A VERSATILE MICROPROCESSOR BASED DATA ACQUISITION SYSTEM FOR A BIOENGINEERING INSTRUMENTATION LABORATORY

by

PHILIP NOLAN KING

B. S., Kansas State University, 1976

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

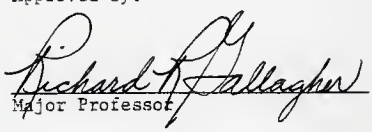
MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

Approved by:



Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The purpose of a data acquisition system is to preserve the information of interest in one or more analog signals for later evaluation. Currently this is generally done by sampling the signals at a sufficiently high rate to preserve the desired information, digitizing these samples, and storing the digital data [1,3,4].

Sometimes the signals of interest are not of the proper magnitude for digitizing. In biological and many electronics applications the signals are very low level and require amplification. This amplification can be done internal to the data acquisition system, or it can be done externally. It is frequently more convenient if the amplification is provided by the data acquisition system, as this relieves the user from having to provide amplifiers [1,3].

Several aspects of a data acquisition system should be user controllable. These include the order in which the data channels are sampled, and the rate at which the input signals are sampled [3]. While user access to system controls complicates the use of a data acquisition system, this access usually makes the system more beneficial to the user.

With consideration for these concepts, a versatile data acquisition system was designed and fabricated for use in a bioengineering instrumentation laboratory offered by the Departments of Electrical Engineering and Anatomy and Physiology at Kansas State University. Two major uses of the system include simplification of equipment setups for a number of physiological and instrumentation experiments, and preservation of data from these experiments for later review and evaluation.

In the course of a semester, physiological signal measurements are made of a number of biological systems. Many of the instrumentation experiments deal with understanding the properties of transducers and signal processing blocks used in the physiological experiments. Most of the experiments are more easily instrumented with the use of a data acquisition system of the type described here.

The instrument is adaptable for use by other teaching and research groups. It is expected that the electrical engineering faculty will find it particularly useful for demonstrations in the digital signal processing classes.

The characteristics of the laboratory signals, and other projected uses of the instrument, dictate some overall system specifications. Several input channels are required for several variables to be measured simultaneously. The maximum sampling rate must be at least 20,000 samples per second to allow cell membrane potential signals to be digitized. The input amplifiers should be capable of transducing one millivolt peak-to-peak signals without noise becoming noticeable, thereby allowing depressed electrocardiogram signals to be processed. A signal resolution of one percent was determined to be sufficient for the types of signals that the system will handle.

The final instrument specifications measure up well against the experimental requirements. The maximum sampling rate of 19,000 samples per second, with the full operating system software, can be increased to 80,000 samples per second with software modifications. The system provides up to eight channels capable of handling signals with a magnitude of 500 microvolts to 10 volts peak-to-peak under a wide range of common mode conditions.

II. DESIGN PHILOSOPHY

The original concept for the data acquisition system emerged from discussions among graduate students and engineering and life science faculty members. At this stage the final configuration was not well established, but the design philosophy and system goals had been determined.

Initially, three objectives were established as the guidelines for the design. These were:

1) to make the system maximally flexible, thereby allowing it to fulfill nearly any user need;
2) to make the system easy for the user to apply to his needs; and
3) to keep the implementation cost low.

A modular design, with allowances for easy upgrading, became a fourth guideline as the design progressed.

Since these design guidelines are generally conflicting, there were many compromises in the design stages to attempt to satisfy all of the criteria. The following paragraphs describe some of these trade-offs and how some of the criteria were satisfied. The hardware and software descriptions in Appendices II and III provide more insight into the design.

Probably the single most important element of the system in meeting the design criteria is the microprocessor used for system control. It allows considerable hardware control flexibility while maintaining simple system operation by providing prompts to the user for required inputs. The microprocessor also provides considerable cost savings when compared to using discrete logic for similar functions.

The hardware is entirely modular. This allows users with special requirements to build custom input units to meet their needs. The modular construction also provides for future upgrading of the system as higher performance electronics become available for reasonable prices.

The software is divided into general purpose subroutines which are used to interface with particular hardware blocks or provide a special function, and an operating system which ties the subroutines together. The user can use custom software simply by removing the memories containing the operating system, and substituting new memories containing the required operating system. The general purpose subroutines make the development of custom software reasonably easy.

III. HARDWARE DESCRIPTION

The hardware can be subdivided into the analog front end, the microprocessor, and the interfaces between the analog and digital sections. Each of these three groups can be further subdivided into small functional blocks. The various functional blocks and their interconnections are shown in Fig. III.1. Several of these blocks are described in detail below.

The 8080 microprocessor was considered for use in the system, largely because of its low cost, availability, and familiarity. Early in the design phase the 8080 was dropped from consideration, and a decision was made to use the then new Z-80 microprocessor because of its superior performance. The microprocessor board used is an S-100 bus compatible Z-80 board designed by M.S.P. Lucas, Department of Electrical Engineering, Kansas State University.

The use of a sample-and-hold amplifier on each input channel was considered. This would have allowed each of the channels to be sampled simultaneously, thereby relating accurately the status of each channnel at the time of the sample. Each of the channels would then be provided an analog-to-digital converter, or the input channels could be multiplexed into a single converter. This idea was rejected for economic reasons.

Each of the input channels is multiplexed to a single sample-and-hold amplifier and analog-to-digital converter. With proper software this allows the several input channels to be sampled at closely spaced intervals, thus simulating simultaneous sampling in applications requiring low data rates. This design also provides reasonably low cost.

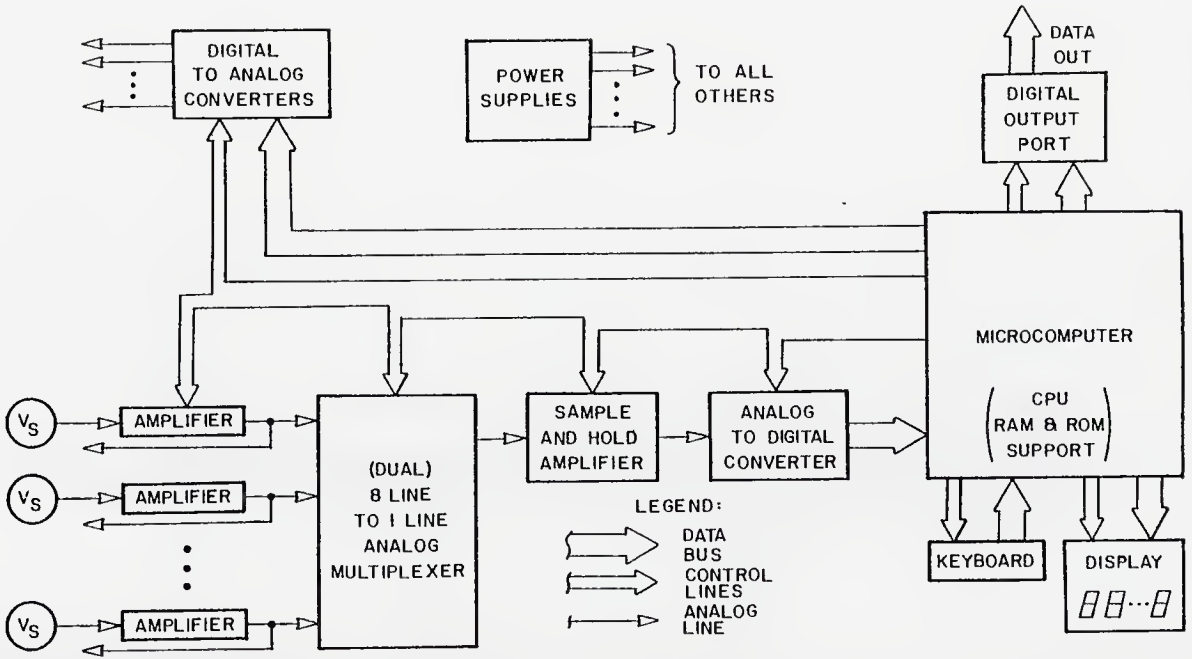


Figure III.1. System Hardware Block Diagram

The multiplexer and analog-to-digital converter are combined in a Burr-Brown MP-20 module. This unit was chosen for its ease of application, high performance, and low cost. The module consists of an eight-bit successive approximation analog-to-digital converter, a sixteen channel analog multiplexer, a buffer amplifier, and address decoding and timing logic. Bypassing the buffer amplifier with an inexpensive sample-and-hold amplifer allows the module to achieve sampling rates in excess of 80,000 samples per second.

Three types of amplifiers were designed for use with the system: a simple amplifier, an instrumentation amplifier, and an isolation amplifier. Each of the amplifiers is built as a plug-in module, allowing the user to configure the system to meet his needs, or to build custom plug-in units for special applications. While each type of amplifier has individual attributes, they all have many common features. The amplifiers provide a gain of 1 to 1000 in 1, 2, 5, and 10 steps. The output of each amplifier is buffered to the front panel for use in monitoring the signal or cascading amplifiers. All inputs and outputs are protected to at least 170 volts peak. The maximum signal input is ± 5 volts.

The three amplifier types use a common gain module. The module was designed for this purpose, and helps the amplifiers meet many of the system requirements. The unit provides gains of 1 to 1000 in 1, 2, 5, and 10 steps. The gain bandwidth product of the LM308A operational amplifier used in the gain module is 1 megahertz for small signal operation, and 25 kilohertz for large signal operation. In some applications this module could limit system performance; however, higher performance versions can be built and substituted in the amplifiers.

The simple amplifier consists of a gain module and a minimum of support hardware. It is intended for use with ground referenced signals. It can be used for input signals. By cascading amplifiers the simple amplifier can provide additional gain for another amplifier in the system.

The instrumentation amplifier, designed around an Analog Devices AD521 instrumentation amplifier, is intended to provide amplification of low level signals under conditions of up to five volts of common mode signal. The common mode rejection is a minimum of 70 decibels, and can be increased for low level signals. This amplifier is suitable for use with signal levels as low as 500 microvolts peak-to-peak. Depending on the configuration, the gain bandwidth product of the AD521 instrumentation amplifier matches or exceeds that of the gain module.

The isolation amplifier is designed around an Analog Devices 284J isolation amplifier module. It is intended for applications requiring the amplifier inputs and output to be electrically isolated, such as demanding medical uses. This amplifier can withstand common mode voltages to 150 volts peak continuously, or a pulse of 1500 volts for 10 milliseconds. The bandwidth of the 284J isolation amplifier is 1100 hertz for small signal, and 700 hertz for large signal operation. While this is a severely limited bandwidth, it is sufficient for many applications requiring this type of amplifier.

The system has the capability to support eight digital-to-analog converters, one per input channel. These converters are especially useful for recovering the signal which is being sent to the computer. This

signal can then be compared to the output of the amplifier on the same channel. This can be used to spot many types of sampling problems, such as aliasing.

With software modifications the digital-to-analog converters can be used to stimulate a device under test. In low frequency systems the converters can be used to measure the tested device's transfer function.

The digital-to-analog converters are built as plug-in modules similar to those used for the amplifiers. Each of the two digital-to-analog plug-in modules can support up to four converters. The converters are built on individual submodules, allowing for ease of designing new, low cost digital-to-analog converters into the system.

The system interfaces with the user through a 24 key keyboard and a 7-1/2 digit display on the front panel. System control instructions and data are entered with the keyboard. The display prompts the user during the information input cycle, and provides system status information. These two components, in conjunction with the software operating system, provide the friendliness of the device.

The system is supplied with two digital interfaces for transferring data to a computer for processing and storage. The first is a sixteen-bit parallel interface with two line handshake; and the second is an IEEE-488 bus compatible, talker only interface [2]. The transferred information includes the converted data, the number of the data channel, and parity information. Using these two interfaces the data acquisition system can be configured to interface with almost any computer.

IV. SOFTWARE DESCRIPTION

The software is the key to fulfilling the design goals which were established at the outset of the project. Through careful software design it is possible to provide a great deal of system flexibility while keeping the user interface simple. Since system flexibility is limited only by the user's imagination, it is not possible to provide support for all possible applications. Two general types of data acquisition routines are provided, and full user support is provided for these two routines.

The software consists of two real time routines for controlling the hardware, a series of general purpose subroutines, and an operating system to tie everything together. Each of these software blocks is described below.

The general purpose subroutines consist of drivers for the display and keyboard, two variable length precision timers, and a BCD to hexadecimal converter. These routines, which handle many of the housekeeping functions of the system, are designed to assist the user in developing custom software if the operating system provided doesn't meet the experimental requirements. As an aid to users developing special software, these routines are stored in a separate EPROM for easy inclusion with the user's special software.

The two real time routines provide two different sampling strategies. The evenly spaced sampling routine samples the channels in the user specified order with a user selected constant time delay between each sample. This type of sampling is used for most general types of data acquisition. The bunched sampling routine samples the channels specified by the user

at the maximum rate possible in the system, and then provides a user selected constant time delay before repeating the sampling. This type of sampling is useful for testing transducers, and similar applications where it is desired to sample a number of variables at essentially the same time, and then allow time for them to change before repeating the sampling. Both of the real time routines subtract, from each reading, an offset which is measured for each channel immediately prior to the beginning of the sampling operation.

The operating system prompts the user to provide information about the sampling requirements, and then initializes the proper real time routine to meet these requirements. The user input data consists of:

1) the real time routine the user wishes to use,
2) the period of delay the user wishes to have between samples, or groups of samples, and
3) the channels the user wishes to have sampled, in the order they are to be sampled.

When all of the parameters are entered, the user can review them if he wishes, or command the system to start sampling. The sampling process can be interrupted at any time and the system will return to the state where the user has the choice of reviewing the input parameters or commanding the system to start sampling.

V. APPLICATIONS

The data acquisition system is being used in several applications in the Department of Electrical Engineering at Kansas State University. Four of these applications are described below to provide insight into the general usefulness of this type of instrument.

One of the original motivations of the data acquisition system was to simplify the instrumentation setups for the physiology experiments which are a part of the bioengineering instrumentation class. These experiments are conducted in conjunction with the Department of Anatomy and Physiology, College of Veterinary Medicine, Kansas State University. Typical bioengineering experiments include physiological signal measurements of cell potentials, electrocardiograms, blood pressure, and respiration. Most of these experiments require a variety of high quality amplifiers which are seldom readily available when setting up these experiments. Also, each of these experiments, with the possible exception of cell potential measurements, lends itself well to data acquisition with a system of this type. Use of the data acquisition system allows for information storage and processing which were previously impossible.

The bioengineering instrumentation class project for the spring semester of 1979 provided an unexpected demonstration of the usefulness of the data acquisition system. The class project involved providing an eight channel analog-to-digital converter interface between a McGaw mass spectrometer and an Intecolor 8080 based microcomputer. Both pieces of equipment are the property of the Department of Anatomy and Physiology, where they are used for respiratory research. The data acquisition system

was used as an evaluation tool to aid the class in specifying and designing the analog-to-digital converter interface hardware and the associated software for the Intecolor computer.

The impedance pneumography research program at Kansas State University provides a third example of the use of the data acquisition system. Impedance pneumography is a noninvasive diagnostic technique which provides information about the respiratory system from impedance measurements obtained from thoracic electrode placements. In its most common form a constant current sinusoidal excitation is applied to the subject with a pair of electrodes and the voltage developed across the subject is detected. The measured parameters of interest are:

1) the baseline impedance magnitude (Z_o),

2) the magnitude of changes in the baseline impedance (ΔZ), and

3) the baseline impedance phase angle (ϕ_o).

The baseline value can be measured directly with the data acquisition system. Since changes in the baseline magnitude are generally small, the average base line value should be subtracted from the current baseline value to simplify measurement of the impedance variations. If a switched electrode array is employed, this is done by applying the baseline signal to one input of a differential amplifier and an estimate of the average value, from a digital-to-analog converter controlled by the computer, to the other input of the differential amplifier. If the excitation signal is also digitized, the phase angle of the baseline impedance can be caluclated by the computer. In this case the availability of a general purpose data acquisition system precluded the necessity of building a dedicated system for this project.

A fourth example of the application of this system is its use in the demonstration of basic signal processing phenomena in introductory classes. Students frequently understand basic theorems better if they can see an example of the application of these theorems. The data acquisition system, along with an oscillator and oscilloscope, can be used to demonstrate minimum sampling rate and aliasing phenomena very easily. Slightly more complicated demonstrations, which might necessitate special software for the data acquisition system, could be useful in describing simple digital filters.

VI. CONCLUSIONS

A versatile microprocessor based data acquisition system has been designed and fabricated for use in a bioengineering instrumentation laboratory. It is used to greatly simplify the equipment setup for many of the physiology and instrumentation experiments. The system will also be useful in other teaching and research settings in providing data acquisition at a reasonable cost.

VII. REFERENCES

1. Arnett, D. W., "Development of Modular Laboratory Equipment for Instruction in Biomedical Instrumentation," IEEE Transactions on Engineering in Medicine and Biology, BME-25: 441-445, 1978.

2. Babb, S. M., J. L. Schmalzel, M. S. P. Lucas, "A General Purpose IEEE-488 Bus Interface," to be published in IEEE Transactions on Industrial Electronics and Control Instrumentation, 1979.

3. Hathaway, J. C., A. M. Cook, W. D. Smith, "A Versatile Microprocessor Based Instrumentation System for Use in Biomedical Engineering Instruction," Proceedings of the 13th Annual RMBS, Biomedical Sciences Instrumentation, Vol. 12, 1976.

4. Vranesic, Z. G., S. G. Zaky, "Nonnumerical Applications of Microprocessors," Proceedings of the IEEE, Vol. 64, No. 6: 954-959, 1976.

VIII. ACKNOWLEDGEMENTS

The author would like to thank his major professor Dr. Richard R. Gallagher and graduate committee members, Dr. Michael S. P. Lucas and Dr. M. Roger Fedde, for their support and guidance throughout this project. Thanks are also due Mr. Frederico Faggin, President of Zilog, for his donation of the Z-80 microprocessor; Mr. John Schmalzel for his invaluable advice and assistance throughout the project; and Mr. Samuel Babb for providing the IEEE-488 bus interface. The author finally thanks his family and friends for their support.

APPENDIX I: USER'S MANUAL

AI.1. Introduction

The data acquisition system allows the user to digitize up to eight analog signals with a maximum bandwidth of ten kilohertz (single channel). The plug-in amplifiers and the system's microprocessor controlled features provide maximal system flexibility and ease of use. The digitized data is prepared for acceptance by a computer for storage and/or processing.

This user's manual describes the use of the instrument and its many features. First, a short tutorial is given on sampling theory to help the user understand the system's requirements and limitations. This is followed by a description of the system hardware which is available for the user's interface to the system. A third section describes the use of the system, and gives three examples of its application for enhancement of the user's understanding. Lastly, there is a brief discussion of several special requirements, followed by a description of the computer interfaces.

AI.2. Sampling Theorem and Options

The idea behind sampling and digitizing a signal is to take the minimum number of samples required to preserve the desired information contained in the signal. One form of the sampling theorem states that, in order to retain all of the information in a signal, the sampling rate must be equal to or greater than twice the highest significant signal frequency component. In the case of general signals this is sufficient, since the contribution of the higher frequency components of a signal is generally small. In the case of digitizing a pure sine wave, it would be desirable to make many more than two samples per cycle in order to minimize the distortion.

The sampling theorem assumes a "brick wall" bandlimited signal. In general the signal of interest will contain higher frequency components than those needed to preserve the signal characteristics. An example is high frequency noise in the signal. If these high frequency components are present in the signal when it is digitized they will appear in the digitized data as signal components at lower frequencies. This phenomenon, known as aliasing, can be a major problem. It is recommended that all signals be bandlimited, either by the nature of the signal or by filtering before conversion.

The system provides two sampling options: evenly spaced samples, and bunched samples. For nearly all data conversion applications the evenly spaced sampling option is preferable. It samples the selected channels in a specific order with a constant delay between each sample that is taken. This insures that the samples for each channel are taken at evenly spaced times. Also, it is possible to sample some channels faster than others without causing sampling problems. The bunched sampling option is especially useful for such applications as testing transducers. This sampling option samples each of the chosen channels at the system's maximum rate, and then provides a delay before sampling the signals again. This sampling scheme is particularly useful for comparing a group of similar transducers with each other or with a standard. This method samples all of the transducers at essentially the same time, and then provides a delay allowing conditions to change before taking another group of samples.

AI.3. User Interface Hardware

The system provides the user with three types of amplifiers. While these amplifiers were designed to meet specific needs, they have some common

characteristics. These common characteristics will be discussed below. A description of the individual amplifiers follows.

All of the amplifiers provide a gain of 1 to 1000 variable in 1, 2, 5, 10 steps. The gain is selected by using the thumbwheel switch on the front panel of each amplifier module. The gain for each setting is tabulated on the front panel of the instrument.

Amplifier gain variations are provided for by a separate unit built into each amplifier. Limitations of the gain unit used in the original design restrict the use of gains to less than or equal to 100 for maximal system performance. The amplifier in the gain unit has a large signal gain bandwidth product of 25 kilohertz. The performance of the amplifiers can be improved by using a higher performance gain unit.

All of the amplifiers are built in plug-in modules. This allows the user to place any of the amplifier types in any of the amplifier slots (the eight leftmost slots). The modular plug-in nature of the amplifiers also allows the user to build special amplifier plug-in units.

