT=(F,,BALT),COPIES=2
// EXEC XPLOT
```

Appendix C

CARRIER CURRENT SOFTWARE

(3 of Last 4 Voting Scheme Employed)

```
8000                    pia1            equ $8000
8800                    pia2            equ $8800
0000                    poll            equ $00
0001                    timeout         equ $01
0002                    recep           equ $02
0003                    trans           equ $03
0004                    dips            equ $04
0005                    rcop            equ $05
0006                    tcop            equ $06
0007                    mux             equ $07
0008                    index           equ $08
000A                    blksig          equ $0A

FFFE                                    org $FFFE
FFFE F8 00                              fcb $F8,$00     ;sets reset vector to
                        *                               ;start of EPROM

F800                                    org $F800       ;start of 2K EPROM
F800 8E 00 7F                           lds #$7F        ;set stack pointer
                        *
                        * PIA Initialization
                        *
F803 4F                                 clra            ;clear acc. A
F804 B7 80 01                           staa pia1+1     ;set A to direction
F807 B7 80 03                           staa pia1+3     ;set B to direction
F80A B7 88 01                           staa pia2+1     ;set A to direction
F80D B7 88 03                           staa pia2+3     ;set B to direction
F810 86 40                              ldaa #$40       ;proper I/O selection
F812 B7 80 00                           staa pia1       ;set dir. of reg. A
F815 86 FF                              ldaa #$FF       ;proper I/O selection
F817 B7 80 02                           staa pia1+2     ;set dir. of reg. B
F81A C6 80                              ldab #$80       ;proper I/O selection
F81C F7 88 00                           stab pia2       ;set dir. of reg. A
F81F B7 88 02                           staa pia2+2     ;set dir. of reg. B

F822 86 04                              ldaa #$04       ;set bit 2
F824 B7 80 01                           staa pia1+1     ;set A back to data
F827 B7 80 03                           staa pia1+3     ;set B back to data
F82A B7 88 01                           staa pia2+1     ;set A back to data
F82D B7 88 03                           staa pia2+3     ;set B back to data

F830 86 40                              ldaa #$40       ;open transmitter
F832 B7 80 00                           staa pia1       ;switch
F835 86 80                              ldaa #$80       ;turn off programming
F837 B7 88 00                           staa pia2       ;error light
F83A 86 80                              ldaa #$80       ;turn off block light
F83C B7 88 02                           staa pia2+2     ;close all BP relays
                        *
                        * Check for programming errors
                        * (more than 1 selection or no selection)
                        *
F83F B6 80 00                           ldaa pia1       ;get A data
F842 84 3F                              anda #$3F       ;just want dipsw. info
F844 97 04                              staa dips       ;keep a copy in RAM

F846 CE 00 06                           ldx #$0006      ;set loop counter to 6
F849 96 04                              ldaa dips       ;get dip info
F84B 5F                                 clrb            ;clear the sum register
F84C 0C                                 clc             ;clear the carry
F84D 46                 rotate          rora            ;rotate right
```

```
F84E 24 01                      bcc clear       ;branch if sw. not set
F850 5C                         incb            ;inc. acc. B
                        *                       ;(B will contain the #
                        *                       ;'ON' sw.s when done)
F851 09                 clear   dex             ;decrement counter
F852 26 F9                      bne rotate      ;do 6 times
F854 C1 01                      cmpb #$01       ;exactly 1 switch set
F856 27 05                      beq memory      ;yes, go on
F858 4F                         clra            ;otherwise, error
F859 B7 88 00                   staa pia2       ;turn on p.e. light
F85C 3E                         wai             ;and halt program
                        *
                        * Initialize the memory
                        *
F85D 96 04              memory  ldaa dips
F85F 81 01                      cmpa #$01
F861 26 17                      bne two
F863 C6 FF                      ldab #$FF
F865 D7 02                      stab recep
F867 7F 00 03                   clr trans
F86A C6 04                      ldab #$04
F86C D7 07                      stab mux
F86E 86 F8                      ldaa #$F8       ;all relays open
F870 B7 88 02                   staa pia2+2     ;block light off
F873 4F                         clra            ;pick highest possible
F874 B7 80 02                   staa pia1+2     ;transmission freq.
F877 7E F8 FE                   jmp timer
F87A 81 02              two     cmpa #$02
F87C 26 19                      bne three
F87E C6 01                      ldab #$01
F880 D7 03                      stab trans
F882 C6 04                      ldab #$04
F884 D7 02                      stab recep
F886 C6 03                      ldab #$03
F888 D7 07                      stab mux
F88A 86 F0                      ldaa #$F0       ;relay 0 closed
F88C B7 88 02                   staa pia2+2
F88F 86 4E                      ldaa #$4E       ;pick 95 kHz
F891 B7 80 02                   staa pia1+2     ;transmission freq.
F894 7E F8 FE                   jmp timer
F897 81 04              three   cmpa #$04
F899 26 19                      bne four
F89B C6 02                      ldab #$02
F89D D7 03                      stab trans
F89F C6 03                      ldab #$03
F8A1 D7 02                      stab recep
F8A3 C6 02                      ldab #$02
F8A5 D7 07                      stab mux
F8A7 86 E8                      ldaa #$E8       ;relay 1 closed
F8A9 B7 88 02                   staa pia2+2
F8AC 86 72                      ldaa #$72       ;pick 87 kHz
F8AE B7 80 02                   staa pia1+2     ;transmission freq.
F8B1 7E F8 FE                   jmp timer
F8B4 81 08              four    cmpa #$08
F8B6 26 19                      bne five
F8B8 C6 03                      ldab #$03
F8BA D7 03                      stab trans
F8BC C6 02                      ldab #$02
F8BE D7 02                      stab recep
F8C0 C6 01                      ldab #$01
```