The amplifiers have two or three BNC connectors on their front panels. The upper one is an output connector, and the lower one(s) are the input connectors.

The output connector provides the user with access to the signal being presented to the analog-to-digital converter. This is useful for monitoring the signal or for cascading two or more amplifiers.

If there is only one input connector, it is ground referenced. These amplifiers should be used only for ground referenced signals. Care should be exercised to avoid ground loops.

The simple amplifier is a ground referenced amplifier. It is useful for analog and slow digital signals in ground referenced systems. This

amplifier is also useful in applications where two or more of the system amplifiers are connected in series.

If the amplifier has two input connectors the inputs are differential. The upper input connector is the noninverting (positive) input, and the lower input connector is the inverting (negative) input. The outer shield of these two connectors are tied together, but they are not tied to the system ground. When differential amplifiers are used, the outer shield connection of one of the inputs (generally the inverting input) is connected to the signal source. The connection will generally be made to the source ground, or at the same point as the inverting input.

One of the differential amplifiers, an instrumentation amplifier, is intended for use with signals involving a common mode component up to ten volts peak-to-peak. It has very high common mode rejection, and is suitable for use with signal levels as low as five hundred microvolts. This amplifier is ideal for most applications.

The other differential amplifier, an isolation amplifier, is intended for use with signals involving high common mode voltages, and in applications where it is necessary to have the amplifier inputs electrically isolated from the output for safety reasons, such as in demanding medical uses. The bandwidth of this amplifier is limited to 700 hertz, an adequate bandwidth for many applications.

The system will support one or two digital-to-analog converter plug-in modules (in the two rightmost slots) each containing up to four digital-to-analog converters. These converters are generally used to monitor the data being passed from the system by converting it back to an analog signal. These digital-to-analog converted signals can be compared to the amplifier

outputs to check for conversion problems. In normal operation each digital-to-analog converter corresponds directly to the amplifier on the same channel number.

AI.4. Preparation for System Use

Initial preparation for the use of any data acquisition system requires specification of the hardware configuration and identification of parameters to be monitored. The following paragraphs describe this process. Three examples then step the user through this process.

The user should faithfully record how the system is configured for each experiment. Identification of the input parameter connections--amplifier types, channel numbers and gains, the sampling scheme and rate, and the channel order list--are extremely important when future data analysis is done. In summary, the user is required to know exactly how the data was obtained.

The description of the various amplifier types on the previous pages should provide considerable insight for the selection of the amplifiers required for most applications. Unfortunately, definite rules about amplifier selection cannot be given.

The user must decide whether to use the evenly spaced or bunched sampling option for each application. Both were described earlier along with the advantages of each. For most applications, the evenly spaced sampling option is preferred.

The sampling rate, chosen on the basis of the sampling theorem discussed earlier, provides two of the system parameters: the timing routine, and the delay. Each of the sampling options requires a minimum amount of

time between samples; in the evenly spaced option this minimum time is 52.5 microseconds, and in the bunched option there is a constant delay of 38 microseconds between samples and a minimum of 56 microseconds between groups of samples. The delay entered by the user is added to the minimum time in each case. The delay the user enters is a four digit decimal number (between 0000 and 9999). If the short timer is used, the added delay is the number entered times 25 microseconds plus 1 microsecond (the added microsecond doesn't apply if the delay entered is 0000). In the evenly spaced sampling routine this provides for sampling rates of 4 to 19047 samples per second. The long timer provides an added delay of 0.2 seconds times the number entered. This yields sampling rates of 5 samples per second to 1.8 samples per hour for the evenly spaced sampling routine.

When calculating the sampling rate required, it must be remembered that each channel must be sampled. The examples should guide the user in calculating sampling rates and the required delay.

The channel order list is the last item to be entered. This tells the processor the order in which the user wishes to have the channels sampled. The input signals and their corresponding channels should be recorded for future reference. The system will provide the computer with the number of the channel from which the data came, but it cannot provide information about what signal was applied to that channel.

AI.5. System Use

The first step in using the system is to configure the hardware to meet the experiment's requirements. This involves placing the correct amplifiers in the appropriate slots. Unused channels may be left blank or may have an unused amplifier stored in them. The system should then be turned on and

allowed to warm-up while connections are being made and the input parameters are being entered. This warm-up time allows the system to stabilize before measurements are made.

Generally it is best to make all connections to the system next. The major exception is connection of simple amplifiers if they are being used directly for inputs. The simple amplifiers should have their inputs shorted so that they will be properly initialized when sampling is started.

The input parameters should now be entered. The processor will prompt the user throughout the input procedure. If inconsistent responses are entered the prompt will be repeated.

When the system is turned on, the prompt "EO bl?" will appear in the display. The user should push the "0" key to select evenly spaced sampling, or the "1" key to select bunched sampling. After the selection is made the "enter" key should be pressed to cause the processor to take the entry.

The second prompt is "Ll SO?". The user should choose the long or short timer using the same method as above for selecting the sampling option. Next the other half of the delay parameter will be requested with the prompt "P?". The user should enter the period of delay he wants as a four digit number (include leading zeros). The enter key should be pressed after each digit.

The final set of parameters to be entered is the channel order list. The prompt "CH?" will be displayed and the user should begin entering channel numbers, one at a time, in the order that the channels are to be sampled. After each channel number is pressed the enter key should be pressed. At the end of the channel order list the "end list" key should be pressed.

After all of the parameters are entered the display will contain the prompt "run?". At this point the user may press the "?" key and the processor will display each of the parameters which were entered. This allows the user to check for entry errors. After each parameter is displayed the user must press the "cont" key to cause the program to continue to the next parameter. When the "run?" prompt is displayed the user may also press the "run" key. This will cause the processor to execute an initialization routine. Following initialization the system will start sampling the channels and passing the data to the computer. The display will change to "run" to indicate the state of the machine.

If the user should wish to stop the data acquisition at any time with the option to start sampling again, the "pause" key should be pressed. This step will return the machine to the state it was in at the beginning of the previous paragraph. This action would be desirable if the user wishes to make equipment adjustments, change input connections, or change the gain of one of the amplifiers.

After the processor has entered the sampling mode ("run" displayed) the shorting connections can be removed from the simple amplifiers being used for input, and their input signals applied.

If a digit key is pressed by mistake during data entry, the error can be recovered by pressing the correct key. The processor always accepts the last digit pressed prior to the "enter" key being pressed. If a mistake is made in entering data (for example the wrong digit is pressed and the "enter" key is pressed) the "reset" key must be pressed and the data entry will start at the beginning.

There are two overrange lights on the front panel of the system. One of them blinks when one of the input channels exceeds the range of the

analog-to-digital converter. The other turns on the first time such an overrange occurs and is latched in the on state. If the first light is blinking regularly, or both appear to be turned on continuously, the amplifier outputs should be checked. Any amplifier which has an output signal exceeding positive or negative five volts should have its gain reduced. Pressing the "pause" or "reset" key will clear the latched overrange light. This will also stop the system's sampling operation.

AI.6. Example One

This example serves to introduce the user to the use of the system, and demonstrates aliasing. This demonstration requires a variable frequency sine wave oscillator and a dual trace oscilloscope, as shown in Fig. AI.1.

The oscillator should be started at 100 hertz. The sampling rate is to be chosen to be sufficient for a 500 hertz input signal. The frequency of the oscillator should be increased beyond 500 hertz to demonstrate aliasing. The oscilloscope will be connected to the oscillator output and the D/A converter output, allowing comparison of the original and reconstructed signals.

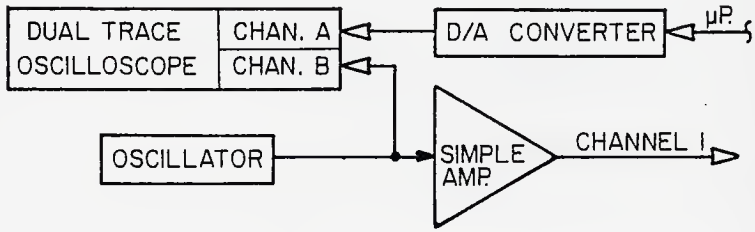


Figure AI.1. Schematic of Example 1

The first step is to select the type of amplifier which will be used. In this application either a simple amplifier or an instrumentation amplifier could be used. The isolation amplifier is ruled out by its limited bandwidth. For connection simplicity the simple amplifier will be used.

The amplifier is installed in channel 1 (the leftmost slot). Since the signal is a pure sine wave it is definitely bandlimited, so a filter is not required.

It is assumed that the output of the oscillator is adjustable to ± 4 volts, so additional gain is not required in the amplifier. The gain of the amplifier is set to 1 (gain switch setting 0).

The connections of the equipment are fairly simple. The output of the oscillator is connected to both the input of the amplifier and channel B of the oscilloscope. The output of D/A converter 1 in the data acquisition system is connected to channel A of the oscilloscope. The connection to the simple amplifier should not be made immediately, as explained in Section AI.5: System Use.

As mentioned before, the sampling rate will be chosen to be sufficient for a 500 hertz signal. From the sampling theorem given at the beginning of this user's manual, it is known that the sampling rate must be at least twice the maximum frequency component, or a minimum of 1000 samples per second in this example. This corresponds to a maximum intersample delay of 1 millisecond, or 1000 microseconds. This delay requires the use of the short timer, as the minimum delay with the long timer is 200 milliseconds.

In this application a signal is being digitized for reconstruction, and therefore it is desired that all of the samples be evenly spaced. For this reason the evenly spaced sampling routine is chosen.

Calculation of the number to be entered as the period for the timer involves remembering the timing requirements of the sampling routine and the timer, and doing a very little math. The evenly spaced sampling routine has an inherent delay of 52.5 microseconds between samples. The short timer provides a delay of 25 microseconds times the period entered plus a constant error of 1 microsecond. Subtracting the 52.5 microsecond delay for the evenly spaced sampling routine from the required 1000 microsecond delay yields 947.5 microseconds to be made up by the short timer. The first step in calculating for the short timer is to subtract the 1 microsecond error, resulting in 946.5 microseconds to be made up by the iterative section of the short timer. Dividing by 25 microseconds for each pass through the timing loop results in 37.86 passes through the loop. A whole number is required, so it must be decided whether to use 37 or 38. Since the minimum sampling rate is to be 1000 samples per second the maximum delay should be 1000 microseconds. Thus 37 is chosen for the constant to be chosen (always choosing the smaller number will result in a sampling rate slightly higher than required). The above calculation is summarized below.

minimum sampling rate = 1000 samples per second

$$\frac{1}{1000 \text{ samples per second}} = 1000 \text{ usec max/sample}$$

1000.0	usec max delay
-52.5	for evenly spaced sampling routine
947.5	usec
- 1.0	for short timer error
946.5	to be made up by timer loop

$$25\overline{)946.50}$$ 37.86 number of times through loop

25 usec for each time through loop

The number of times through the loop is always rounded to the next lower whole number, thus the number 37 is entered as the period.

The actual sampling rate will be 1022 samples per second. This calculation is shown below.

37.0	number of times through timer loop
x 25.0	usec per time through loop
925.0	usec due to timer looping
+ 1.0	usec short timer error
926.0	usec from the timer
+ 52.5	usec inherent in even sampling
978.5	usec between samples

$$\frac{1 \text{ sample}}{978.5 \text{ usec}} = 1022 \text{ samples per second}$$

Since the only channel being sampled is channel 1, the channel order list consists of just this one channel.

All of the input parameters are now defined. The choice of sampling routines is the evenly spaced sampling routine (enter "0"). The timer is the short timer (enter "0"). The period is 37 (enter "0037"). The channel order list is the channel 1 (enter "1", and then press the "end list" key).

The input data may now be reviewed, or the sampling process may be begun at this point. This is described in detail in Section AI.5. System Use.

Once the sampling is started the user should vary the frequency of the oscillator away from the initial setting of 100 hertz. As the frequency is increased up to 500 hertz the user will notice on the oscilloscope that the output of the D/A converter becomes a rougher and rougher approximation of the input waveform, until at 500 hertz it becomes a square wave. As the frequency is increased above 500 hertz the frequency of the output of the D/A converter will appear to decrease. This phenomenon is known as aliasing, and is the result of not sampling the input wave form at

a sufficiently high rate. The decrease in apparent frequency will continue until the oscillator reaches a frequency of 1000 hertz, at which time the output of the D/A converter will be DC. As the frequency of the oscillator is increased above 1000 hertz the apparent frequency at the output of the D/A converter will increase until the oscillator reaches 1500 hertz when the frequency will again begin to decrease. This frequency cycling will continue for higher and higher frequencies.

AI.7. Example Two

Suppose one desires to monitor a common three-lead electrocardiogram (ECG) from a healthy dog. The desired bandwidth is 100 hertz. The nature of the signal is not bandlimited and the signal levels preclude filtering the signal before amplification.

For this application either an isolation or instrumentation amplifier could be chosen. The isolation amplifier provides a greater degree of safety to the subject. For the purposes of this example the subject is assumed to be healthy, and not involved in a class A medical environment; consequently instrumentation amplifiers are selected. See Fig. AI.2.

Since the signals are not bandlimited, the outputs of the instrumentation amplifiers will be fed into 100 hertz low pass filters. The outputs of the filters will be connected to the inputs of simple amplifiers.

The instrumentation amplifier for lead I will be installed in channel 4, lead II in channel 5, and lead III in channel 6. The simple amplifiers will go in channels, 1, 2, and 3, respectively. The gains of the instrumentation amplifiers are set at 200 (it is suggested to raise the AD521 gain to do this--see the detailed hardware documentation in Appendix II for details). The simple amplifiers will then be set for gains which provide

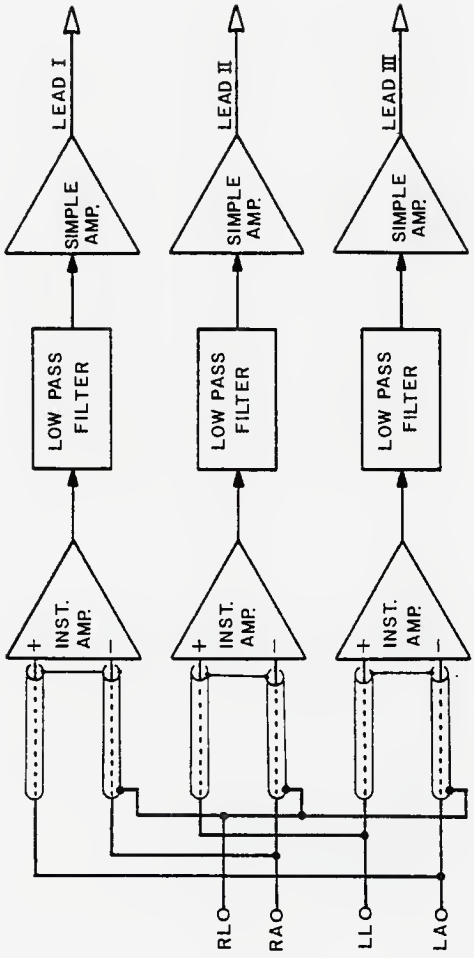


Figure AI.2. Schematic of Example 2

signals with a peak of approximately 4 volts. A gain of 50 for channel 1, and a gain of 10 for channels 2 and 3 would be good starting points.

The bandwidth of 100 hertz per channel requires sampling each at a rate of 200 samples per second for a total of 600 samples per second. This requires an intersample delay of 1666.7 microseconds, so the short timer will be used. Subtracting the 52.5 microsecond inherent delay for the evenly spaced sampling option one gets a result of 1614.2 microseconds of delay to be made up by the timer. Subtracting the extra 1 microsecond for the short timer, and dividing the result by 25 one gets 64.528 as the number of periods. Since the number must be an integer, 64 is selected. The next lower integer is selected so that the sampling rate will always be greater than or equal to that desired.

The input parameters are now defined. The choice of sampling routine is evenly spaced samples. The timer is the short timer with a delay parameter of 0064. The channel order list is 1, 2, 3.

AI.8. Example Three

Two bandlimited signals are assumed, one with a maximum frequency component of 900 hertz and another with a maximum frequency component of 700 hertz. Both signals are fed to a multiplier. The result is a signal with a maximum frequency component of 1600 hertz. All three signals are 0.8 volts peak-to-peak. It is desired to digitize each of the three signals.

Three instrumentation amplifiers are used for the signal inputs. The 700 hertz signal is connected to channel 1, the 900 hertz signal to channel 2, and the 1600 hertz signal to channel 3. Each instrumentation amplifier has a gain of 10.

Since the output of the multiplier has a bandwidth of about twice that of the input signals, a sampling trick can be employed to avoid having to

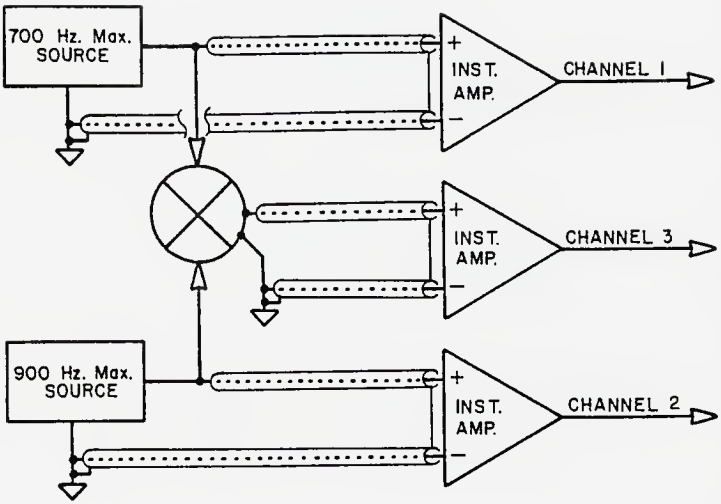


Figure AI.3. Schematic of Example 3

sample the input signals at an unnecessarily high rate. The channels are sampled in the order 1, 3, 2, 3. This results in the multiplier output being sampled at twice the rate of the inputs. The evenly spaced sampling option provides for all samples to be properly spaced.

The limiting factor in the sampling will be the 900 hertz bandwidth input. This channel requires a sampling rate of 1800 samples per second, yielding a total of 7200 samples per second. Using a calculation like the one in the previous examples one gets a delay constant of 3.

The input parameters for this example will be: evenly spaced samples, short timer, delay of 0003, and a channel order list of 1, 3, 2, 3.

AI.9. Special Requirements and Operational Problems

To meet special user applications not provided for in the standard system, the user can often modify the system. The hardware description in Appendix II explains the requirements for special amplifier and digital-to-analog converter modules. The detailed software description in Appendix III gives some insight into developing special software. Special plug-in modules should be able to provide for nearly any interface requirements the user may have, while special software can provide sampling rates up to 80,000 samples per second.

In some applications it is not possible to bandlimit the input signal before it is input to one of the amplifiers. In this case the output of the amplifier can be run to a filter network, and the output of the filter can be fed to a simple amplifier for input to the system (as in Example 2). The signal should then be sampled from the simple amplifier and not from the input amplifier. If the input amplifier is an instrumentation or isolation amplifier the input of the simple amplifier will not have to be shorted on system startup.

If the system fails to operate, the power switch should be turned off and the unit disconnected from line power. The user should then check the fuse on the back panel. If this is not the problem, service should be referred to a qualified person.

AI.10. Computer Interfaces

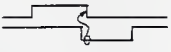
Two computer interfaces are available to the user: a sixteen bit parallel interface, and an IEEE-488 bus interface. Both interfaces are used to pass data to a computer for storage and processing. The data provided consists of the eight bits of data from the converter, four bits of channel

address, and parity for the data. The system will not attempt to place more data on either interface until it has received an acknowledgement from the computer that the data has been received. Though new data will not be placed on the interface, the system will continue taking samples; consequently data can be lost. The use of both interfaces is described below.

The link between the 16-bit parallel interface and computer is a 16 conductor ribbon cable. Fourteen of the lines are used for data transfer, and two are used for handshaking. Handshaking consists of a line to the computer indicating that the data is ready, and a line from the computer indicating that the data has been received.

Pin Assignments:

pin	function
1	data Ø (LSB)
2	data 1
3	data 2
4	data 3
5	data 4
6	data 5
7	data 6
8	data 7 (MSB)
9	address Ø (LSB)
10	address 1
11	address 2
12	address 3 (MSB)
13	even parity
14	odd parity
15	data ready
16	data accepted



The IEEE-488 bus interface provides a computer interface in accordance with the requirements of the IEEE-488 standard for talker only operation. This interface plugs into the 16-bit parallel interface, and provides proper handshaking with it. The data is passed to the computer in byte serial format.