```
F8C2 D7 07                          stab mux
F8C4 86 D8                          ldaa #$D8         ;relay 2 closed
F8C6 B7 88 02                       staa pia2+2
F8C9 86 97                          ldaa #$97         ;pick 79 kHz
F8CB B7 80 02                       staa pia1+2       ;transmission freq.
F8CE 7E F8 FE                       jmp timer
F8D1 81 10              five        cmpa #$10
F8D3 26 19                          bne six
F8D5 C6 04                          ldab #$04
F8D7 D7 03                          stab trans
F8D9 C6 01                          ldab #$01
F8DB D7 02                          stab recep
F8DD C6 00                          ldab #$00
F8DF D7 07                          stab mux
F8E1 86 C0                          ldaa #$C0         ;relay 3 closed
F8E3 B7 88 02                       staa pia2+2
F8E6 86 C0                          ldaa #$C0         ;pick 71 kHz
F8E8 B7 80 02                       staa pia1+2       ;transmission freq.
F8EB 7E F8 FE                       jmp timer
F8EE C6 FF              six         ldab #$FF
F8F0 D7 03                          stab trans
F8F2 7F 00 02                       clr recep
F8F5 C6 FF                          ldab #$FF         ;never receive
F8F7 D7 07                          stab mux
F8F9 86 ED                          ldaa #$ED         ;pick 63 kHz
F8FB B7 80 02                       staa pia1+2       ;transmission freq.
F8FE C6 0A              timer       ldab #$0A         ;set time-out at 10msec
F900 D7 01                          stab timeout      ;store it
F902 7F 00 0A                       clr blksig        ;clear out blk info.
F905 CE 10 00                       ldx #$1000        ;set delay counter
F908 09                 bounce      dex               ;allow for button
F909 26 FD                          bne bounce        ;bounce

                        *
                        * Fault detection loop
                        *
F90B B6 80 00           fault       ldaa pia1         ;get fault info
F90E 2A FB                          bpl fault         ;no fault, keep waiting
F910 BD F9 E0                       jsr delay1        ;wait, then look again
F913 B6 80 00                       ldaa pia1         ;chk for flt. again
F916 2A F3                          bpl fault         ;no fault, keep waiting
F918 B6 88 02                       ldaa pia2+2       ;get PIA 2, reg. B info
F91B 84 7F                          anda #$7F         ;set bit 7 low to
F91D B7 88 02                       staa pia2+2       ;block the relay

                        *
                        * Decide whether to transmit or receive first
                        * (Even goes first, odd second)
                        *
F920 96 04                          ldaa dips         ;get dipswitch info
F922 46                             rora              ;rotate right
F923 24 03                          bcc even1         ;if even branch
F925 7E F9 5C                       jmp rx            ;if odd then receive
F928 46                 even1       rora              ;bit 2 into carry
F929 46                             rora              ;bit 3 into carry
F92A 24 03                          bcc even2         ;if even then branch
F92C 7E F9 5C                       jmp rx            ;if odd, then receive
F92F 46                 even2       rora              ;bit 4 into carry
F930 46                             rora              ;bit 5 into carry
F931 24 03                          bcc tx            ;if even then transmit
```

```
F933 7E F9 5C                       jmp rx          ;if odd then receive
                                    *
                                    * Begin transmit routine
                                    *
F936 4F                   tx        clra            ;clear bit 6
F937 B7 80 00                       staa pia1       ;turn on trans switch
F93A 96 03                          ldaa trans      ;load trans counter
F93C 97 06                          staa tcop       ;store it temporarily

F93E CE 00 05             startt    ldx #$0005      ;set counter
F941 BD F9 E0             wait      jsr delay1      ;0.163 msec delay loop
F944 01                             nop
F945 01                             nop
F946 01                             nop
F947 01                             nop
F948 01                             nop
F949 01                             nop
F94A 01                             nop
F94B 01                             nop             ;8 nops
F94C 09                             dex             ;store it temporarily
F94D 26 F2                          bne wait        ;wait 1 msec total
F94F 7A 00 01                       dec timeout     ;dec timer
F952 26 03                          bne dcount      ;check for timeout
F954 7E F9 B6                       jmp unblok      ;quit if timeout
F957 7A 00 06             dcount    dec tcop        ;dec. trans counter
F95A 26 E2                          bne startt      ;loop back if not done
                                    *
                                    * Begin receive routine
                                    *
F95C 86 40                rx        ldaa #$40       ;set bit 6
F95E B7 80 00                       staa pia1       ;turn off trans. switch
F961 96 02                          ldaa recep      ;load recep. counter
F963 97 05                          staa rcop       ;temporary storage
F965 7F 00 00                       clr poll        ;clear poll area,
                                    *               ;fill with NO votes
F968 B6 88 02                       ldaa pia2+2     ;get pia2 'B' data
F96B 84 F8                          anda #$F8       ;zero lowest 3 bits
F96D 9A 07                          oraa mux        ;sets lowest 3 bits the
                                    *               ;way they need to be
F96F B7 88 02                       staa pia2+2     ;save settings
F972 CE 00 04             startr    ldx #$0004      ;4 poll counter init.
F975 BD F9 E0                       jsr delay1      ;routine to allow for
                                    *               ;PLL lock-in time
                                    *
                                    * Polling Routine
                                    *
F978 B6 88 00             tonpoll   ldaa pia2       ;get pia2 data
F97B 84 40                          anda #$40       ;just want bit 6
F97D 49                             rola            ;bit 6 to bit 7
F97E 49                             rola            ;bit 7 to carry
F97F 79 00 00                       rol poll        ;put into LSB of
                                    *               ;the poll register

                                    *
                                    * The democratic routine (voting time)
                                    *
F982 5F                   demo      clrb            ;init. the vote counter
F983 96 00                          ldaa poll       ;get the ballots
F985 0C                             clc             ;clear the carry
F986 46                             rora            ;rotate right
```

```
F987 24 01                          bcc ballot2     ;if a NO vote, branch
F989 5C                             incb            ;add 1 to vote counter
F98A 46                 ballot2     rora
F98B 24 01                          bcc ballot3
F98D 5C                             incb
F98E 46                 ballot3     rora
F98F 24 01                          bcc ballot4
F991 5C                             incb
F992 46                 ballot4     rora
F993 24 01                          bcc decide
F995 5C                             incb            ;B contains the number
                        *                           ;of YES votes
F996 C1 02              decide      cmpb #$02       ;compare to 2
                        *                           ;3 of 4 voting scheme
F998 23 06                          bls plcntr      ;branch if B <= 2
F99A 86 80                          ldaa #$80       ;indic. block received
F99C 97 0A                          staa blksig     ;store it
F99E 20 06                          bra dectim      ;dec timer next
F9A0 BD F9 EB           plcntr      jsr delay2      ;0.123 msec delay
F9A3 09                             dex             ;dec. poll counter
F9A4 26 D2                          bne tonpoll     ;continue polling
F9A6 7A 00 01           dectim      dec timeout     ;decrement timer
F9A9 26 03                          bne noto        ;branch if no time-out
F9AB 7E F9 B6                       jmp unblok      ;quit if time-out
F9AE 7A 00 05           noto        dec rcop        ;dec. receive counter
F9B1 26 BF                          bne startr      ;loop back if not done
F9B3 7E F9 36                       jmp tx          ;if done, transmit
                        *
                        * Routine to unblock relay and wait for fault to clear
                        *
F9B6 86 40              unblok      ldaa #$40       ;make sure trans.
F9B8 B7 80 00                       staa pia1       ;switch is off
F9BB 96 0A                          ldaa blksig     ;check if blk rec'd.
F9BD 2B 08                          bmi done1       ;if yes, don't unblock
F9BF B6 88 02                       ldaa pia2+2     ;get B data
F9C2 8A 80                          oraa #$80       ;set bit 7 to unblock
                        *                           ;the relay
F9C4 B7 88 02                       staa pia2+2     ;resave settings
F9C7 CE 30 00           done1       ldx #$3000      ;set counter for
F9CA BD F9 E0           holdlite    jsr delay1      ;delay of 2.5 sec.
F9CD 09                             dex             ;to allow viewing
F9CE 26 FA                          bne holdlite    ;of block light
F9D0 B6 88 02                       ldaa pia2+2     ;make sure that
F9D3 8A 80                          oraa #$80       ;the block light
F9D5 B7 88 02                       staa pia2+2     ;is OFF
F9D8 B6 80 00           done2       ldaa pia1       ;get fault info.
F9DB 2B FB                          bmi done2       ;if fault persists
                        *                           ;then continue waiting
F9DD 7E F8 FE                       jmp timer       ;wait for another flt.

                        *
                        * 0.163 msec. delay routine
                        *
F9E0 DF 08              delay1      stx index       ;store index reg.
F9E2 CE 00 12                       ldx #$12        ;18 itts. of loop
F9E5 09                 decr1       dex
F9E6 26 FD                          bne decr1
F9E8 DE 08                          ldx index       ;retrieve index reg.
F9EA 39                             rts             ;return
```

```
                        *
                        *  0.123 msec. delay routine
                        *
F9EB DF 08              delay2          stx index       ;store index reg.
F9ED CE 00 0D                           ldx #$0D        ;13 itts. of loop
F9F0 09                 decr2           dex
F9F1 26 FD                              bne decr2
F9F3 DE 08                              ldx index       ;retrieve index reg.
F9F5 39                                 rts             ;return