Data Format:

first byte:

bit	function
Ø (LSB)	data Ø (LSB)
1	data 1
2	data 2
3	data 3
4	data 4
5	data 5
6	data 6
7 (MSB)	data 7 (MSB)

Data Format:

second byte:

bit	function
Ø (LSB)	address Ø (LSB)
1	address 1
2	address 2
3	address 3 (MSB)
4	even parity
5	odd parity
6	unused
7 (MSB)	unused

AI.11. Specifications

1. power requirements
   - 100 - 125 Vacrms
   - 60 Hz
   - 3 Aacrms

2. amplifers
   - all:
     - gains of 1 to 1000 in 1, 2, 5, 10 steps
     - output voltage: ±5 Vpeak
     - input impedance: 30 Mohm minimum
     - output impedance: 5.2 kohm
     - inputs and outputs protected to 170 Vpeak
   - simple (308A):
     - large signal gain bandwidth: 25 kHz
   - instrumentation (AD521 + LM308A):
     - large signal gain bandwidth
       - AD521 gain = 1: 25 kHz
       - AD521 gain > 4: 100 kHz
     - common mode rejection ratio
       - AD521 gain = 1: 70dB
       - AD521 gain = 1000: 100dB
   - isolation (284J + LM308A):
     - large signal gain bandwidth: 700 Hz
     - common mode rejection ratio: 110dB

3. digital-to-analog converter module
   - output voltage: ±5 Vpeak
   - output impedance: 2.5 kohm

4. sampling rate
   - with standard software:
     - 19,047 samples per second maximum
     - 1.8 samples per hour minimum
   - with special software:
     - 80,000 samples per second maximum
     - no minimum sampling rate

APPENDIX II: DETAILED HARDWARE DOCUMENTATION

AII.1. General Comments

The hardware can be broken into three broad classes: analog, digital, and interfaces between the analog and digital circuits. The analog section is used mostly for interfacing with the real world; and the digital section is used mostly for control, and interface with the user and the device to which the data is being transferred.

In the following paragraphs, each of the sections will be described in terms of the features it offers the user. In the pages following this, each printed circuit board will be described in greater detail.

AII.2. Analog

The analog section consists of the input amplifiers and the power supply. The input amplifiers are built in plug-in modules so the user can configure the system to meet his needs, and in the case of special requirements, it allows the user to build his own plug-in modules. The power supply supports the rest of the system and can handle some expansion without modification.

The original system contains three types of amplifers: a simple amplifier, an instrumentation amplifier, and an isolation amplifier. The simple amplifier is intended for use with ground referenced signals free of common mode noise. The instrumentation amplifier is intended for use with signals requiring very high quality amplification under conditions of up to five volts of common mode noise. The isolation amplifier is intended for applications which require very high voltage isolation between the

inputs and output. This includes industrial applications involving very high common mode voltages, and demanding medical applications. All three of the amplifiers have gains variable from one to one thousand and are designed to provide a bipolar five volt output swing.

The power supply consists of three separate supplies: the logic supply, and the positive and negative analog supplies. All three are high current, unregulated supplies. The entire system uses on-board local regulators for voltage regulation.

AII.3. Digital

The digital section consists of the microprocessor, memory, user interface, and the interfaces to the device to which the data is being transferred.

The microprocessor provides control signals for the rest of the system and allows for great system flexibility while simplifying the logic design. The processor is a modified S-100 bus compatible Z-80 board. The memory includes EPROM's to provide a firmware operating system, and RAM to provide temporary storage. Through the use of the various analog and digital interfaces, the processor can be used to provide control outside of the system for enhanced measurement capabilities.

The user interface consists of the keyboard and the eight digit display on the front panel. The keyboard allows the user to enter decimal data and control signals to the processor. The display prompts the user during system setup, and allows the user to receive some information on the status of the system. The display provides a hex character set and a limited number of alphabetic characters.

The digital interfaces in the original system include a parallel sixteen bit output port with limited handshaking, and an IEEE-488 bus interface configured for talker only operation.

AII.4. A/D and D/A Interfaces

This section of the hardware consists of the S-100 bus compatible A/D board housed in the upper section of the unit, and the D/A converter modules which are plug-in units similar to the amplifiers. Both types of converters are eight bit, five volt full scale, bipolar converters. The A/D converter produces 2's complement code. The D/A converters use straight binary code, but they have been modified to accept 2's complement code.

The A/D converter is used to convert the signals from the amplifiers. It consists of a sixteen channel multiplexer, a sample-and-hold amplifier, a successive approximation A/D converter, and support circuitry. It is configured to accept bipolar five volt signals. When addressed by the processor, it passes the data from the previous conversion and begins a conversion on the next specified channel. The maximum sampling rate of the converter is slightly over eighty-thousand samples per second.

Each of the D/A modules support up to four D/A converters. The converters are individually addressable and latch the data passed to them. For convenience, the port addresses of the converters are the same as the lower byte of the address of the input amplifiers (as seen by the A/D converter). This significantly simplifies the software when it is desired to use the D/A converters to feed back the data converted from the amplifiers.

AII.5. System Addresses

memory (by hex address):

0000-03FF	EPROM
0400-0FFF	unused
1000-1FFF	RAM
2000-803F	unused
8040-8047	lower eight A/D converter channels
8048-804F	upper eight A/D converter channels
8050-FFFF	unused

I/O ports (by hex address):

00	keyboard
01-3F	unused
40-4F	D/A converters
50-DE	unused
DF	data output
E0-EE	unused
EF	control for shorting and disconnecting amplifiers
F0-FF	display

AII.6. Amplifier Module - 308 Version

The purpose of the amplifier module is to provide variable gain with wide bandwidth, low offset, and low noise. These requirements are largely met by the 308 version.

The gain is variable from 1 to 1000 in 1, 2, 5, 10 steps. The gain variation is accomplished by using CMOS analog gates to connect different feedback pairs to the summing node. This technique provides low gain error and low noise. While gains to 1000 are available, gains above 100 are impractical in this unit for reasons described below.

The gain bandwidth product of the LM308A amplifier is 1 MHz. For gains up to 100, this allows for signals whose maximum frequency component is 10 kHz, provided the output voltage swings are not large (small signal operation). The large signal bandwidth of the LM308A is about 25 kHz, which significantly limits its useful bandwidth.

The offset of the LM308A is specified to be less than 0.5 mV, but is not easily trimmable (special circuit techniques have to be used). This offset can be compensated by the system software, but for large gains (above 100) it can be objectionable because of its magnitude relative to the signal.

Suggestions:

For demanding applications, it would be wise to consider using another op amp. Some of the new high-performance types, such as the LF356 and the LF357, would be good choices as would be the ultra-wide bandwidth op amps, such as the Harris 2625, or 2525. Most of these can be substituted pin-for-pin in the LM308A socket, but it is suggested that the board be redesigned to add the ability to trim the input offset voltage.

Functional Description:

Amplifier IC1 provides the circuit gain. The analog switches in IC2, IC3, and IC4 connect one of the feedback resistor pairs to the amplifier, thereby setting the gain. Resistor R2 along with diodes CR3 and CR4 provide input overvoltage protection. Resistor R1 along with diodes CR1 and CR2 provide protection for the output to the front panel.

Parts List:

resistors are carbon film, 1/4 W, 1% unless specified

C1	cap, cer disk, 100pF, 20V, ±10%
C2-C4	cap, cer disk, 0.1uF, 16 V, ±20%
CR1-CR2	diode, 1 A, 400 PIV, 1N4004
CR3-CR4	diode, small signal, 1N914
IC1	op amp, LM308A
IC2-IC4	CMOS analog gate, CD4016
R1-R2	res, car comp, 5.1k, 1/4 W, 5%
R3-R13	res, car comp, 10k, 1/4 W, 5%
R14	res, 10k
R15	res, 4.99M
R16	res, 10k
R17	res, 2.49M
R18	res, 4.99k
R19	res, 1.02M
R20	res, 5.07k
R21	res, 499K
R22	res, 4.99k
R23	res, 249k
R24	res, 5.07k
R25	res, 100k
R26	res, 5.23k
R27	res, 49k
R28	res, 5.49k
R29	res, 24.9k
R30	res, 6.19k
R31-R32	res, 10k
R33	res, car comp, 5.1k, 1/4 W, 5%
R34	not used (open circuit)

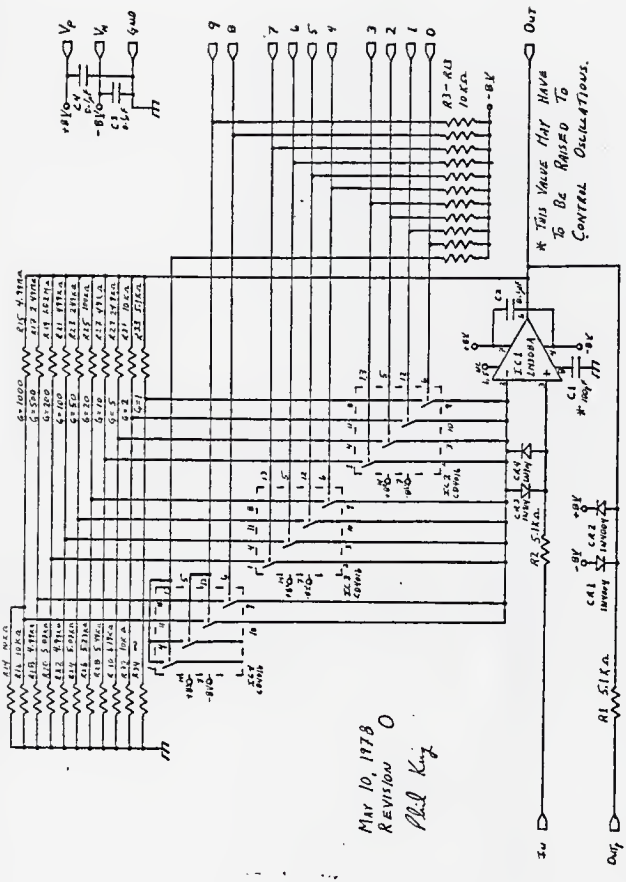


Figure AII.1. Schematic of Amplifier Module - 308 Version

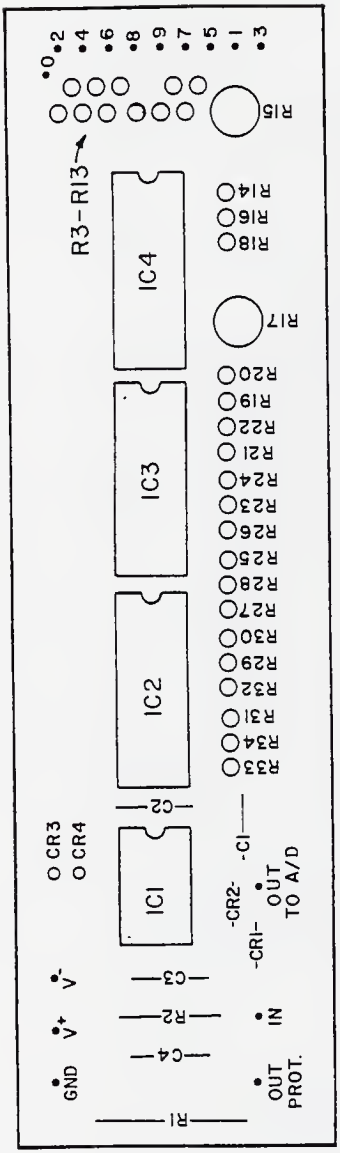


Figure AII.2. Parts Placement of Amplifier Module

AII.7. Simple Amplifier

The simple amplifier consists of an amplifier module and a minimum of support hardware. It is intended to provide additional gain for other amplifiers, or for use in applications involving ground referenced signals which do not have a common mode signal associated with them.

Functional Description:

The amplifier module, A1, provides most of the circuit function. Switch S1 controls the gain of the amplifier. Resistor R1 and diodes CR1 and CR2 provide additional input overvoltage protection for the amplifier module. Zener diodes CR3 and CR4 provide voltage limiting to the input protection circuitry when the unit is not turned on.

Parts List:

A1	amplifier module, previously described
C1-C4	cap, tant, 3.3uF, 35V, ±20%
C5-C6	cap, cer disk, 0.1uF, 16 V, ±20%
CR3-CR4	diode, zener, 9.1 V, 500 mW, 1N5239
IC1	voltage regulator, +8 V, LM340T-8
IC2	voltage regulator, -8 V, LM320T-8
R1	res, car comp, 5.1k, 1/4 W, 5%
S1	switch, thumbwheel, SP10T, decimal

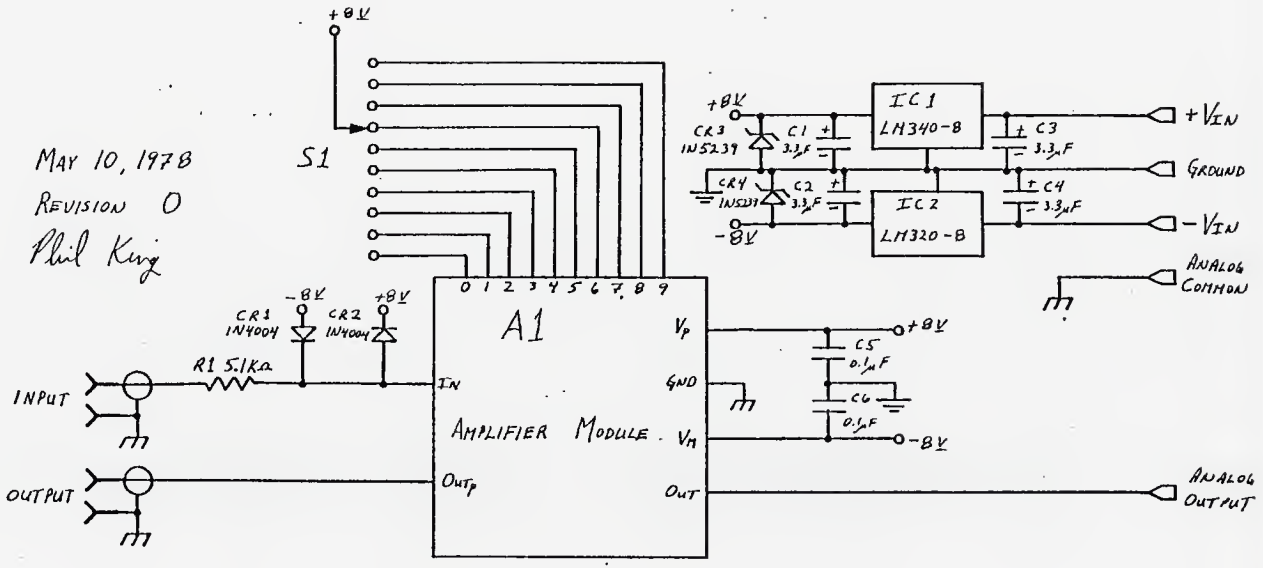


Figure AII.3. Schematic of Simple Amplifier

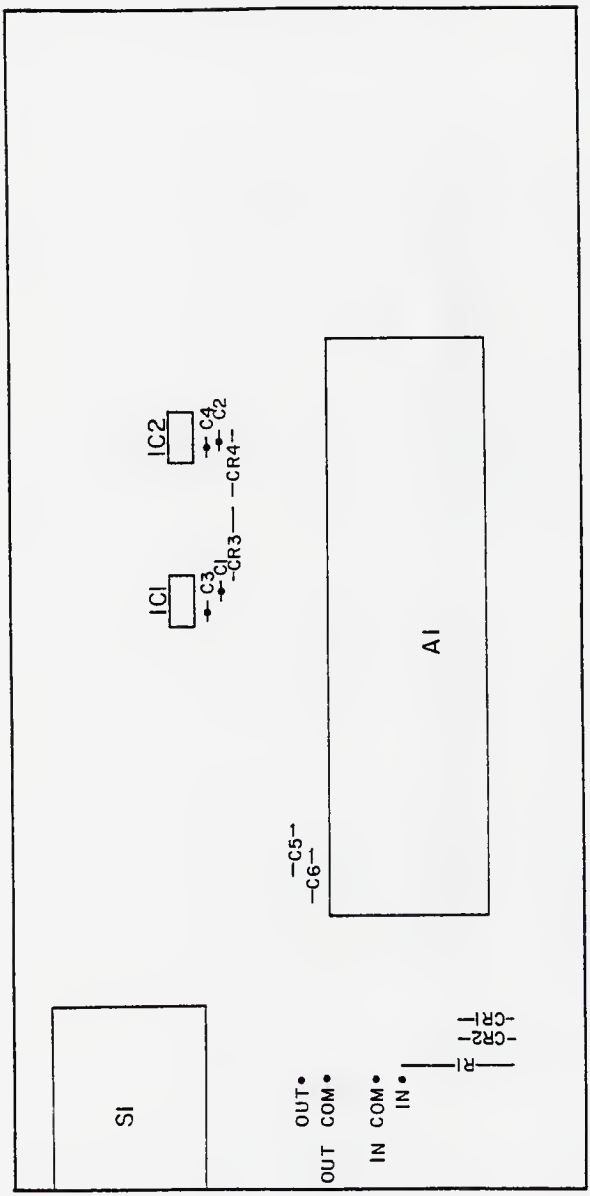


Figure AII.4. Parts Placement of Simple Amplifier

AII.8. Instrumentation Amplifier

The instrumentation amplifier is intended to provide extremely high-grade amplification of low level signals under conditions of up to 5 volts of common mode noise. This unit, designed around the Analog Devices AD521, meets these requirements quite well.

The small signal gain bandwidth product of the AD521 is at least 2 MHz with a large signal gain bandwidth of 100 kHz. The module's common mode rejection is a minimum of 70 dB at a gain of 1 rising to 100 dB at a gain of 1000. The module's input noise is sufficiently low for the amplifier to be used with 12 bit A/D converters for signals down to 5 mV(rms). The AD521 input offset voltage can be trimmed, and the amplifier inputs can be disconnected from the front panel connectors and shorted together to allow for software compensation of the input offset voltage of the entire module.

Suggestions:

If the unit is used with very small signals, the gain of the AD521 should be increased. This improves the common mode rejection and reduces the demands on the amplifier module for gain. The gain change is made by replacing R4, which should be mounted on standoffs for easy removal. The value of R4 for unity gain is 100k. For each decade of reduction of the value of R4, the gain increases a decade, to a maximum recommended gain of 1000. More information is available on the manufacturer's data sheet.

Adjustments:

The input offset voltage of the AD521 should be trimmed with the unit thoroughly warmed-up (at least 30 minutes) in an area free from drafts.

The inputs should be shorted either with a shorting connector on the front panel, or with the internal shorting provisions. Measurements should be made at the output of the AD521 and not at the output of the amplifier module.

Functional Description:

Neon lamp DS1, resistors R1 and R2, and diodes CR1-CR4 provide input overvoltage protection. Analog switch IC1 is used to short the inputs of the instrumentation amplifier IC2. Amplifier module A1 provides the variable gain, under the control of switch S1. The dual comparator IC3 level shifts the TTL control signals for shorting and disconnecting the front panel connections. The Zener diodes CR6 and CR7 provide voltage limiting to the input protection circuitry when the unit is turned off.