Errors: 0
```

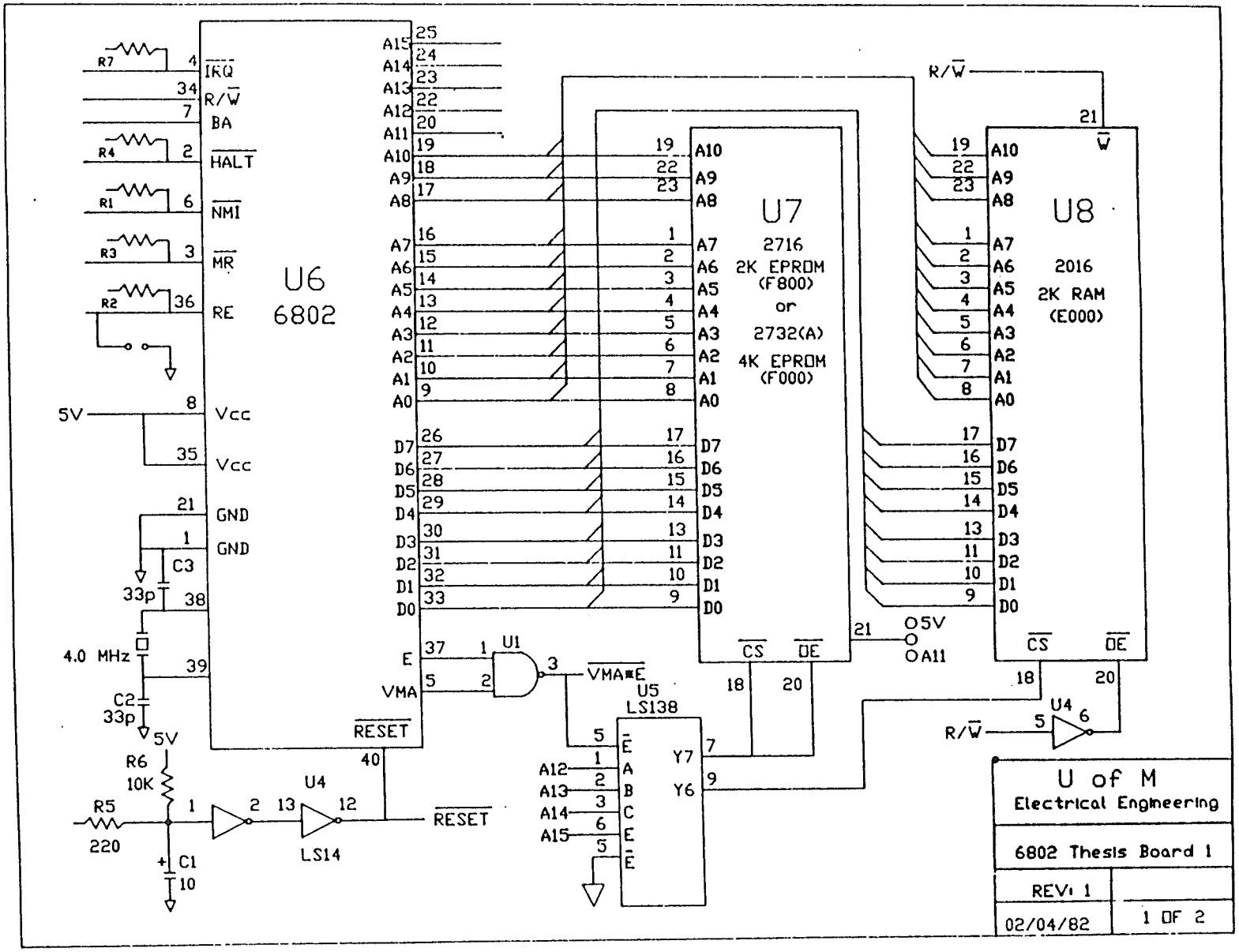
Appendix D

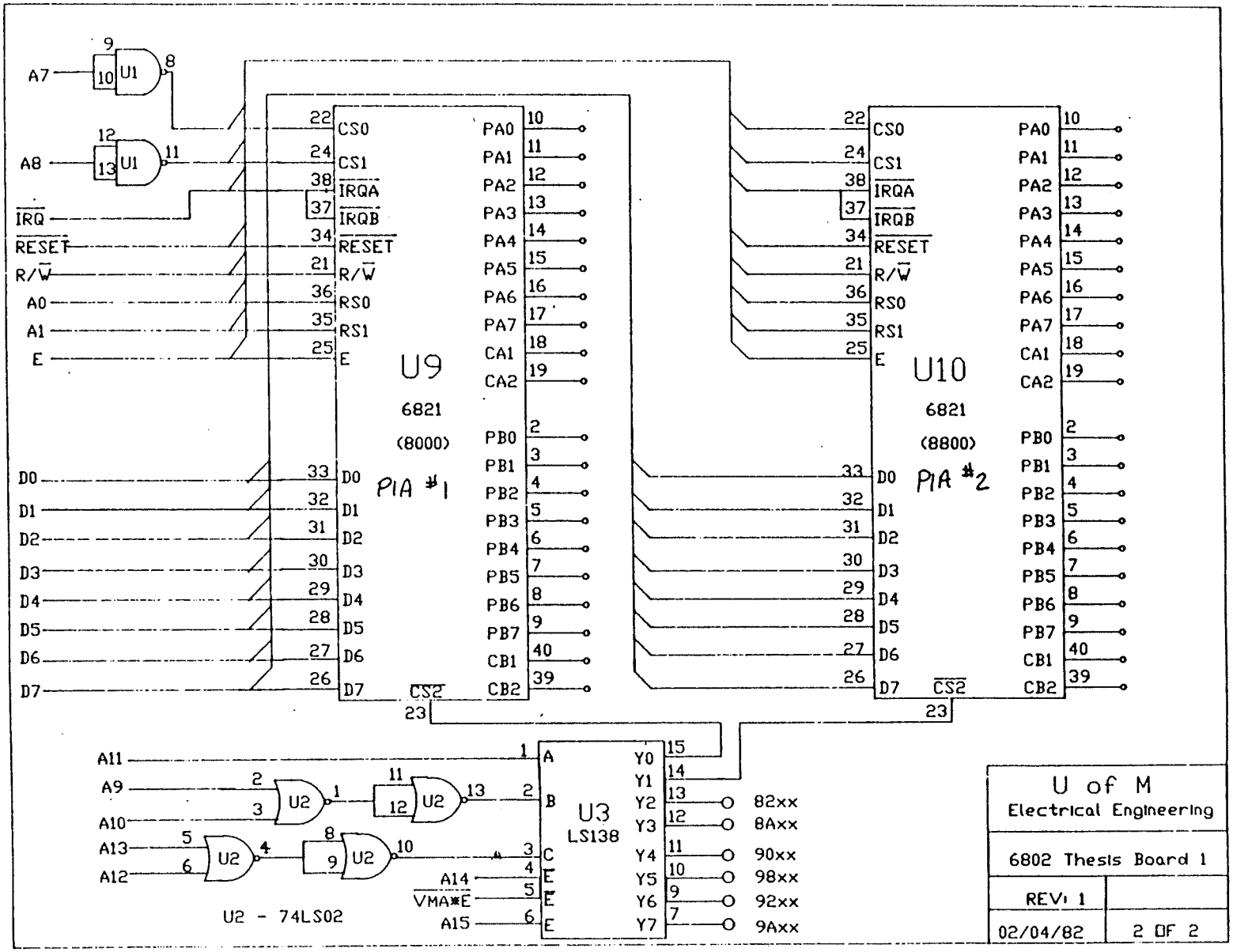
PIA PORT MAP

	Bit	Function	Direction	Pin
PIA #1:	A0	- Tone selection switch #1	INPUT	29
	A1	- Tone selection switch #2	INPUT	12
	A2	- Tone selection switch #3	INPUT	28
	A3	- Tone selection switch #4	INPUT	13
	A4	- Tone selection switch #5	INPUT	27
	A5	- Tone selection switch #6	INPUT	14
	A6	- Transmitter relay	OUTPUT	26
	A7	- Test button / Trip line	INPUT	15
	B0	- D/A Converter, bit 0	OUTPUT	25
	B1	- D/A Converter, bit 1	OUTPUT	16
	B2	- D/A Converter, bit 2	OUTPUT	24
	B3	- D/A Converter, bit 3	OUTPUT	17
	B4	- D/A Converter, bit 4	OUTPUT	23
	B5	- D/A Converter, bit 5	OUTPUT	18
	B6	- D/A Converter, bit 6	OUTPUT	22
	B7	- D/A Converter, bit 7	OUTPUT	19
PIA #2:	A0	- Not presently used	?	39
	A1	- Not presently used	?	02
	A2	- Not presently used	?	38
	A3	- Not presently used	?	03
	A4	- Not presently used	?	37
	A5	- Not presently used	?	04
	A6	- Data from MUX	INPUT	36
	A7	- Programming error light	OUTPUT	05
	B0	- Selection bit 0 to MUX	OUTPUT	35
	B1	- Selection bit 0 to MUX	OUTPUT	06
	B2	- Selection bit 0 to MUX	OUTPUT	34
	B3	- BP relay select line 0	OUTPUT	07
	B4	- BP relay select line 1	OUTPUT	33
	B5	- BP relay select line 2	OUTPUT	08
	B6	- BP relay select line 3	OUTPUT	32
	B7	- Block light	OUTPUT	09

Appendix E

MICROPROCESSOR BOARD INFORMATION

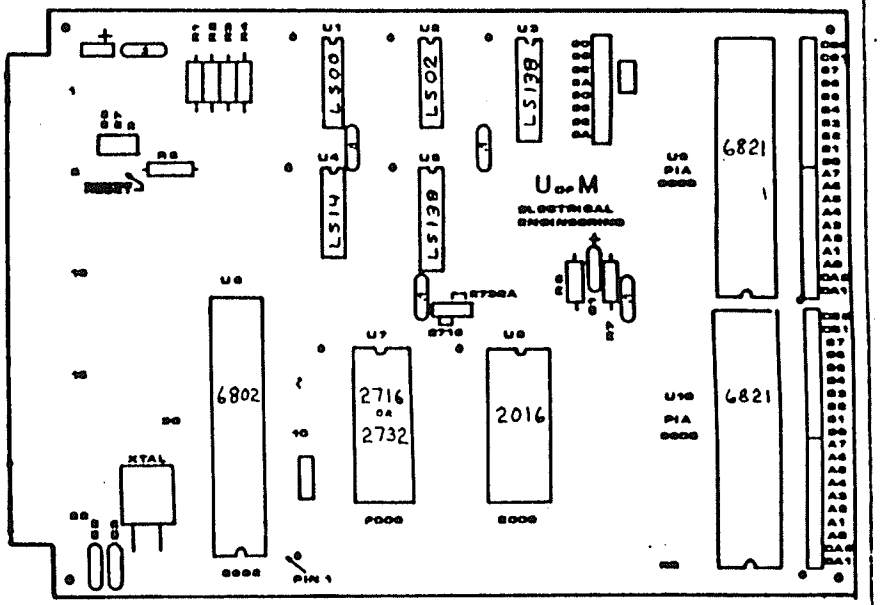

U6
6802
U7
2716
2K EPROM
(F800)
or
2732(A)
4K EPROM
(F000)
U8
2016
2K RAM
(E000)
U5
LS138
U4
LS14
U1
R7
R4
R1
R3
R2
R5
R6
10K
220
C1
10
C2
33p
C3
33p
4.0 MHz
5V
IRQ
R/W
BA
HALT
NMI
MR
RE
Vcc
Vcc
GND
GND
E
VMA
RESET
VMA=E
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
CS
OE
W
O5V
OA11
U of M
Electrical Engineering
6802 Thesis Board 1
REV: 1
02/04/82
1 OF 2


A7
A8
IRQ
RESET
R/W
A0
A1
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
U9
6821
(8000)
PIA #1
U10
6821
(8800)
PIA #2
CS0
CS1
IRQA
IRQB
RESET
R/W
RS0
RS1
E
D0
D1
D2
D3
D4
D5
D6
D7
CS2
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
CA1
CA2
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
CB1
CB2
A11
A9
A10
A13
A12
A14
VMA*E
A15
U2
U3
LS138
A
B
C
E
E
E
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
82xx
8Axx
90xx
98xx
92xx
9Axx
U2 - 74LS02
U of M
Electrical Engineering
6802 Thesis Board 1
REV: 1
02/04/82
2 OF 2

44 Pin Edge Connector

1	+5 Volts
2	$\overline{\text{HALT}}$
3	
4	MR
5	$\overline{\text{Reset}}$ I/P
6	$\overline{\text{Reset}}$ O/P
7	A13
8	A11
9	A9
10	A8
11	A7
12	A6
13	A5
14	A4
15	A3
16	A2
17	A1
18	A0
19	R/$\overline{\text{W}}$
20	$\overline{\text{NMI}}$
21	E (Clock)
22	Ground

23-A	+5 Volts
24-B	VMA
25-C	$\overline{\text{VMA}}$•E
26-D	
27-E	
28-F	BA
29-H	A12
30-J	A14
31-K	A10
32-L	A15
33-M	D7
34-N	D6
35-P	D5
36-R	D4
37-S	D3
38-T	D2
39-U	D1
40-V	D0
41-W	Vcc Stdby
42-X	Ram Enable
43-Y	$\overline{\text{IRQ}}$
44-Z	Ground



U of M Electrical Engineering	
6802 Thesis Board 1	
Rev : 1	
02/04/82	1 of 1

Appendix F

DATA SHEETS FOR SELECTED DEVICES

INTERNATIONAL RECTIFIER IOR CRYDOM

SERIES PVR
BOSFET®
PhotoVoltaic Relay

Two Pole, 180 mA
0-300V AC/DC

GENERAL DESCRIPTION:

The Crydom Photovoltaic Relay (PVR) is a two pole, normally open solid state replacement for electromechanical Reed Relays. It utilizes as an output switch a unique bidirectional (AC or DC) mosfet power IC termed a BOSFET.® The BOSFET is controlled by a photovoltaic generator of novel construction, which is energized by radiation from a dielectrically isolated Light Emitting Diode.

PVR FEATURES