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

A1	amplifier module, previously described
C1-C10	cap, cer disk, 0.1uF, 16 V, ±20%
C11-C14	cap, tant, 3.3uF, 35 V, ±20%
CR1-CR5	diode, 1 A, 400 PIV, 1N4004
CR6-CR7	diode, zener, 9.1 V, 500 mW, 1N5239
DS1	lamp, neon, NE2
IC1	CMOS analog gate, CD4016
IC2	instrumentation amplifier, AD521J
IC3	dual comparator, LM319
IC4	voltage regulator, +8 V, LM340T-8
IC5	voltage regulator, -8 V, LM320T-8
K1	relay, dry reed, DPDT, 12 Vdc, 615 ohm coil
Q1	transistor, NPN, small signal, 2N2222
R1-R2	res, car film, 1.2k 1/4 W, 1%
R3-R4	res, car film, 100k, 1/4 W, 1% R4 may vary for non-unity gain
R5	res, variable, cermet, 10k, 10 turn, 1/4 W
R6	res, 6.8k
R7	res, 220 ohm
R8-R9	res, 3.3k
R10	res, 6.8k
R11	res, 12k
R12	res, 2.7k
R13-R14	res, 10k
S1	switch, thumbwheel, SP10T, decimal

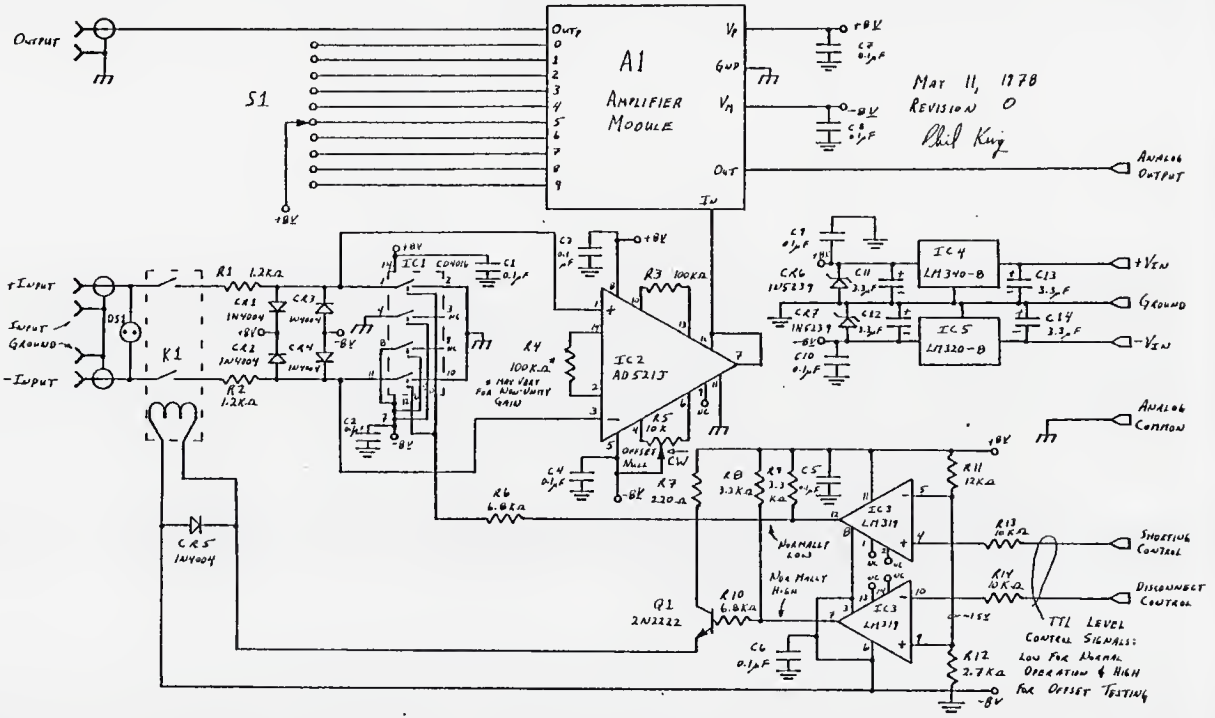


Figure AII.5. Schematic of Instrumentation Amplifier

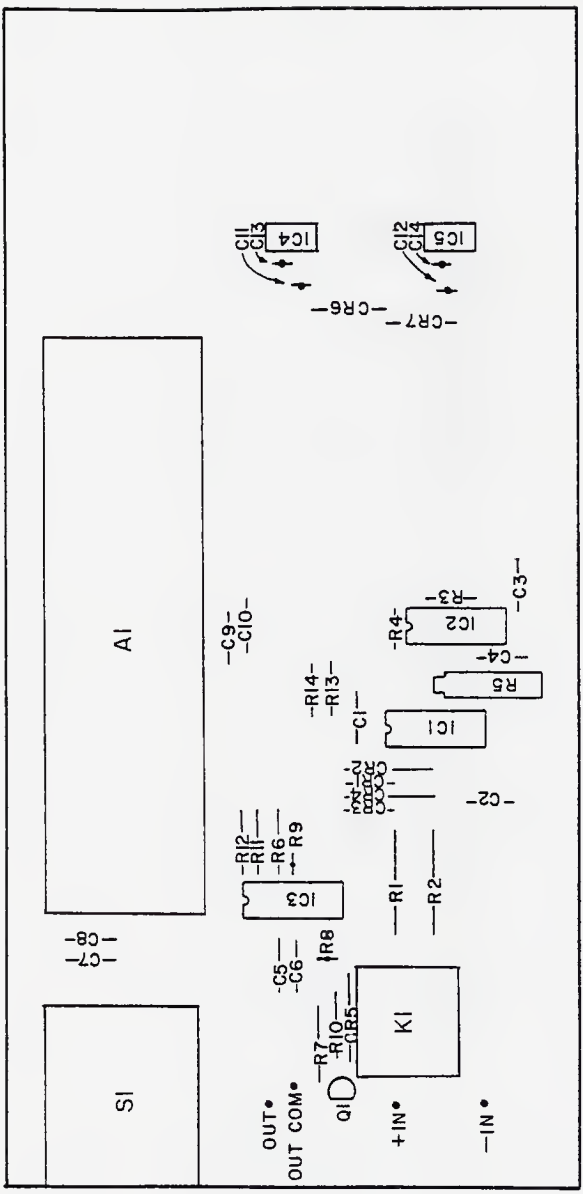


Figure AII.6. Parts Placement of Instrumentation Amplifier

AII.9. Isolation Amplifier

The isolation amplifier was designed for use with common mode voltages up to 150 peak volts continuously, or a pulse of 1500 volts for 10 milliseconds (not to be repeated sooner than 10 seconds). It is useful for applications, such as medical instrumentation, where the inputs and output must be electrically isolated to prevent the possibility of fault currents.

The bandwidth of the amplifier is extremely limited, as the small signal bandwidth is 1100 Hz and the large signal bandwidth is 700 Hz. While this bandwidth is quite low, it is usually sufficient for most isolation amplifier applications (such as ECG measurements). As with the instrumentation amplifier, the input offset voltage can be trimmed and removed by software compensation.

Adjustments:

The offset voltage is trimmed just as for the instrumentation amplifier. Care should be taken as the adjustment has a time constant of several minutes.

Functional Description:

Neon lamps DS1 and DS2 provide input overvoltage protection. Module A2 is the input isolation amplifier. Amplifier module A1 provides the circuit's variable gain under the control of switch S1. The dual comparator IC4 level shifts the TTL control signals for shorting and disconnecting the isolation amplifier inputs. The zener diodes CR3 and CR4 provide voltage limiting.

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

A1	amplifier module, previously described
A2	isolation amplifier module, AD284J
C1-C9	cap, cer disk, 0.1uF, 16 V, ±20%
C10-C11	cap, cer disk, 0.022uF, 20 V, ±20%
C12-C17	cap, tant, 3.3uF, 35 V, ±20%
CR1-CR2	diode, 1 A, 400 PIV, 1N4004
CR3-CR4	diode, zener, 9.1 V, 500 mW, 1N5239
DS1-DS2	lamp, neon, NE2
IC1	voltage regulator, +15 V, LM340T-15
IC2	voltage regulator, +8 V, LM340T-8
IC3	voltage regulator, -8 V, LM320T-8
IC4	dual comparator, LM319
K1-K2	relay, dry reed, DPDT, 12 Vdc, 615 ohm coil
Q1-Q2	transistor, NPN, small signal, 2N2222
R1	res, car film, 5M, 1/2 W, 1%
R2	res, 56 ohm
R3	res, variable, cermet, 20k, 10 turn, 1/4 W
R4-R5	unused
R6-R7	res, 220 ohm
R8-R9	res, 3.3k
R10-R11	res, 6.8k
R12	res, 12k
R13-R14	res, 10k
R15	res, 2.7k
S1	switch, thumbwheel, SP10T, decimal

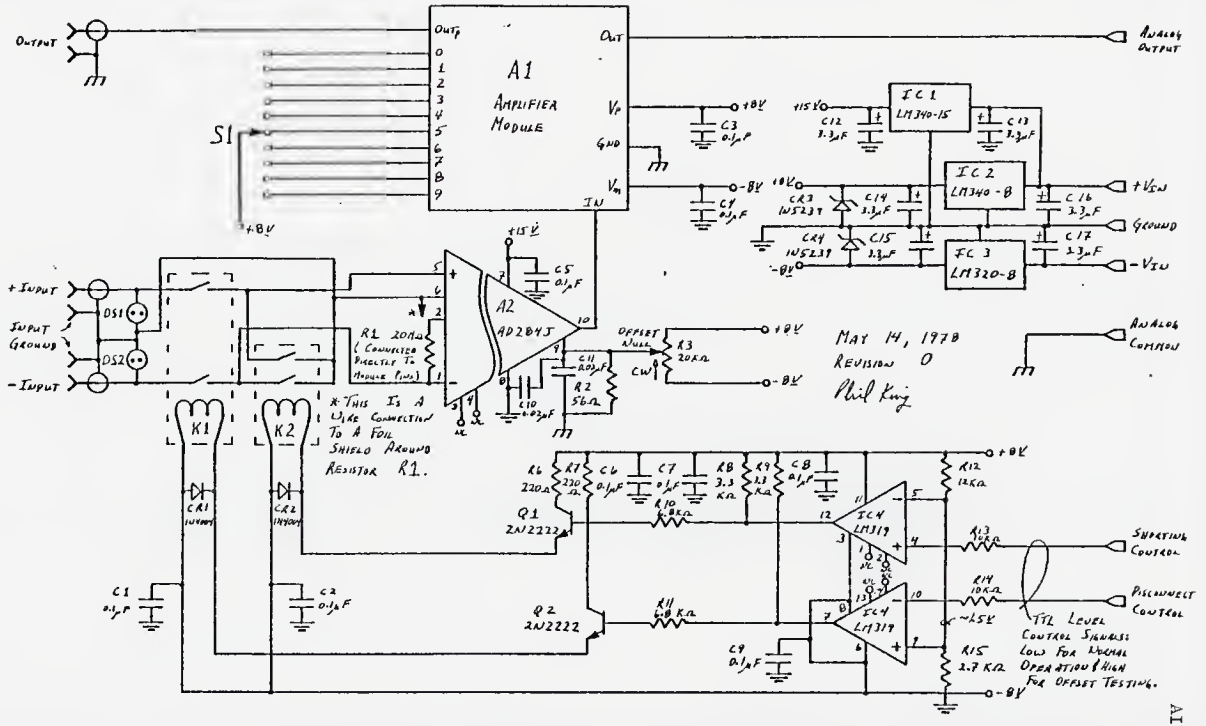


Figure AII.7. Schematic of Isolation Amplifier

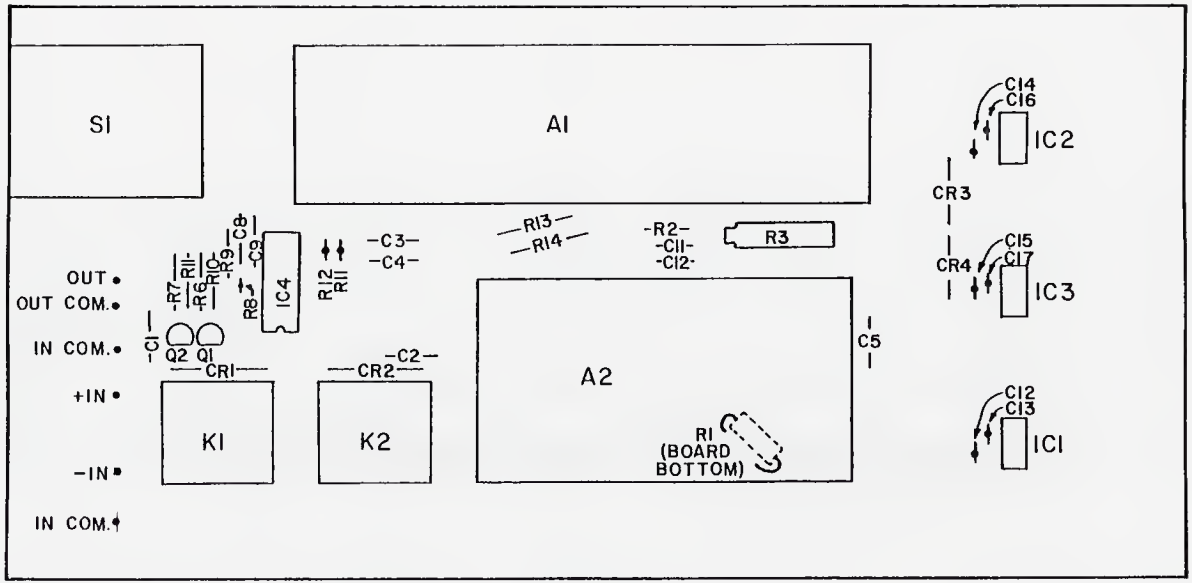


Figure AII.8. Parts Placement of Isolation Amplifier

AII.10. D/A Converter Module - 7522 Version

The purpose of this module is to convert data from the microprocessor to an analog signal. This can be used for displaying the input signal to check for system errors, or it could be used to stimulate the unit being measured.

Some output protection is provided by the 2.2 k-ohm output resistor and in the case of a serious overload, the LM741 op amp would probably be the only component damaged.

Suggestions:

The AD7522 converter provides good results but is a bit expensive and requires numerous external components. If it is desired to add several more D/A converters, it is suggested that the user design a new board around one of the new, low cost converters such as the Signetics NE5018.

Adjustments:

To trim the offset voltage of the LM747 op amp, the AD7522 must be removed. The output of each of the amplifiers is then connected to its inverting input with a 10 k-ohm resistor. After these preparations, the offset is trimmed in the usual manner.

Functional Description:

IC1 is a monolithic, current output D/A converter. The dual op amp IC2 provides current to voltage conversion, and allows the converter to be used to provide a bipolar output. Op amp IC3 is an output buffer

providing a gain of 0.5, so that the output signal is ±5 volts, making it compatible with the A/D converter input. Resistor R9 provides limited output protection.

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

C1-C4	cap, cer disk, 0.1uF, 16 V, ±20%
C5-C6	cap, cer disk, 100pF, 20 V, ±20%
CR1-CR4	diode, Schottky, small signal, MB07Ø1
IC1	digital to analog converter, AD7522JD
IC2	dual op amp, LM747
IC3	op amp, LM741
R1-R2	res, variable, cermet, 500 ohm, 10 turn, 1/4 W
R3-R4	res, variable, cermet, 10k, 10 turn, 1/4 W
R5-R6	res, car film, 5.1k, 1/4 W, 1% matched to 0.1%
R7	res, 10k
R8	res, 5.1k
R9	res, 2.2k

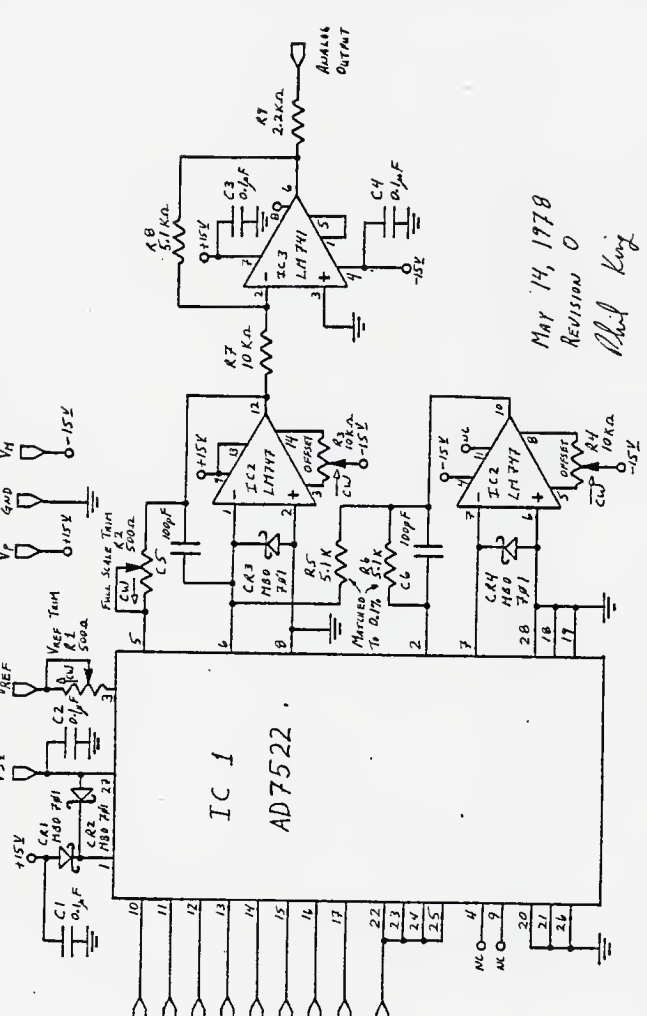


Figure AII.9. Schematic of D/A Converter Module - 7522 Version

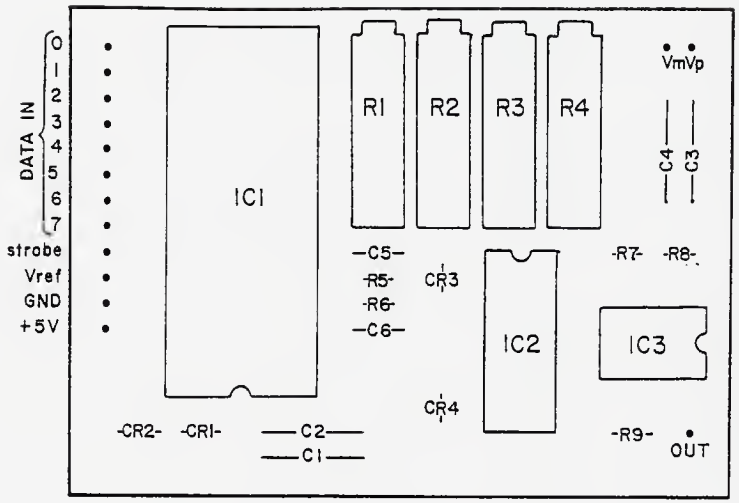


Figure AII.10. Parts Placement of D/A Converter Module - 7522 Version

AII.11. D/A Mother Board

The D/A mother board provides power supplies and interface to the processor for up to four D/A converter modules. Each of the converters is independently addressable.

Functional Description:

The 5 volt regulator IC3 is powered from the 15 volt regulator IC4 to help meet the D/A converter (AD7522) requirement that the digital supply voltage never exceed the analog supply voltage. The 3 line to 8 line decoder IC1, along with jumpers, selects which of the eight possible D/A converters are selected when data is sent from the microprocessor.

Parts List:

A1-A4	D/A converter module, previously described
C1-C6	cap, tant, 3.3uF, 35 V, ±20%
IC1	TTL 3 line to 8 line decoder, 74LS138
IC2	voltage regulator, +10.00 V, LH0070
IC3	voltage regulator, +5 V, LM340T-5
IC4	voltage regulator, +15 V, LM340T-15
IC5	voltage regulator, -15 V, LM320T-15

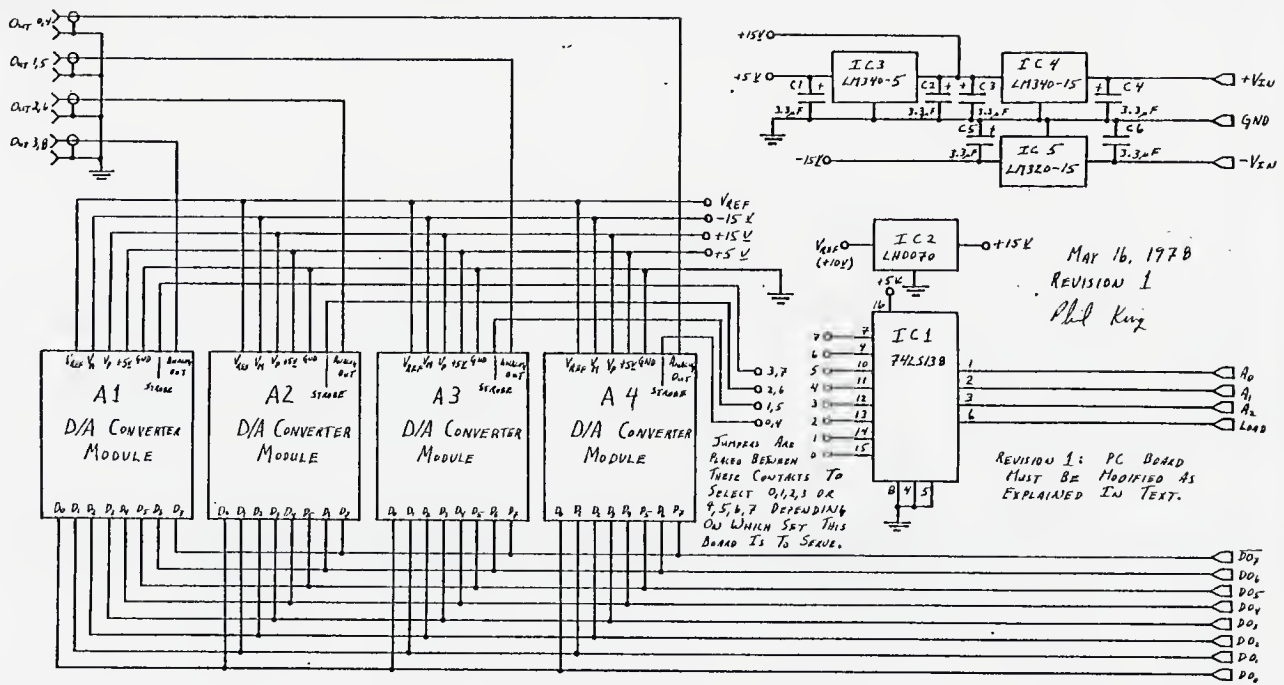


Figure AII.11. Schematic of D/A Mother Board

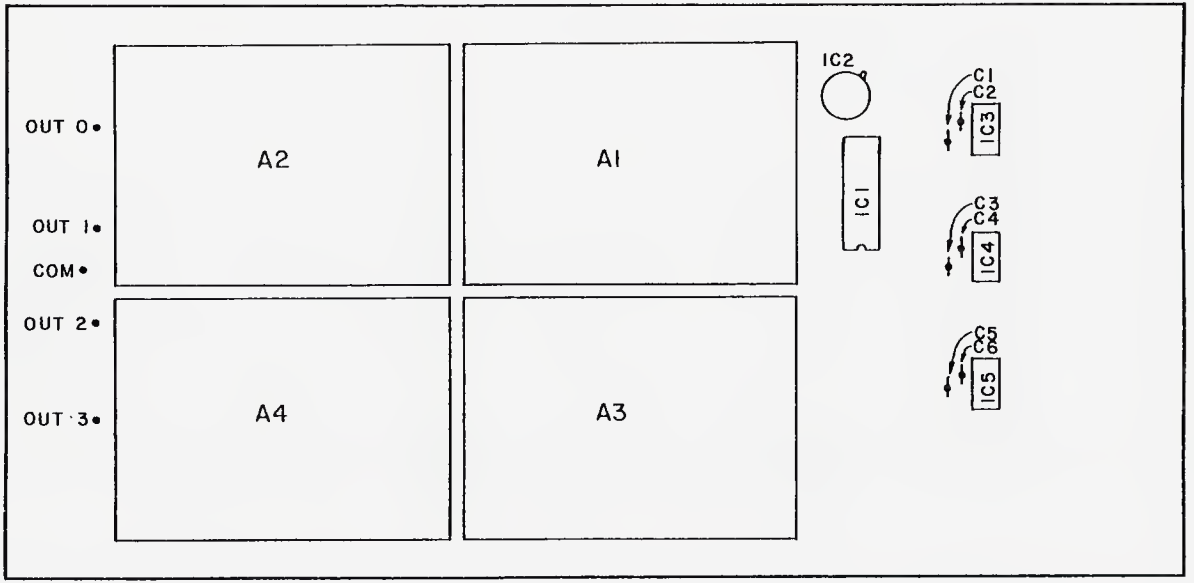


Figure AII.12. Parts Placement of D/A Mother Board

AII.12. Keyboard

The keyboard allows the user to enter data and control signals to the system. It is connected to the front panel board which interfaces it to the microprocessor.