The PVR overcomes the limitations of Reed Relays by offering the solid state advantages of long life, high operating speed, low pick-up power, bounce free operation, low thermal voltages and miniaturization. These advantages allow product improvement and design innovations in many applications such as process control, multiplexing, telecommunications, automatic test equipment, and data acquisition.

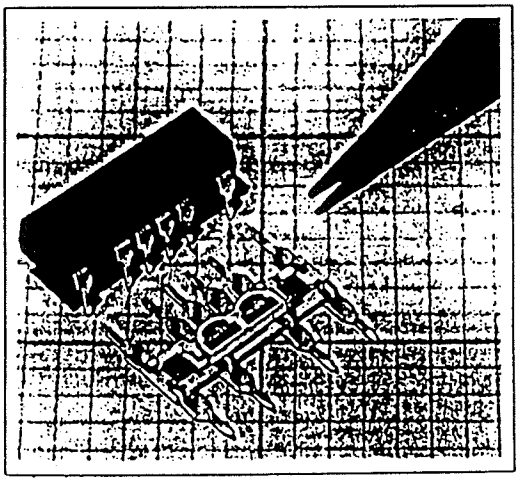
The PVR switches analog signals from thermocouple level to 300 volts peak AC or DC polarity. Signal frequencies into the RF range are easily controlled and switching rates up to 5 kHz are achievable. The extremely small thermally generated offset voltages allow increased measurement accuracies.

Unique silicon technology developed by International Rectifier forms the heart of the Crydom PVR. The monolithic BOSFET contains a bidirectional N channel power mosfet output structure. In addition, this power IC chip also has input circuitry for fast turn-off and gate protection functions. This section of the BOSFET chip utilizes both bipolar and MOS technology to form NPN transistors, P channel mosfets, resistors, diodes and capacitors.

The photovoltaic generator similarly utilizes a unique International Rectifier alloyed multi-junction structure. The excellent current conversion efficiency of this technique results in the very fast response of the Crydom PVR.

This advanced semiconductor technology has created a radically new control device. Designers can now develop analog switching systems to new standards of electrical performance and mechanical compactness.

- **BOSFET® Power IC**
- **10^{10} Operations**
- **250 μSec Operating Time**
- **0.2 μVolt Thermal Offset**
- **5 milliwatts Pick-Up Power**