Keyboard Encoding:

key number	key top (function)
1	reset
2	pause
3	run
4	7
5	8
6	9
7	-unused-
8	-unused-
9	?
10	4
11	5
12	6
13	-unused-
14	-unused-
15	end of channel order list
16	1
17	2
18	3
19	-unused-
20	-unused-
21	cont.
22	0
23	0
24	enter

Keyboard Layout:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Functional Description:

The comparator IC1, if installed, provides a power-up reset. This is not needed when the M.S.P. Lucas Z-80 board is used, as the power-up reset function is implemented on it.

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

C1	cap, tant, 10uF, 10 V, ±20%
CR1	diode, small signal, 1N914
IC1	comparator, LM311
IC2	TTL tri-state octal inverter, 81LS98
R1-R8	res, 10k
R9	res, 100k
R10	res, 1.2k
R11	res, 4.7k
R12	res, 2.2k
S1-S24	switch, SPST-MC, low bounce keyboard packaged in groups of four

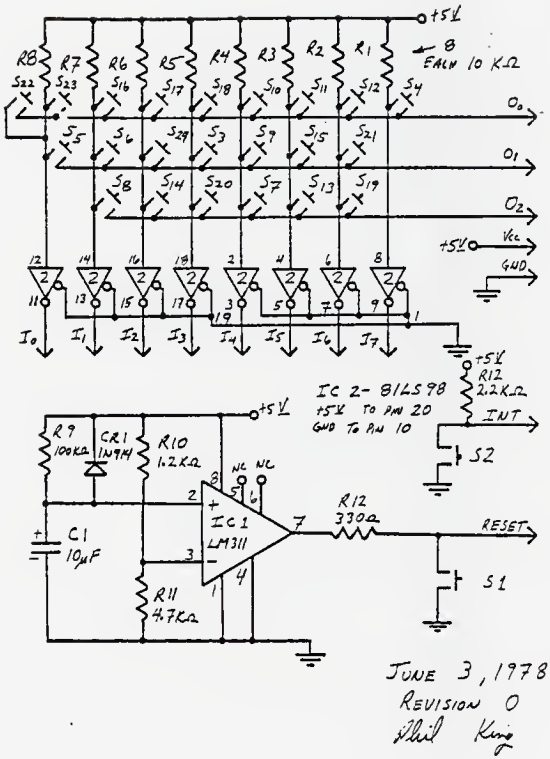


Figure AII.13. Schematic of Keyboard

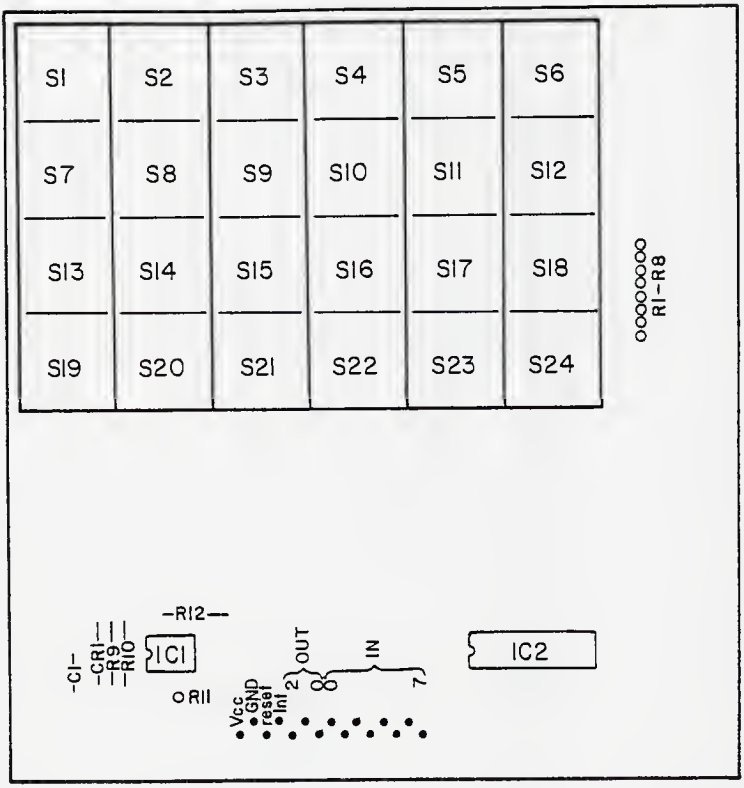


Figure AII.14. Parts Placement of Keyboard

AII.13. Front Panel Board

The front panel board serves two major purposes. First, it supports the EPROM's which contain the instrument's operating system software. Secondly, it provides a 7-1/2 digit display for user interface and the support hardware for the keyboard.

The EPROM addresses can be set to any of the 64 1-kilobyte sections of memory addressable by the processor. To prevent interference with other devices and to insure proper start-up, it is recommended that the EPROM address be the lowest section of memory (0000H to 03FFH).

The display is located on I/O ports F0H to FFH. Using 16 I/O ports allows addressing each of the 8 display digits with two character sets. The keyboard is located on I/O port 00H.

Suggestions:

Electronics designs become rapidly outdated, due largely to denser chips in the case of digital designs. Unfortunately, the front panel board was designed using parts on hand, and therefore was outdated before it was finished. There are now much more advanced parts available (such as the Intel 8279 keyboard/display controller) which would make a redesign of the board almost trivial.

Functional Description - edge card connector and buffers:

For use with the system software, the jumper for keyboard interrupts should be left out.

Functional Description - EPROM's:

The seven bit dip switch allows the EPROM addresses to be changed. An open switch provides a one for that bit and a closed switch provides

a zero for that bit. When jumper pair A is in place, setting-up the board for decoding 4 1702-EPROM's, the EPROM's decode for the upper 6 bits of address being as specified by the dip switch. For jumper pair B installed, two 1702's are decoded for the upper 7 bits of address being the same as the dip switch setting.

Function Description - display:

The display operates in two modes: addressed from the processor, and free-running. Both modes are described below, with the free-run mode described first.

Briefly in the free-run mode the display consists of two 8 location memories driving the 8 digit display through a character generator on a bank switch basis. This allows the display to produce 32 characters using memories which could generally only address 16 characters in the character generator. This is described in more detail below.

The 2112 RAM is used as a 16 by 4-bit RAM. Two locations are assigned to each display digit. Each location contains one character which is written to the associated display digit. Each of the two RAM locations allows a different character set to be accessed. Each character set contains a blank character, and when a character is chosen from one set for display a blank is generally chosen as the character in the alternate RAM location for that display digit.

The 82S123 is a 32 by 8-bit PROM used for character generation. It contains two character sets of sixteen characters each (one blank and fifteen others). The 2112 and 82S123 shift between upper and lower character sets synchronously. The output of the 2112 selects which address in the 82S123 is accessed, thereby choosing the character. The output of the

82S123 is buffered to drive the display segments. A zero on the output of the 82S123 turns off the associated segment of the display digit, and a one on the output line turns the segment on.

The display is multiplexed in synchronism with the locations in the 2112 being accessed. The proper digit is selected by the 74LS138 demultiplexer along with the drive transistors.

The LM555 is used as an astable multivibrator to provide a clock for the 74LS93 counter. The counter controls the selection of the character set, and synchronizes the selection of the location in the 2112 RAM with turning on the proper display digit.

When addressed by the processor, the functions described above cease and the 2112 RAM is configured to be written into by the microprocessor.

Display Character Sets:

port address table

display digit	0	1	2	3	4	5	6	7
lower char. set	F0	F2	F4	F6	F8	FA	FC	FE
upper char. set	F1	F3	F5	F7	F9	FB	FD	FF

(note: display digit 0 is the leftmost digit)

display character resulting from hex data written to a display digit

data	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
lower set	0	1	2	3	4	5	6	7	8	9	A	b	C	d	.	␣
upper set	E	F	H	h	n	o	P	r	u	?	-	L	␣	␣	.	␣

(␣ = blank)

Contents of 82S123 PROM:

address binary A4--A0	hex	output B0 -- B7 (segments) (ABCDEFG.)	character displayed
00000	00	11111100	0
00001	01	11111110	8
00010	02	01100110	4
00011	03	10011100	C
00100	04	11011010	2
00101	05	11101110	A
00110	06	10111110	6
00111	07	00000001	.
01000	08	01100000	1
01001	09	11110110	9
01010	0A	10110110	5
01011	0B	01111010	d
01100	0C	11110010	3
01101	0D	00111110	b
01110	0E	11100000	7
01111	0F	00000000	blank
10000	10	10011110	E
10001	11	00111000	u
10010	12	00101010	n
10011	13	00000000	blank
10100	14	01101110	H
10101	15	00000010	-
10110	16	11001110	P
10111	17	00000001	.
11000	18	10001110	F
11001	19	11001010	?
11010	1A	00111010	o
11011	1B	00000000	blank
11100	1C	00101110	h
11101	1D	00011100	L
11110	1E	00001010	r
11111	1F	00000000	blank

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

C1	cap, cer disk, 0.1uF, 16 V, ±20%
C2	cap, cer disk, 0.01uF, 20 V, ±20%
C3-C5	cap, cer disk, 0.1uF, 16 V, ±20%
C6-C8	cap, tant, 10uF, 25 V, ±20%
C9-C16	cap, cer disk, 0.1uF, 16 V, ±20%
C17-C19	cap, tant, 10uF, 25 V, ±20%
CR1	diode, 1 A, 400 PIV, 1N4004
CR2-CR5	diode, LED, KC556R
IC1-IC4	EPROM, 1702A
IC5	voltage regulator, -8 V, LM320T-8
IC6-IC7	TTL hex inverter, open collector, 7406
IC8-IC9	TTL 3 line to 8 line decoder, 74LS138
IC10	unused
IC11-IC12	TTL 4-bit comparator, 7485
IC13	astable multivibrator, LM555
IC14	TTL 4-bit binary counter, 74LS93
IC15	TTL quad 2-input NOR gate, 74LS02
IC16	TTL quad 2-input NAND gate, 74LS00
IC17	TTL hex buffer, open collector, 7407
IC18	PROM, 82S123
IC19	TTL tri-state hex buffer, 8T97
IC20	TTL hex buffer, open collector, 7407
IC21	RAM, 2112
IC22-IC23	unused
IC24	TTL tri-state hex buffer, 8T97
IC25	TTL dual D flip-flop, 74LS74
IC26	TTL quad 2-input NOR gate, 74LS02
IC27	TTL quad 2-input NAND gate, 74LS00
IC28	TTL dual 5-input NOR gate, 74LS260
IC29-IC31	TTL tri-state hex buffer, 8T97
IC32-IC33	tri-state latched octal buffer, 8212
IC34	TTL tri-state hex buffer, 8T97
IC35-IC36	voltage regulator +5, LM340T-5
DS1	LED display, 7-segment, Litronix DL722
DS2-DS4	LED display, 7-segment, Litronix DL728
Q1-Q8	transistor, NPN, small signal, 2N2222
R1-R8	res, 820 ohm
R9	res, 10k
R10	res, 1.2k
R11-R20	res, 2.2k
R21-R28	res, 120 ohm
R29	res, 2.2k
R30-R31	res, 180 ohm
R32-R33	unused
R34-R35	res, 1k
R36-R43	res, 2.2k
R44	res, wire wound, 1 ohm, 1 W, 5%
R45-R47	res, 2.2k
S1-S7	switch, 7-segment dip
S8-S14	switch, 7-segment dip

IC power connections:

	+5V	GND
IC6	14	7
IC7	14	7
IC8	16	8
IC9	16	8
IC11	16	8
IC12	16	8
IC14	5	10
IC15	14	7
IC16	14	7
IC17	16	8
IC19	16	8
IC20	16	8
IC24	16	8
IC25	14	7
IC26	14	7
IC27	14	7
IC28	14	7
IC29	16	8
IC30	16	8
IC31	16	8
IC34	16	8

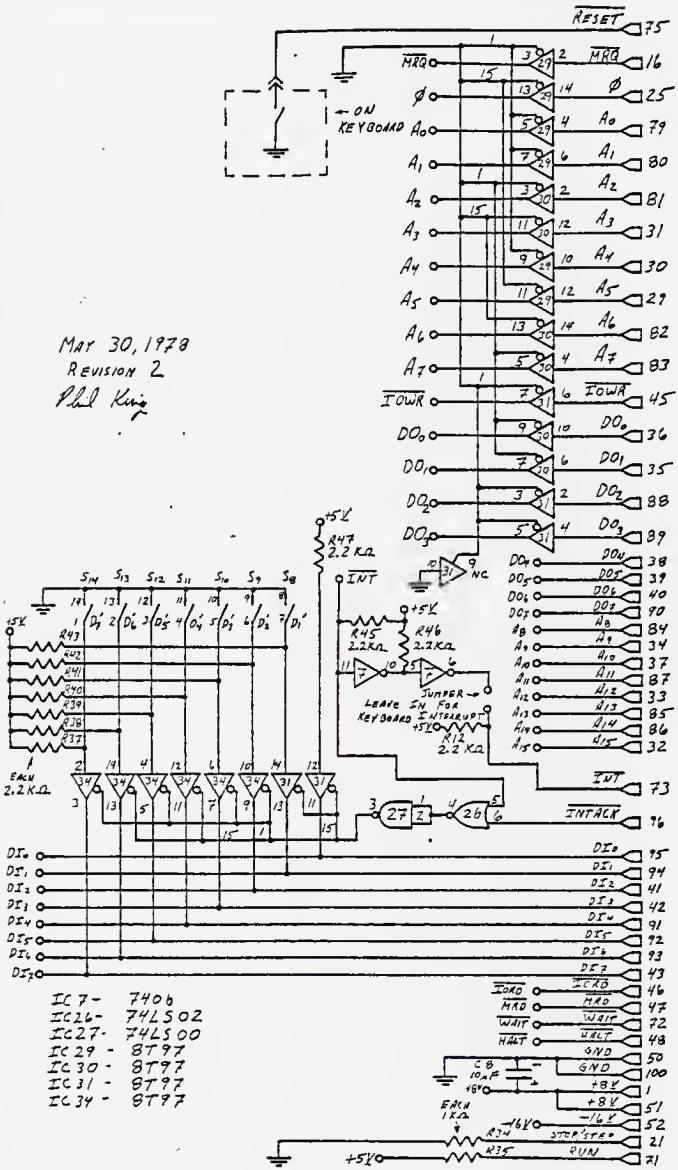


Figure AII.15. Schematic of Front Panel Board - Edge-Card Connector & Buffers

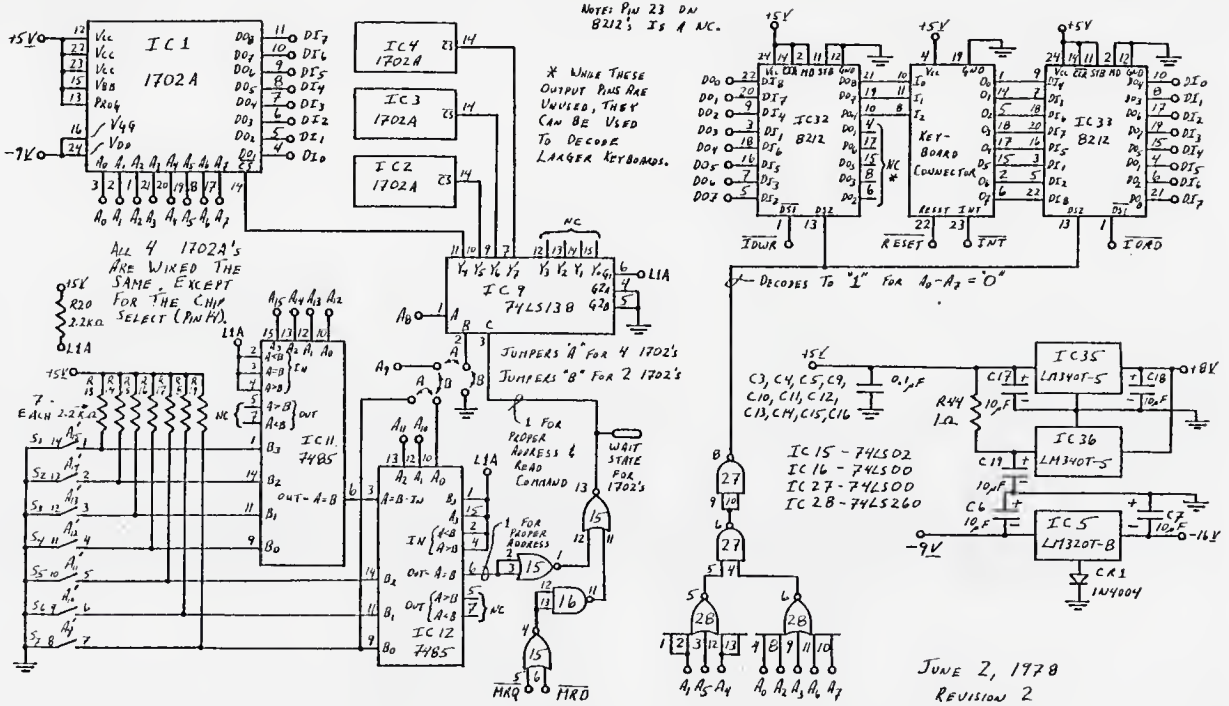


Figure AII.16. Schematic of Front Panel Board - PROM, Keyboard I/O, & Power Supply

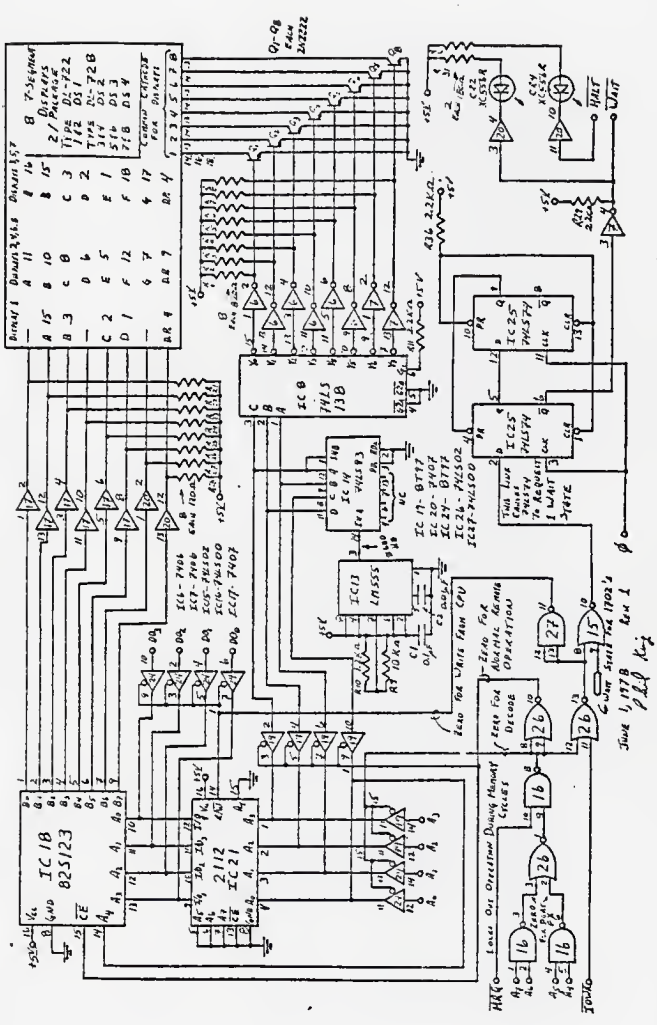


Figure AII.17. Schematic of Front Panel Board - Display

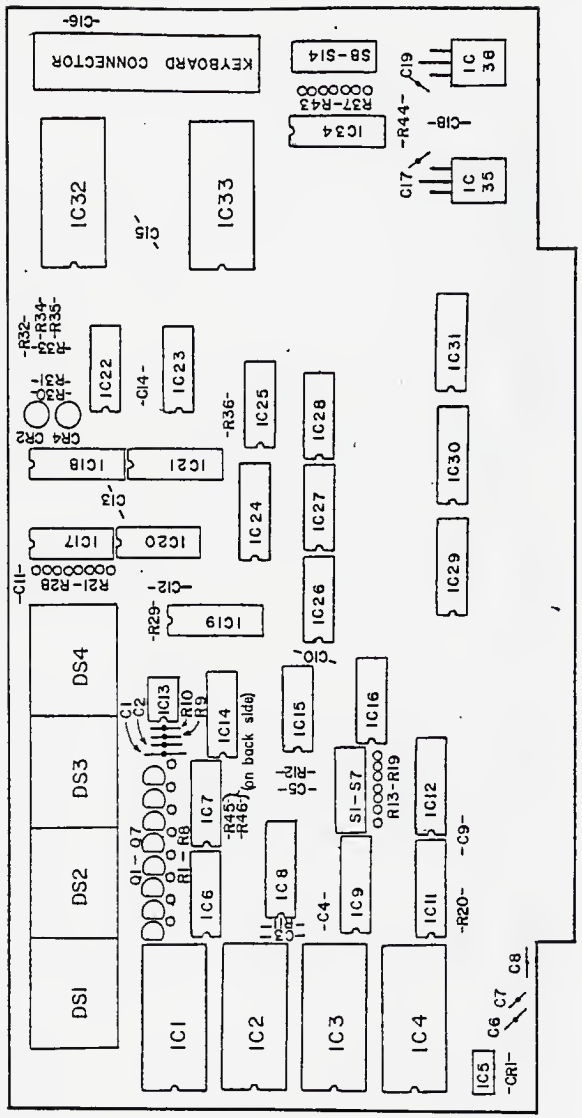


Figure AII.18. Parts Placement of Front Panel Board

AII.14. A/D Converter Board

The major purpose of the A/D converter board is to convert the analog signals from the amplifiers to a 2's complement digital representation. It also provides control signals for the amplifiers and the D/A converters.

The board can be used with either the standard S-100 bus or the M.S.P. Lucas modified S-100 bus. The bus type is jumper selectable as indicated on the DAC interface and amplifier control schematic.

When the A/D converter section is addressed by the processor, the data from the previous conversion is placed on the data bus, and the sample-and-hold is placed in the sample mode. After a short delay, the sample-and-hold is placed in the hold mode and the conversion is started.

The converted data is checked for a possible overrange condition by testing for a result which is $\pm$ full scale. If an overrange occurs, it is indicated by a pair of LED's on the front panel. One LED lights momentarily to indicate that an overrange has just occurred, and the other LED is latched to indicate that an overrange has occurred since the run key was pressed.

The D/A converters on ports 40H to 47H are supported by the system as originally configured. The port addresses can be changed by jumpers, but care should be exercised to avoid conflicting addresses.

On port EF, which controls the shorting and disconnect capabilities of some amplifiers, only bits 0 and 1 are important. Bit 0 controls the disconnect features and bit 1 controls the shorting feature. When these lines are low ("0"), the amplifiers are in their normal operating mode (amplifiers unshorted and connected to the front panel connectors). Since the relays do not switch instantaneously, care should be taken in sequencing

these controls to avoid shorting the signal source connected to the amplifier due to differences in relay reaction time. Software delays should be used between relay operations.

Adjustments:

To adjust the offset voltage trim on the sample-and-hold, the MP-20 module and IC5 must be removed, and pins 2, 14, and 13 must be shorted together on the sample-and-hold. After sufficient warm-up, R38 should be adjusted for zero volts on pin 7 of the sample-and-hold.

Functional Description - DAC interface & amplifier controls:

When the port address for one of the D/A converters or the amplifier controls appears on the address bus along with a "0" on the $\overline{\text{IOWR}}$ line, the data on the data bus is transferred to the converter or the amplifier control lines as appropriate.

The port addresses of the D/A converters can be changed by using the dip switch connected to IC11.

For this system, in which DMA is not used, the DMA jumper set D should be used. For use with a standard S-100 bus CPU, jumper set C should be used.

Functional Description - sample-and-hold:

When the A/D converter is addressed by the microprocessor, a one-shot, IC5, is triggered. The output of the one-shot places the sample-and-hold amplifier IC4 in the sample mode for the duration of the pulse (approximately 7 usec).

Parts List:

resistors are carbon composition, 1/4 W, 5% unless specified

A1	A/D converter module, Burr-Brown MP-20
C1-C6	cap, tant, 3.3uF, 35 V, ±20%
C7-C12	cap, cer disk, 0.1uF, 16 V, ±20%
C13	cap, cer disk, 0.0022uF, 20 V, ±10%
C14-C17	cap, cer disk, 0.1uF, 16 V, ±20%
C18	cap, elect, 47uF, 50 V, ±20%
C19	cap, polystyrene, 0.001uF, 50 V, ±10%
CR1-CR32	diode, 1 A, 400 PIV, 1N4004
CR33-CR36	diode, zener, 9.1 V, 500 mW, 1N5239
CR37-CR38	diode, LED, XC5556R (same as CR3 & CR5 on front panel board)
IC1	voltage regulator, +15 V, LM340T-15
IC2	voltage regulator, -15 V, LM320T-15
IC3	voltage regulator, +5 V, LM340T-5
IC4	sample-and-hold, Datel SHM-IC-1
IC5	TTL monostable multivibrator, 74123
IC6	TTL quad 2-input NAND gate, 74LS00
IC7-IC9	TTL tri-state hex buffer, 8T97
IC10	TTL dual D flip-flop, 74LS74
IC11	TTL 4-bit comparator, 74LS85
IC12	TTL duad D flip-flop, 74LS74
IC13	TTL 8-input NAND gate, 74LS30
IC14	TTL quad 2-input NAND gate, 74LS00
IC15	TTL 8-input NAND gate, 74LS30
IC16	TTL quad 2-input NOR gate, 74LS02
IC17	TTL dual 5-input NOR gate, 74LS260
IC18	TTL quad 2-input NAND gate, 74LS00
R1-R16	res, 5.1k
R17	res, 1k
R18-R19	res, 270 ohm
R20	res, 1k
R21	res, 100k
R22-R23	res, 1k
R24	res, 10k
R25	res, 1k
R26	res, 10k
R27	unused
R28-R32	res, 2.2k
R33-R36	res, 10k
R37	unused
R38	res, variable, cermet, 100k, 10 turn, 1/4 W
S1-S4	switch, 4-bit dip (may be replaced with jumpers)

IC power connections:

	+5V	GND
IC5	16	8
IC6	14	7
IC7	16	8
IC8	16	8
IC9	16	8
IC10	14	7
IC11	16	8
IC12	14	7
IC13	14	7
IC14	14	7
IC15	14	7
IC16	14	7
IC17	14	7
IC18	14	7

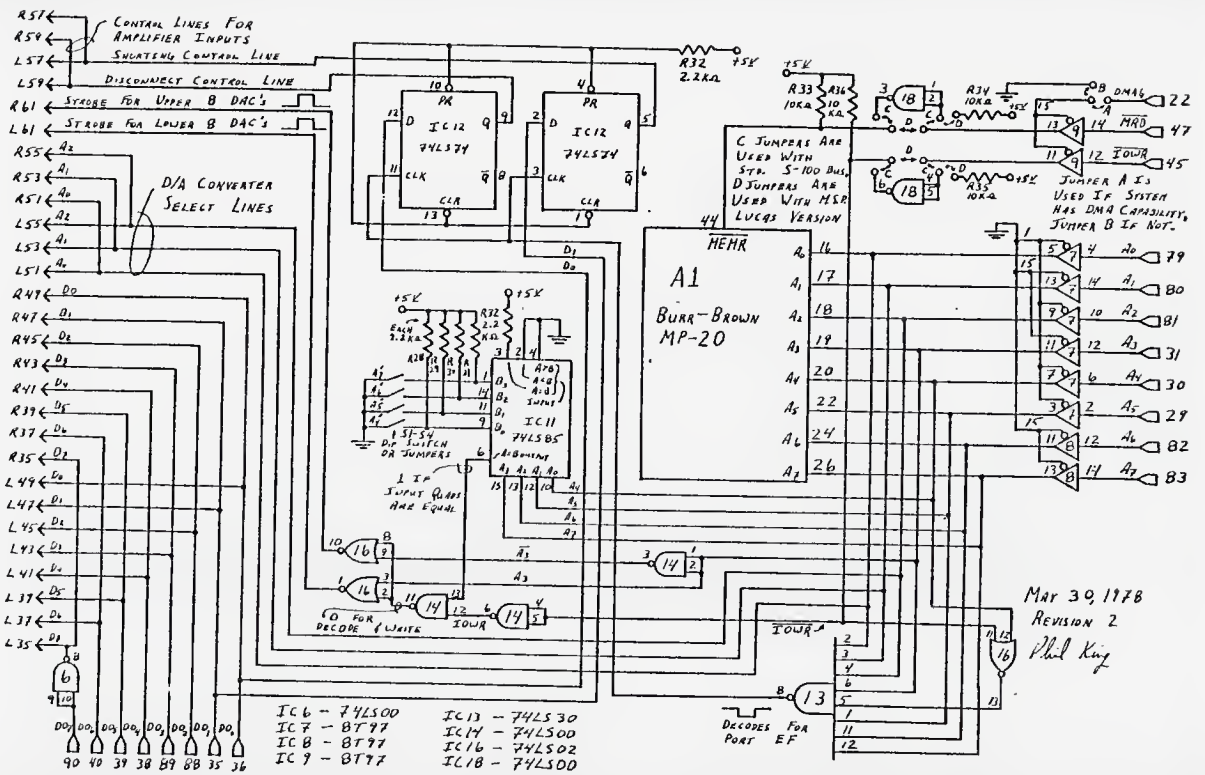


Figure AII.19. Schematic of A/D Board - DAC Interface & Amplifier Controls

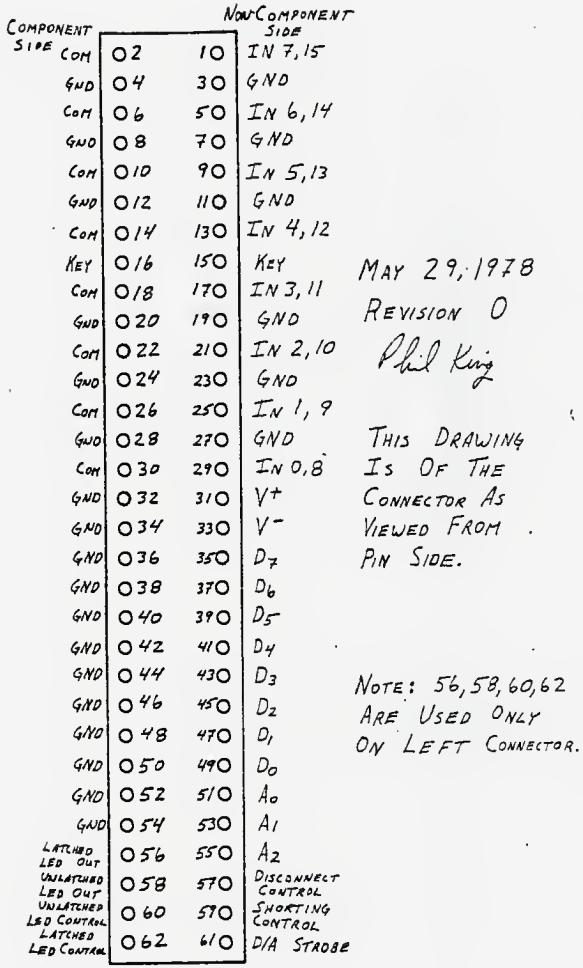


Figure AII.20. Schematic of A/D Board - Analog Connectors

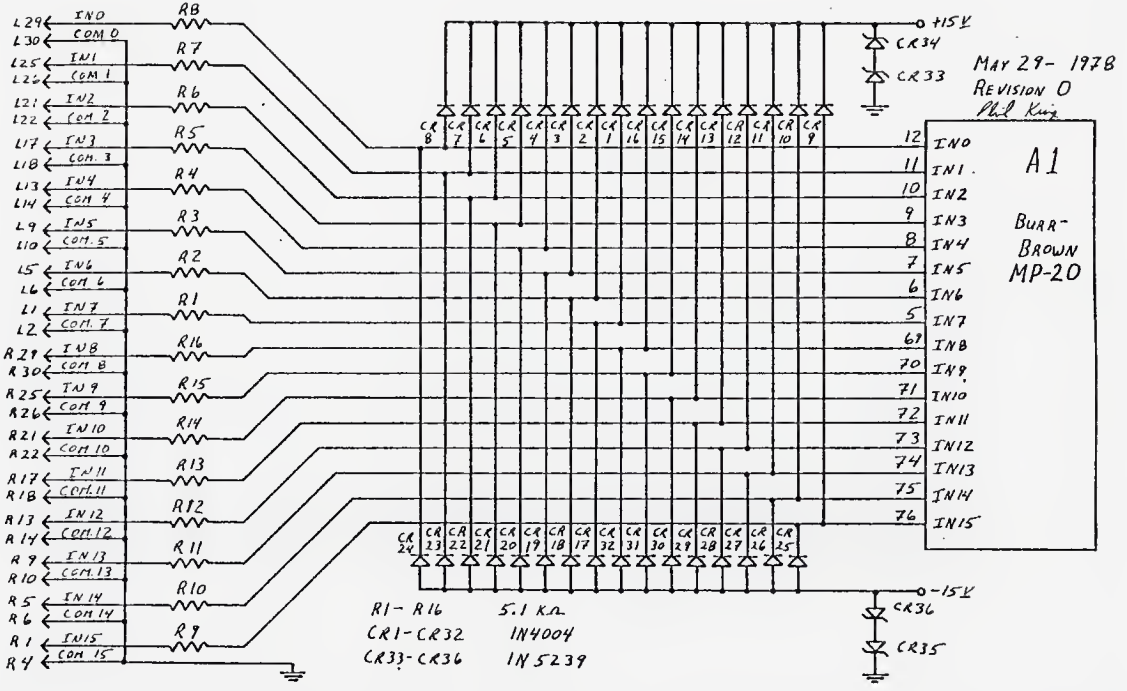


Figure AII.21. Schematic of A/D Board - Input Protection

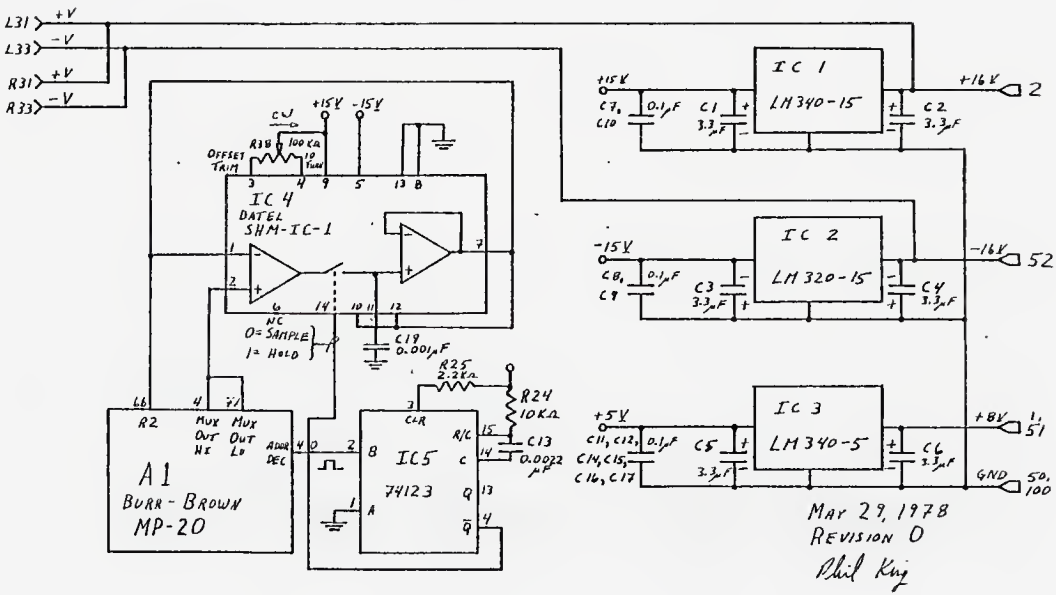


Figure AII.22. Schematic of A/D Board - Sample-and-Hold & Power Supplies

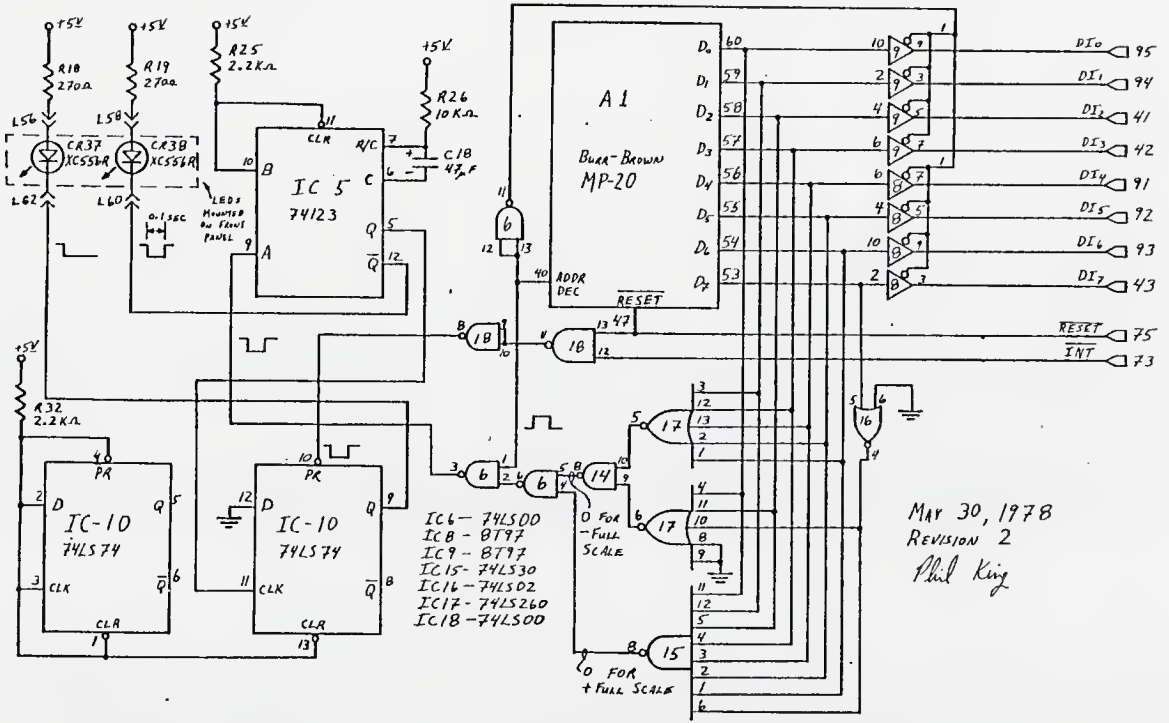


Figure AII.23. Schematic of A/D Board – Overrange Detection & Indication

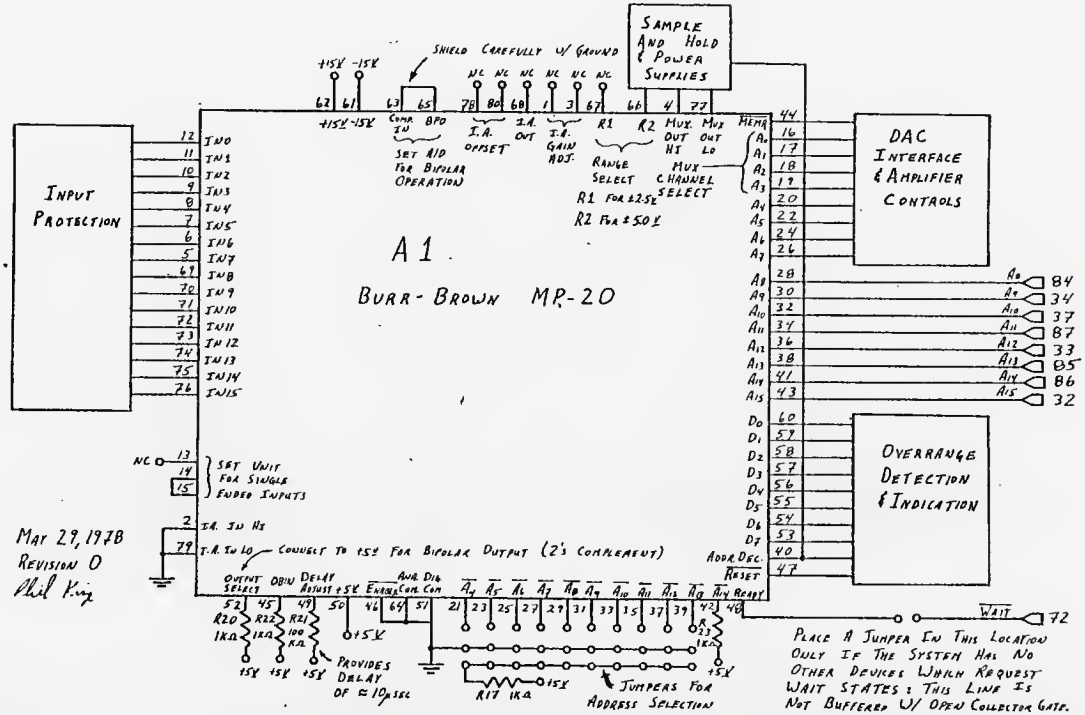


Figure AII.24. Schematic of A/D Board - MP-20 Detail

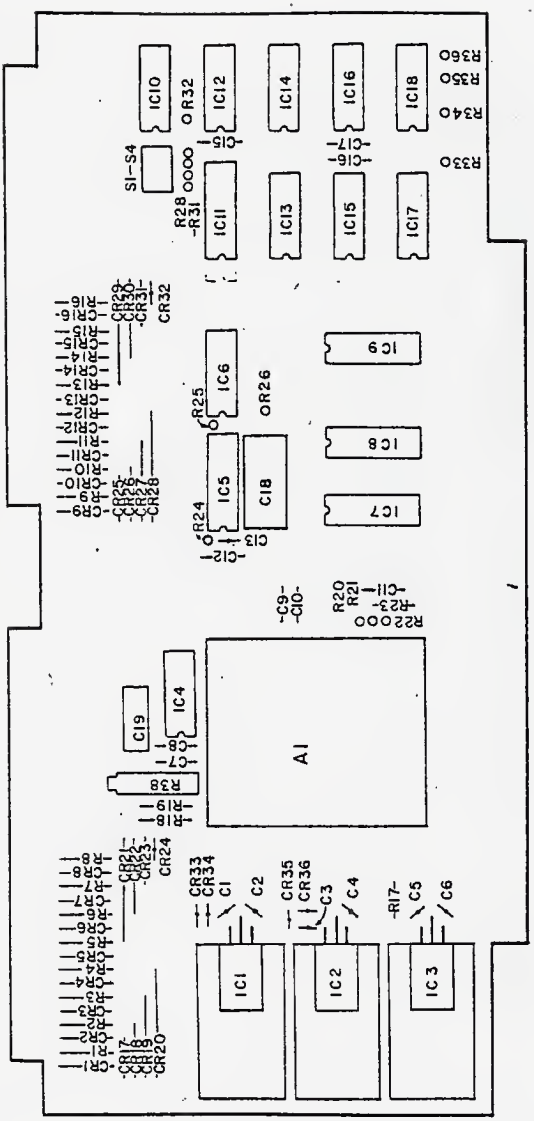


Figure AII.25. Parts Placement of A/D Board

AII.15. Power Supplies

The purpose of the power supplies is to provide high current, unregulated power to the three supplies in the system: the logic supply, and the positive and negative analog supplies. All supplies are then regulated to the proper voltage by local, on-board regulators. Bleed-down resistors discharge the filter capacitors to safe levels (200 mV max) in less than 5 minutes.