- **1000V/μsec dv/dt**
- **Bounce Free**
- **TO-116 Pinout**
- **-40°C to 80°C**



Part Identification

Part No.	Operating Voltage	Off-state Resistance
PVR2300	0-200V AC/DC	10^8 ohms
PVR3300	0-300V AC/DC	
PVR3301		10^{10} ohms

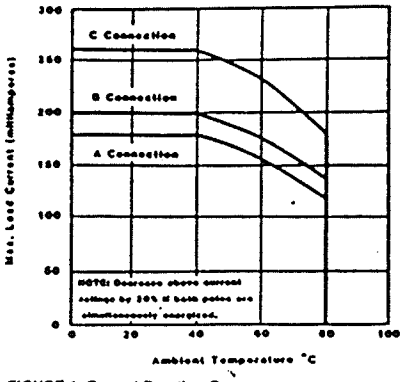


FIGURE 1. Current Derating Curve

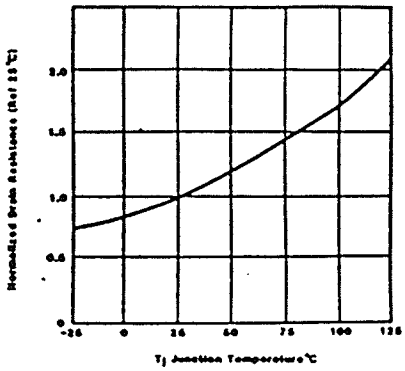


FIGURE 2. Normalized On-Resistance

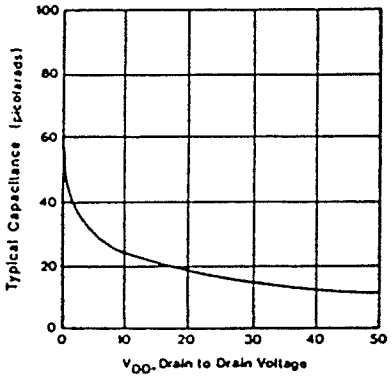


FIGURE 3. Typical Output Capacitance

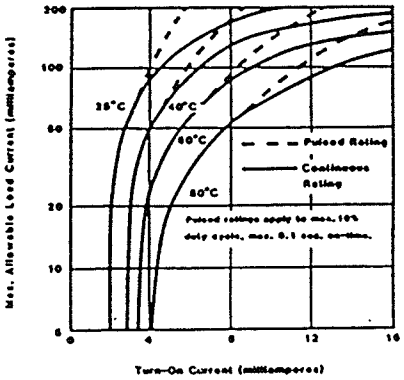


FIGURE 4. Minimum Control Current For Full Turn-on

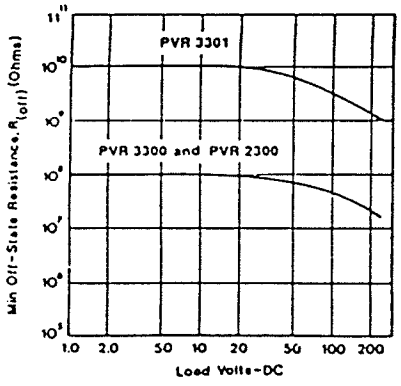


FIGURE 5. Off-State Resistance

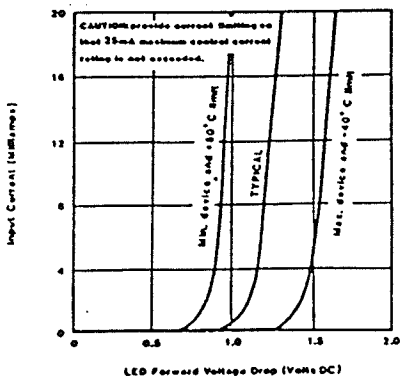


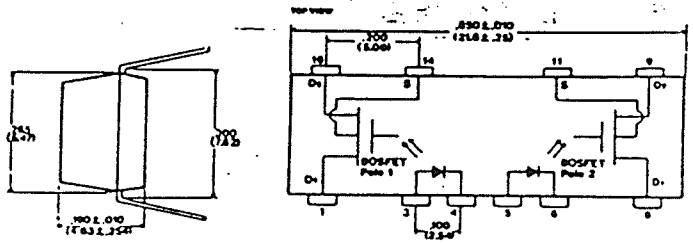
FIGURE 6. LED Input Characteristics

IOR CRYDOM BOSFET® PhotoVoltaic Relay

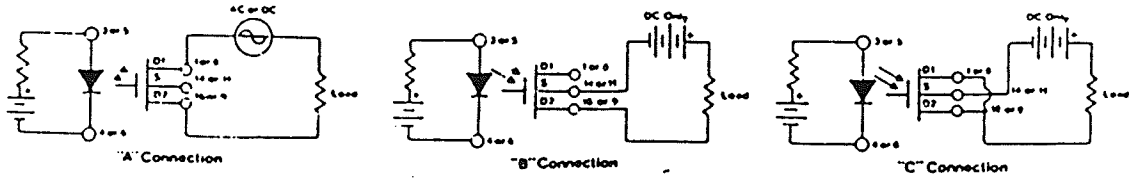
SPECIFICATIONS: (-40°C ≤ T_A ≤ 80°C unless otherwise specified)	PART NO. PVR2300	PVR3300	PVR3301	Units
Input Characteristics (See Fig. 4)				
Min. Allowable Control Current:				
For 20mA Continuous Load Current		2.0 @ 25°C		mA (DC)
For 100mA Continuous Load Current		5.0 @ 25°C		mA (DC)
For 20mA Continuous Load Current		5.0 @ 80°C		mA (DC)
Min. Turn-Off Current		10		μA (DC)
Min. Turn-Off Voltage		0.6		V (DC)
Control Current Range (Caution: Current limit input LED. See Fig. 6)		2.0 to 25		mA (DC)
Max. Reverse Voltage		-7.0		V (DC)
Response Time (See Fig. 7)				
Max. $T_{(on)}$ @ 8 mA Control, 100 mA load, 100 VDC, 25°C, 0 to 90%		250		microsec
Max. $T_{(off)}$ @ 8 mA Control, 100 mA load, 100 VDC, 25°C, 100% to 10%		50		microsec
Output Characteristics				
Operating Voltage Range	0 ±200	0 ±300		V (peak)
Max. Load Current 40°C (See Fig. 1)				
AC (See Wiring Diagram "A")		180		mA (peak)
DC (See Wiring Diagram "B")		200		mA (DC)
DC (See Wiring Diagram "C")		260		mA (DC)
Max. On-State Resistance 25°C (See Fig. 2) (50 mA load, 8 mA Control)				
AC Connection (See Wiring Diagram "A")		24		Ohms
DC Connection (See Wiring Diagram "B")		12		Ohms
DC Connection (See Wiring Diagram "C")		6		Ohms
Min. Off-State Resistance at 10 VDC, 25°C (see Fig. 5)	$1x10^8$		$1x10^{10}$	Ohms
Min. Off-State Resistance at 240 VDC, 25°C (see Fig. 5)	$0.2x10^8$		$1x10^9$	Ohms
Max. Thermal Offset Voltage, 5.0 mA Control		0.2		μ volts
Min Off-State dv/dt		1000		v/μs
Output Capacitance (See Fig. 3)		12		pf @ 50 VDC
General Characteristics				
Dielectric Strength-Input/Output		1500		V (RMS)
Insulation Resistance @ 500 VDC-Input/Output		10^9		Ohms
Max. Capacitance-Input/Output		1.0		pf
Ambient Temperature Range: Operating		-40 to 80		°C
Ambient Temperature Range: Storage		-40 to 100		°C

Mechanical Specifications
TO-116 Pinout

Dimensions in Inches (Millimeters)



Wiring Diagrams



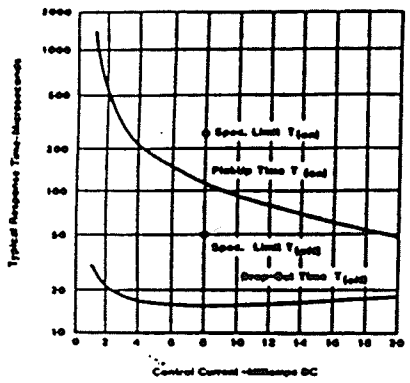


FIGURE 7. Typical Response Time

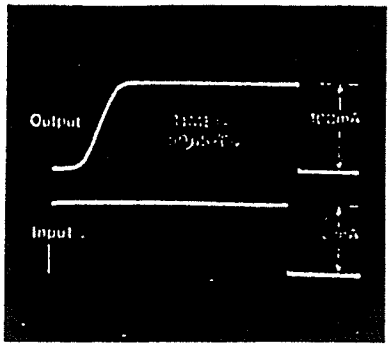


FIGURE 8. Switching Waveforms

IOR CRYDOM
A DIVISION OF INTERNATIONAL RECTIFIER

WORLD HEADQUARTERS: 1521 E.Grand Ave., El Segundo, CA 90245. (213) 322-4987. TWX: 910-348-6283

EUROPEAN HEADQUARTERS
IR Great Britain
Hurst Green, Oxted
Surrey RH8 9BB, England
Telephone: Oxted (08833) 3215
Telex: 95219

INTERNATIONAL SALES OFFICES
IR Canada
101 Bentley Street
Markham, Ontario L3R 3L1
Telephone: (416) 475-1897
Telex: 06966650

IR Germany
Savignystrasse 55
D-6000 Frankfurt/Main 1
Telephone: 0611-74-50-74
Telex: 04-13123

IR Italy
Via Liguria 49
10071 Borgaro, Torino
Telephone: (011) 470-14-84
Telex: 221257

Sales Offices, Agents and Distributors in Major Cities throughout the World.