Suggestions:

If the system is expanded, the +5 V logic supply must not be required to supply more than 5 A in order to insure that the unregulated voltage is high enough to keep the IC voltage regulators operating properly. One solution, if more current is required, is to replace the diode bridge, CR5-CR8, with Schottky diodes having a much lower forward drop under conditions of large forward current.

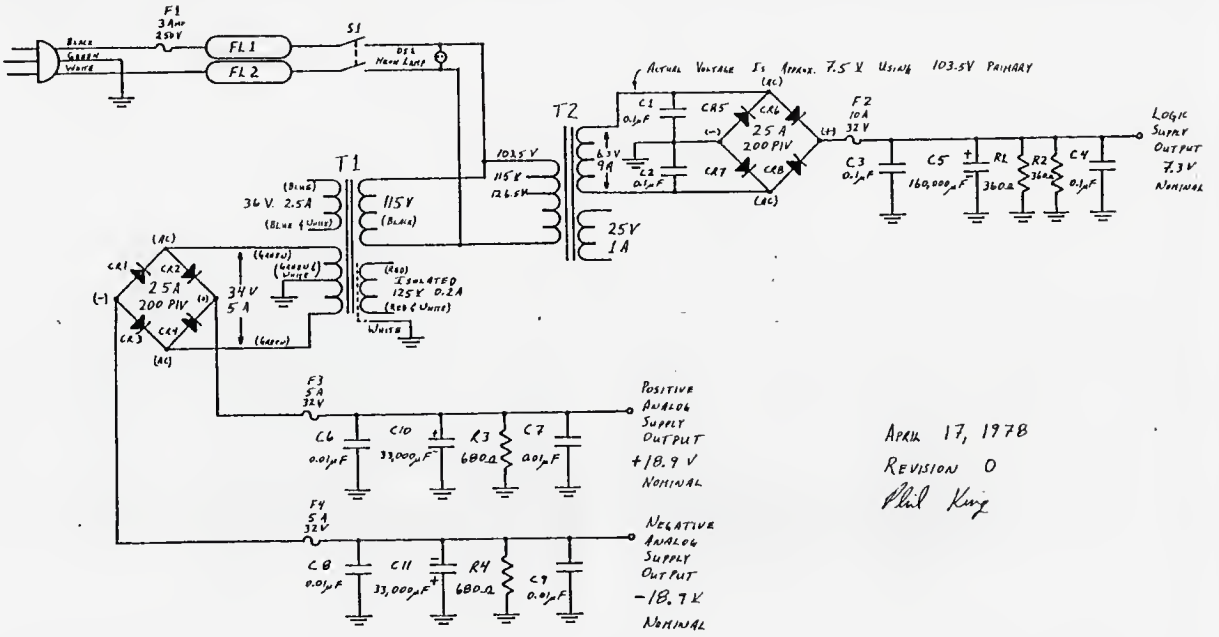
Parts List:

C1-C4	cap, cer disk, 0.1uF, 16 V, ±20%
C5	cap, elect, 160,000uF, 20 V
C6-C9	cap, cer disk, 0.01uF, 100V, ±20%
C10-C11	cap, elect, 33,000uF, 40 V
CR1-CR4	diode bridge, 25 A, 200 PIV
CR5-CR8	diode bridge, 25 A, 200 PIV
DS1	panel lamp, neon, 120 Vac (Radio Shack #272-705)
F1	fuse, 3 A, 250 V, AGC3
F2	fuse, 10 A, 32 V, AGC10
F3	fuse, 5 A, 32 V, AGC5
FL1-FL2	line filter, 5 A, 125 Vac, 400 Hz (Gudeman FLO-155-2)
S1	switch, DPDT, toggle, 250 Vac, 5 A
T1	transformer, 34 Vct, 5 A
T2	transformer, 7.5 V, 9 A

Test Data:

Power Supply	Test Load	Transformer Output	Supply Output	Ripple Voltage
Logic	None	7.64 Vac	9.24 Vdc	
	2.0 ohms	7.50 Vac	7.50 Vdc	
	1.5 ohms	7.52 Vac	7.30 Vdc	200 mV
	1.0 ohms	7.42 Vac	6.87 Vdc	
Positive Analog	None	34.3 Vac	23.3 Vdc	
	6.4 ohms	33.3 Vac	19.0 Vdc	
	4.8 ohms*	31.8 Vac	17.9 Vdc	400 mV
Negative Analog	None	34.3 Vac	-23.2 Vdc	
	6.4 ohms	33.3 Vac	-18.8 Vdc	
	5.2 ohms*	31.8 Vac	-17.7 Vdc	500 mV

* These two tests were run simultaneously. All others were run with the other supplies unloaded.



April 17, 1978
Revision 0
Phil King

Figure AII.26. Schematic of Power Supply

AII.16. Other Boards

There are several boards in the data acquisition system which I did not design. These are discussed in brief below, though complete documentation is not given.

CPU Board:

The CPU board is a Z-80 board designed by Dr. M. S. P. Lucas at Kansas State University. It is compatible with the S-100 (Altair) bus, with the exception of the memory read and write and I/O read and write control lines which are active low. More information on this board is available from Dr. Lucas.

4k RAM Board:

The RAM board was purchased from S.D. Sales Company (P.O. Box 28810, Dallas, Texas 75228). It is their standard S-100 bus compatible 4k RAM board. To make it compatible, with the M.S.P. Lucas modified S-100 bus, IC39 (74S00) was replaced with a 74LS02.

Parallel Interface:

The parallel interface board, built by John Schmalzel, takes the channel number information and the data from the conversion and places them on a 16-bit parallel interface bus. Information on the use of this board is given in the user manual, and more information on the design is available from Dr. R. R. Gallagher (Elect. Engg. Dept., KSU).

IEEE-488 Bus Interface:

This interface, built by Sam Babb, takes the channel number information and the conversion data from the parallel interface and passes them to a computer in byte serial mode, according to the requirements of the IEEE-488 bus standard. Information on the use of the interface is given in the user manual, and more information on the design is available in reference 2.

AII.17. Protection Circuitry

All amplifier inputs and outputs and the inputs to the MP-20 module are protected against damage due to overvoltage. This protection circuitry limits the voltage and current which reach the protected components, and--in the case of a severe overload--a resistor will be sacrificed to save the more expensive components. The protection is effective to 170 volts peak ac or dc (120 Vacrms).

All inputs and outputs are first connected to a resistor which is clamped to the power supplies at the protected device end. The diode clamps to the supplies provide voltage limiting, and the resistor provides current limiting. For protection when the unit is turned off, the power supplies are equipped with zener diode clamps on their output terminals. Additional protection for the instrumentation and isolation amplifiers is provided by the reed relays on their inputs which open when power is disconnected, and by the neon lamps across their inputs. (Note that the isolation amplifier does not have all of the protection circuitry described, because its floating input stage provides excellent overvoltage protection.)

AII.18. Requirements for Added Boards

It is likely that users will want to add custom analog plug-in modules to meet their specific requirements. The following provides information on the system's requirements and provisions for plug-in modules.

power supply:
- ±20 V (approximate) unregulated
- not to exceed 250 mA per slot worst case

analog output from amplifier slot plug-in:
- one analog output line and analog common
- output signal level ±5 Vpp
- analog common is low current signal return

control and data lines:
- plug-in should provide no more than one LSTTL load
- logic levels are TTL

uncommitted lines:
- two uncommitted lines are provided for user assignment
- uncommitted lines are common to all slots
- uncommitted lines are not returned to the A/D board

Pinout Listing for Plug-in Modules:

Note: pin 1 is at the top of the connector

amplifier slots

1	analog signal to converter
2	analog common
3	not to be used
4	not to be used
5	not to be used
6	not to be used
7	not to be used
8	not to be used
9	not to be used
10	positive power supply
11	disconnect control line
12	shorting control line
13	power supply ground
14	negative power supply
15	uncommitted
16	uncommitted
17	A2
18	A0
19	D1
20	D3
21	$\overline{D5}$
22	$\overline{D7}$

D/A converter slots

1	unconnected
2	unconnected
3	unconnected
4	unconnected
5	unconnected
6	positive power supply
7	power supply ground
8	negative power supply
9	uncommitted
10	uncommitted
11	DAC load control line
12	A2
13	A1
14	A0
15	D0
16	D1
17	D2
18	D3
19	D4
20	D5
21	$\overline{D6}$
22	$\overline{D7}$ (D7 is inverted to get the 2's complement code from the A/D to agree with the straight binary coding of the D/A converters.)

AII.19. Upgrading to 4 MHz Operation

Someday it may be desired to upgrade the system to 4 MHz operation to allow faster data handling. The following discusses what would be required to do this.

The front panel board would have to be scrapped and redesigned as it is being pushed to run at 2 MHz. This should not be an insurmountable task, as there are new chips available, and most of the board could be replaced by two chips (Intel 8279 keyboard/display controller, and Intel 2716 2048x8 EPROM).

The processor clock needs to be modified to run at 4 MHz, but this should be trivial.

The 4k RAM board needs to have faster RAM's. The 21L02's on it can only be used to 2 MHz.

The A/D board will run unmodified in a 4 MHz system. The interface boards will also work in a 4 MHz system, though some data could be lost due to transfer cycle times involving the computer the data is being passed to.

AII.20. Modifications

All of the boards in the system, other than the RAM board, were built as prototypes, and as such there are modifications to be made on them. These modifications are due in part to design changes, and in part to layout errors. The necessary modifications are listed below. Modifications to the front panel board are not discussed, because it is discouraged that anyone ever consider trying to duplicate it.

Amplifier Module - 308 Version:

The positive and negative power supplies need to be swapped. This can be accomplished by cutting the traces on the top side of the board shortly after they come onto the board, and then reconnecting them with jumpers.

Instrumentation Amplifier:

Power supply connections must be made to the AD521 instrument amplifier with jumpers (V+ to pin 8, V- to pin 5).

Isolation Amplifier:

The +8 volt supply must be jumpered to the LM319 and to resistors R6 & R7. Ground has to be connected to the 284J isolation amplifier.

D/A Converter Module - 7522 version:

The trace on the bottom of the PC board to pin 4 of the LM741 should be cut and connected to pin 3 of the LM741. Pins 8 and 21 on the AD7522 should be connected with a jumper.

D/A Mother Board:

All modifications on this board refer to IC1. On the bottom of the board, the connection between pins 6 and 8 needs to be cut, and the connection to pin 4 needs to be cut. Pins 4 and 8 need to be connected, and the trace formerly connected to pin 4 needs to be connected to pin 6.

A/D Converter Board:

There are several jumpers which must be installed on the board to meet the option selections. These are noted on the DAC interface and amplifier controls schematic and the MP-20 detail schematic.

The analog power supplies must be swapped just as they come on board. This involves cutting the traces coming on the board on S-100 pins 2 and 52 (both are available on the bottom side of the board) and adding jumpers.

D07 to the D/A converters needs to be inverted to match the codes of the A/D and D/A converters. Cut the trace to S-100 bus pin 90 shortly after it comes on board. On IC6 cut the trace connecting pins 10 and 12. Jumper S-100 pin 90 to pin 9 on IC6 and then connect pin 8 of IC6 to the trace which was connected to S-100 pin 90.

Connect pin 3 of IC11 to pin 1 of IC12. Pins 2 & 4 of IC11 should be connected to pin 8 of IC11.

Cut trace connecting pins 2 & 10 of IC16 (top of board) and connect pin 2 to pin 8.

Pin 1 of IC7 should be connected to pin 8 of IC7.

The trace between pin 14 of IC14 and pin 2 of IC12 should be cut and pin 14 of IC14 should be connected to pin 14 of IC12.

Cut traces to pin 1 of IC8 and to pin 1 of IC9. Connect both to pin 11 of IC6. Cut traces to pins 12 & 13 of IC6 (leaving them connected together) and connect them to pin 1 of IC6.

The traces to pins 2 and 12 of IC12 should be cut. Pin 2 should be connected to S-100 pin 35 and pin 12 should be connected to S-100 pin 36.

Several traces to buffers need to be rearranged. Cut the traces to pins 4, 5, 6, 7, 11, 12, 13, and 14 on IC8 and reconnect them to pins 12, 11, 14, 13, 5, 4, 7, and 6, respectively.

APPENDIX III: DETAILED SOFTWARE DESCRIPTION

AIII.1. General Comments

The software can be divided into two parts: 1. software which interfaces with the hardware, and 2. software which interfaces with the user. The division is not always clear-cut, but can be approximated by the division between that which is designed to run in real time, and that which is used to support the real time routines. Some general aspects of both will be presented, followed by a detailed description of the programs.

The hardware interface must be able to deal with a wide variety of input and output configurations and provide very high data rates while handling problems such as offset voltages. These requirements are generally conflicting and therefore must be balanced against each other to provide the best capabilities in the general case. For extremely high-performance requirements, custom programs must be written as will be discussed later.

The user interface should be as "friendly" as possible, making the instrument easy to use. The ideal would be a turn-key system, but for equipment of the type described here more flexibility is required than can be obtained with a turn-key system. The object then is to minimize the amount of information the user must provide.

AIII.2. Hardware Interface

The hardware interface routines run in real time, and therefore an attempt was made to make them run at maximal speed while still providing reasonable flexibility. The input channels are sampled in the order

specified in the channel order list, and the corresponding amplifier's offset is subtracted from the measurement. The data is encoded using two's complement code, thereby making it unnecessary to worry about overflow in the offset adjustment as it can be detected and corrected by later processing. The adjusted data is output to the corresponding D/A converter, and, along with the channel number, it is also output to the digital interface. At present, there are two digital interfaces available: a sixteen bit parallel interface, and an IEEE-488 bus interface which sends two bytes of information for each conversion.

Two hardware interface routines are provided to cover two different modes of operation. The evenly spaced sampling routine is intended for use in most data acquisition applications. Samples are taken continuously with a user selected constant time delay between samples. The bunched sampling routine makes one pass through the channel order list at the maximum sampling rate, and then provides a user selected delay before starting through the list again. This routine is intended mostly for applications, such as testing transducers, where it is desired to compare several inputs at an instant in time, and then repeat the sampling after the measurable inputs have had a chance to change.

The user should take special precautions when using the bunched sampling routine, because of the timing involved. If on-line processing is being done, one of two approaches must be used to account for the timing problems involved. The problem arises because of the system timing between the processor and the analog-to-digital converter. When the processor instructs the A/D converter to start a conversion, the converter passes the data from the previous conversion to the processor. Because of

this timing, at the end of a sample group the converter has not passed the data from the last channel. The last channel data will be passed at the beginning of the next group, but it is now out of timing sequence. A dummy channel can be added to the end of the channel order list, requiring the first data point of each group to be discarded. An alternative approach requires the data processing device to recognize the timing problem and not start processing the current set of samples until the next set is obtained.

For special requirements, such as sampling a single channel at maximal speed without the need for subtracting offsets or using the D/A converter, significant speed improvements can be realized by writing custom routines. For this type of operation, it would be easy to write a short routine which would approach or exceed the maximum converter speed of just over 80,000 samples per second.

AIII.3. User Interface

The major purpose of the user interface software is to obtain from the user those parameters needed to setup the real time routines in such a way as to meet the user's needs. It is attempted to make this part of the programs as "friendly" as possible, thereby minimizing operator problems. The ideal case would be for someone totally unfamiliar with the instrument to be able to use it with no instructions other than those printed on the front panel. This goal is not fully realized, but for the user who has read the operating instructions, it should be possible to use the system without having to refer to these operating instructions again.

The user interface software first initializes the hardware and the processor. This puts the processor in a known state and transfers the operating system to RAM for faster execution. The inputs to the instrumentation and isolation amplifiers are disconnected from the front panel and shorted together. This allows for offset voltage measurement and provides protection to the system and anything connected to it.

The second step is to obtain the operating parameters from the user. A prompt is displayed and then the software accepts a response from the user. In case the operator makes an inappropriate response (other than pressing the "reset" key), the program will ignore the response and redisplay the prompt. Each of the required inputs is described in the documentation for that section of code and more fully in the operating instructions.

After the operating parameters are entered, the user is given a chance to review the input data by pressing the "?" key. This step can be skipped or repeated as many times as desired. The program will stay in this mode until the operator presses the "run" key. Pressing the "run" key causes the user interface program to setup the processor and hardware as specified by the user, and then transfers control to the appropriate real time routine.

AIII.4. Transferring Control Between the User and Hardware Routines

The real time routines can only be entered through the Pause and Query routine of the user interface routine. This requires the user to enter all of the required data at least once. If the real time routines are exited by using the "pause" key, they can be re-entered simply by pressing the "run" key.

Control can be transferred from the real time routines to the user interface program three ways: pressing the "pause" key, pressing the "reset" key, and cycling the power. Pressing the "pause" key causes control to be transferred to the beginning of the Pause and Query routine, allowing the user to review the parameters which were entered and to re-enter the real time routines. This is useful if it is desired to either check the input parameters or modify the input hardware setup. Pressing the "reset" key or cycling the power results in the user interface routine being entered at the beginning, requiring all parameters to be entered again.

AIII.5. Overview of the Detailed Software Descriptions

The description of each of the sections of code includes a verbal description of what the routine is intended to do, a list of the register usage and subroutines called, a flow chart, and a program listing with comments.

The subroutines in the highest address EPROM (page 3 in the EPROM map given below) are general purpose subroutines which the user might find helpful in developing special system software. These subroutines have additional documentation to describe in detail how to use them. In addition to the usage instructions given, the user should insure that all subroutines called by the chosen subroutines are available.

Below and on the following pages, the user will find a key to the program listings and a listing of system addresses used by the software.

Key to the software listings:

The general instruction form is:

OPCODE OPERAND, OPERAND

The opcodes used are those assigned by Zilog for the Z-80 microprocessor.

Operands can be:

1) internal register (8 bits)
2) internal register pair (16 bits)
3) hex data (8 bits)
4) hex address (16 bits)
5) address label
6) hex port address (8 bits)

An operand enclosed by parentheses is an indirect operand. It is used to point to the memory location or input/output port where the actual operand can be found.

On relative branching instructions, a second line is included which gives the displacement as a decimal number.

EPROM map:

Page 0
00-24 General System Startup
25-BB Pause and Query
BC-BF unused
CO-DB Real Time Routines - Even Spacing
DC-DF unused
EO-F7 Real Time Routines - Bunched Spacing
F8-FF unused

Page 1
00-9D Setup for Real Time Operation
9E-FF unused

Page 2
00-16 unused
17-3F Even or Bunched Sampling
40 unused
41-9F Get Period
AO-FF Channel Order List

Page 3

```
00-01     unused
02-3B     4 BCD Digits to 2 Hex Bytes
3C-4C     Display Driver
4D-56     Clear Display
57-75     Long Timer
76-83     Short Timer
84-A7     Decoder
A8-FF     Keyboard Checker
```

prompts (located in code section)

```
RUN       10B8 - 10BB
SPCDRVR   123B - 123F
DELAY     1299 - 129D
PERIOD    129E - 129F
CHDRVR    12F5 - 12F7
CHEND     12F8 - 12FA
CHERR     12FB - 12FF
```

A/D converter channel address:

8040-8047

channel number same as last digit

RAM addresses:

```
       1000    EPROM contents
        :
        :
       13FF
STOR   1D00    input period
       1D01
       1D02
       1D03
       1D04
HBCD   1D05    period in hex for timer
       1D06
PFLAG  1D07    long or short timer indicator
SPSAVE 1D08    even or bunched sampling indicator
       1E00    channel order list with offsets
        :
        :
       1EFF
       1F00    stack
        :
        :
       1FFF
```

I/O ports:

```
00        keyboard
40-47     D/A converters - channel number same as last digit
DF        output data interface
EF        amplifier controls
F0-FF     display
```

The keyboard and display are handled by the subroutines Keyboard Checker and Display Driver, respectively. If the user writes custom software, he will have to perform the amplifier controls he desires. The amplifiers are controlled by bits 0 and 1 on port EF. Bit zero controls the disconnect feature, and bit one controls the shorting function. When these lines are in the low state (0), the amplifiers are in their normal operating mode (amplifiers unshorted and connected to the front panel connectors). Since the relays do not switch instantaneously, care should be taken in sequencing these controls to avoid damaging the signal source connected to the amplifier.

AIII.6. Comments on General System Startup (GENSYSST)

The purpose of this section of code is to initialize the processor and control the calling of the subroutines which obtain the operating parameters from the user. This section of code is entered whenever a hardware reset is executed: upon system power-up, and when the keyboard "reset" key is pressed. It requires no input parameters. The parameters passed out of it are an initialized stack pointer, and the delay parameter for the timers set for a one second delay.

register usage:

A - output control command to amplifiers
B - parameter for transferring EPROM contents to RAM
C - ditto
D - ditto
E - ditto
H - ditto
L - ditto
D' - timer delay length parameter
E' - ditto

subroutines called:

SPACING
GTPERIOD
CHORDER

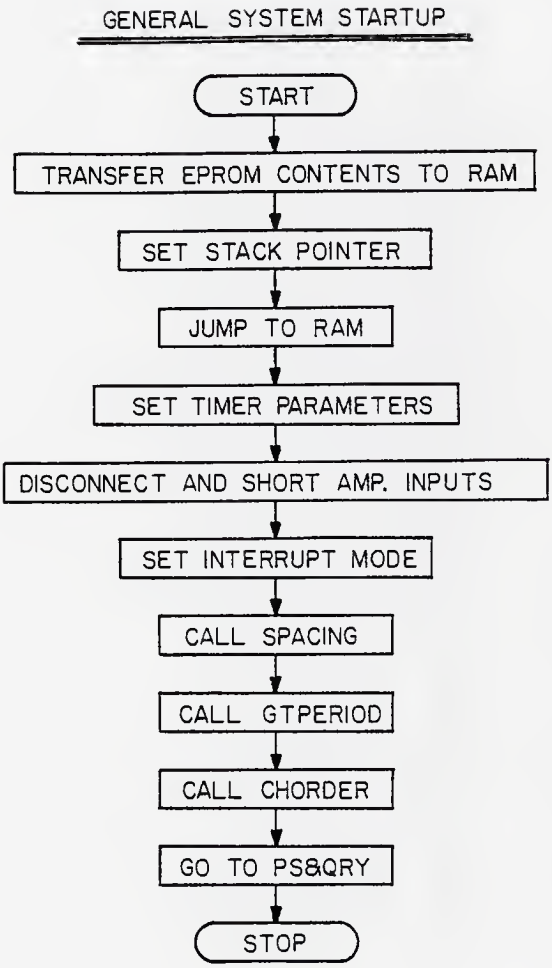


Figure AIII.1. Flowchart of General System Startup

General System Startup:

```
GENSYSST  LD HL,ROM       21   1000  move PROM contents to RAM
                          00      1
                          00      2
          LD DE,RAM       11      3
                          00      4
                          10      5
          LD BC,length    01      6
                          00      7
                          04      8
          LDIR            ED      9
                          B0      A
          LD SP,1FFE      31      B  set stack pointer
                          FE      C
                          1F      D
          JP RAM          C3      E  jump to RAM
                          11     0F
                          10     10
          EXX             D9      1  setup for 1 second delays
          LD DE,0005      11      2  in LTMR
                          05      3
                          00      4
          EXX             D9      5
          LD A,03         3E      6  disconnect and short
                          03      7  amplifier inputs
          OUT (EF),A      D3      8
                          EF      9
          IM 1            ED      A  set interrupt mode 1
                          56      B  (jump to 0038)
          CALL SPACING    CD      C  get input parameters from
                          17      D  user
                          12      E
          CALL GTPERIOD   CD     1F
                          41     20
                          12      1
          CALL CHORDER    CD      2
                          A0      3
                          12   1024
```

AIII.7. Comments on Pause and Query (PS&QRY)

This section of code allows the user to check the system operating parameters which were entered during the General System Startup. This routine is entered upon completion of the General System Startup routine or by pressing the "pause" key on the keyboard while a real time routine is running ("run" showing in the display). The ability to enter this section of code from the real time routines provides for interruption of the real time routines to modify input connections or check on the operating parameters without having to re-enter the operating parameters. Upon entry, the amplifier inputs are disconnected and shorted. This section of code can be exited by pressing the "run" key anytime "run?" appears in the display. The routine requires no input parameters other than those stored in memory by the subroutines called by General System Startup. The parameters passed out are an initialized stack pointer and the proper timer delay length parameter for use by the real time routine.

register usage:

A - general purpose I/O and testing
B - length control parameter for DISPDRVR
C - counter/display address
D - return parameter from keyboard
H - memory pointer
L - ditto
D' - timer delay length parameter
E' - ditto

subroutines called:

DISPDRVR
KEBDCHK
LTMR

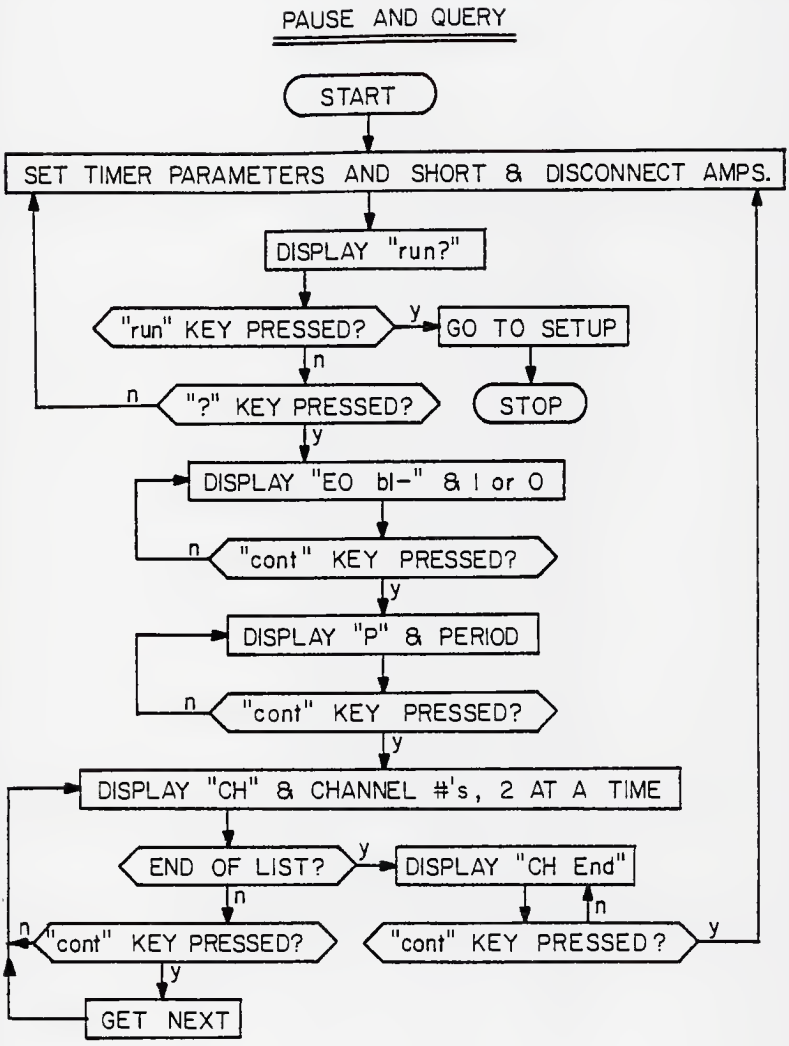


Figure AIII.2. Flowchart of Pause and Query

Pause and Query:

```
PS&QRY    EXX              D9  1025  setup for 1 second delays
          LD DE,0005       11     6  in LTMR
                           05     7
                           00     8
          EXX              D9     9
          LD A,03          3E     A  disconnect and short
                           03     B  amplifier inputs
          OUT (EF),A       D3     C
                           EF     D
RENTER    LD HL,RUN        21     E  output "run?" to display
                           B8    2F
                           10    30
          LD B,04          06     1
                           04     2
          CALL DISPDRVR    CD     3
                           3C     4
                           13     5
          JR +05           18     6  jump over interrupt handler
                           03     7
          JP PS&QRY        C3     8  on interrupt (exit real time)
                           25     9  jump to PS&QRY routine
                           10     A  in RAM
          CALL KEBDCHK     CD     B  get user response from
                           A8     C  keyboard
                           13     D
          LD A,D           7A     E  check to see if input was
          CP "run"         FE    3F  "run"
            (23)           23    40
          JPZ SETUP        CA     1  if it was "run" jump to
                           00     2  setup for real time
                           11     3  operation
          CP "?"           FE     4  if not "run" check for "?"
            (24)           24     5
          JRNZ RENTER      20     6  if not "run" or "?" try
            (-24)          E6     7  prompt again
SPOUT     LD HL,SPCDRVR    21     8  output "E0 b1" to display
                           3B     9
                           12     A
          LD B,04          06     B
                           04     C
          CALL DISPDRVR    CD     D
                           3C     E
                           13    4F
          LD A,(SPSAVE)    3A    50  get user response to even
                           08     1  or bunched sampling question
                           1D     2
          OUT (FE),A       D3     3  output user response to
                           FE     4  display
          CALL KEBDCHK     CD     5  check keyboard for input
                           A8     6
                           13  1057
```

Pause and Query:

```
          LD A,"cont"      3E  1058  check to see if input was
            (26)           26     9  "cont"
          CP D             BA     A
          JRNZ SPOUT       20     B  if not "cont" go back and
            (-19)          EB     C  output user response again
POUT      LD HL, PERIOD    21     D  output "P" to display
                           9E     E
                           12    5F
          LD B,01          06    60
                           01     1
          CALL DISPDRVR    CD     2
                           3C     3
                           13     4
          LD HL,STOR+4     21     5  load location of user input
                           04     6  period in HL
                           1D     7
          LD C,F8          0E     8  set counter
                           F8     9
OUTP      DEC HL           2B     A  output a digit of the period
          LD A,(HL)        7E     B
          OUT (C),A        ED     C
                           79     D
          INC C            0C     E  update counter
          INC C            0C    6F
          JRNZ OUTP        20    70  if not done go back and
            (-6)           F8     1  do next digit
          CALL LTMR        CD     2  waste time for key debounce
                           57     3
                           13     4
          CALL KEBDCHK     CD     5  check keyboard for input
                           A8     6
                           13     7
          LD A,"cont"      3E     8  check to see if input was
            (26)           26     9  "cont"
          CP D             BA     A
          JRNZ POUT        20     B  if not "cont" go back and
            (-30)          E0     C  output period again
          LD HL,LIST       21     D  load HL with beginning of
                           04     E  channel order list
                           1E    7F
CHOUT     PUSH HL          E5    80  save HL
          LD HL, CHDRVR    21     1  output "CH" to display
                           F5     2
                           12     3
          LD B,02          06     4
                           02     5
          CALL DISPDRVR    CD     6
                           3C     7
                           13  1088
```

Pause and Query:

```
          POP HL           E1  1089  recover HL
          LD A,(HL)        7E     A  input channel number
          CP FF            FE     B  check for end of list
                           FF     C
          JRZ ESCAPE       28     D  if end, go to ESCAPE
            (+21)          13     E
          OUT (FE),A       D3     F  output channel number
                           FE    90  to display
          PUSH HL          E5     1  save HL
          CALL KEBDCHK     CD     2  check keyboard for input
                           A8     3
                           13     4
          POP HL           E1     5  recover HL
          LD A,"cont"      3E     6  check to see if input was
            (26)           26     7  "cont"
          CP D             BA     8
          JRNZ CHOUT       20     9  if not "cont" output most
            (-25)          E5     A  recent channel number again
          INC HL           23     B  increment to next address
          INC HL           23     C  containing a channel number
          CALL LTMR        CD     D  waste time for key debounce
                           57     E
                           13    9F
          JR CHOUT         18    A0  go get next channel number
            (-32)          DE     1
ESCAPE    LD HL,CHEND      21     2  output "CH End" to display
                           F8     3
                           12     4
          LD B,03          06     5
                           03     6
          CALL DISPDRVR    CD     7
                           3C     8
                           13     9
          CALL LTMR        CD     A  waste time for key debounce
                           57     B
                           13     C
          CALL KEBDCHK     CD     D  check keyboard for input
                           A8     E
                           13    AF
          LD A,"cont"      3E    B0  check to see if input was
                           26     1  "cont"
          CP D             BA     2
          JRNZ ESCAPE      20     3  if not "cont" go back and
            (-17)          ED     4  output "CH End" again
          JP RENTER        C3     5  go back to beginning
                           2E     6
                           10     7
RUN         "r"            57     8  prompt storage location
            "u"            78     9
            "n"            94     A
            "?"            B9  10BB
```

AIII.8. Comments on Real Time Routines: Even Spacing (RTE) and Bunched Spacing (RTB)

There are two real time routines which differ in their sampling strategy, but both have the same purpose. The real time routines control the input channel selection, interface with the conversion hardware (both A/D and D/A), compensate for amplifier offset, and interface with the data output hardware. The Evenly Spaced Sampling routine samples the input channels with a constant time delay between each sample. The Bunched Spacing Sampling routine samples all channels in the channel order list at the maximum possible rate and then delays before starting over. The input parameters to these routines are: the channel order list, the offset of each channel, and the delay parameter for the timing routine. The only output from the routine is the data from the converted samples.

register usage:

A - I/O and miscellaneous
B - counter for channel order list length
C - last channel address and D/A port
D - A/D converter module address
E - ditto
H - channel order list pointer
L - ditto

subroutines called:

LTMR or STMR

The real time routines were written to maximize speed when running and have lost a modular nature lending itself to flow charting. For this reason, flow charts are not given for these routines. The interested user should direct his attention to the program listings and try to understand the operation from them. As an aid, it is suggested that the user remember that all parallel loops are constrained to being of equal length.

If the user develops special real time routines to meet special needs, he should be careful to insure that all parallel loops require the same number of clock cycles for execution. The number of clock cycles per instruction are tabulated in Zilog's documentation for the Z-80 microprocessor. Parallel loops generally occur when branching is done in the program.

Real Time Routines - Even Spacing:

```
RTE       LD D,MP20U        16  10C0  upper byte of MP20 address
                            80    1
BEGIN     LD HL,LIST-1      21    2   top of channel order list
                            03    3
                            1E    4
          LD B,(HL)         46    5   length of list
MORE      INC HL            23    6
          LD E,(HL)         5E    7   channel number to sample
          CALL timer        CD    8   waste time
            (or 3 NOPs)     XX    9   (timer address or NOPs
                            XX    A   loaded by SETUP)
          LD A,(DE)         1A    B   get data from converter
          INC HL            23    C
          SUB (HL)          96    D   subtract offset
          OUT (DF),A        D3    E   output data to digital
                            DF   CF   interface board
          OUT (C),A         ED   D0   output data to D/A
                            79    1
          LD C,E            4B    2   save channel number
          DEC B             05    3   test for end of channel
          JRZ BEGIN         28    4   order list
            (-18)           EC    5
          LD IY,MORE        FD    6   waste time to even up
                            21    7   parallel loops, and set
                            C6    8   jump address
                            10    9
          JP (IY)           FD    A
                            E9  10DB
```

Real Time Routines - Bunched Spacing:

```
RTB       LD D,MP20U        16  10E0  upper byte of MP20 address
                            80    1
BEGIN2    LD HL,LIST-1      21    2   top of channel order list
                            03    3
                            1E    4
          LD B,(HL)         46    5   length of list
          CALL timer        CD    6   waste time
           (or 3 NOPs)      XX    7   (timer address or NOPs
                            XX    8   loaded by SETUP)
MORE2     INC HL            23    9
          LD E,(HL)         5E    A   channel number to sample
          LD A,(DE)         1A    B   get data from converter
          INC HL            23    C
          SUB (HL)          96    D   subtract offset
          OUT (DF),A        D3    E   output data to digital
                            DF   EF   interface board
          OUT (C),A         ED   F0   output data to D/A
                            79    1
          LD C,E            4B    2   save channel number
          DEC B             05    3   test for end of channel
          JRNZ MORE2        20    4   order list
            (-11)           F3    5
          JR BEGIN2         18    6
            (-20)           EA  10F7
```

AIII.9. Comments on Setup for Real Time Operation (SETUP)

The purpose of this section of code is to take the system operating parameters obtained from the user during the General System Startup and put them in the proper format for use by the real time operating routines. It also obtains the offset for each of the input amplifiers. The input and output parameters are all of the system operating parameters which were stored in memory by the General System Startup, and the amplifier offset voltages.

The organization of the channel order list is complicated by the interaction between the conversion hardware and the software. When the converter is directed to start a conversion, it gives the processor the results of the prior conversion. This requires the offset for the prior channel to be placed after the current channel address in the list. As an example: if one wishes to sample three channels--1,2,3--in that order, the channel order list would be:

```
address of 1
offset of  3
address of 2
offset of  1
address of 3
offset of  2
```

register usage:

A - general purpose
B - A/D converter module address
C - ditto
D - counter
E - constant used as mask to set address
H - channel order list pointer
L - ditto
A' - general purpose
D' - timer length parameter
E' - ditto

subroutines called:

LTMR
STMR

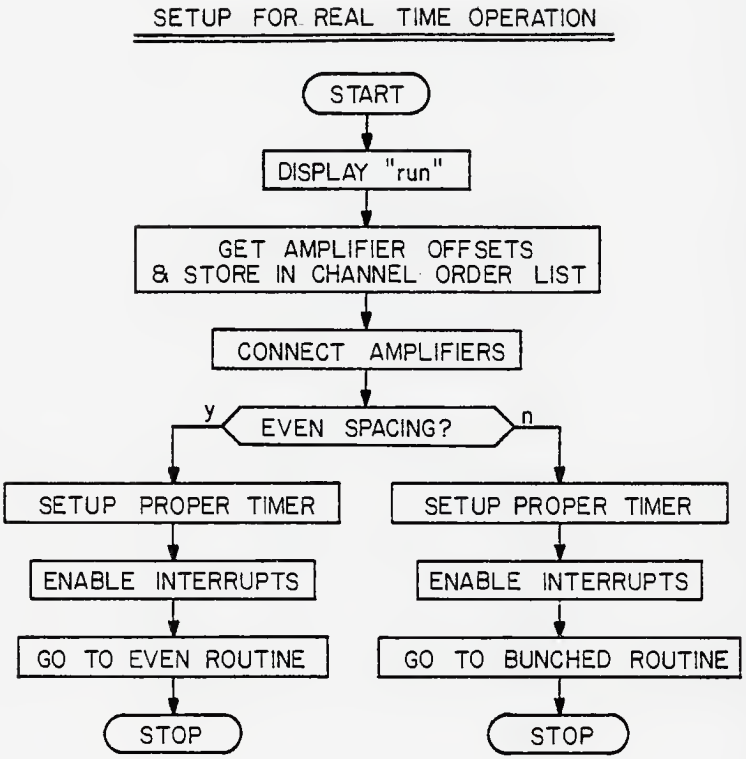


Figure AIII.3. Flowchart of Setup for Real Time Operation

Setup for Real Time Operation:

```
SETUP     LD A,0F          3E  1100  clear "?" from display
                           0F     1  leaving "run"
          OUT (FB),A       D3     2
                           FB     3
          LD B,MP20U       06     4  upper byte of MP20 address
                           80     5
          LD E,40          1E     6  constant used as mask
                           40     7
          LD HL,LIST       21     8  location of channel order
                           04     9  list
                           1E     A
          LD D,FF          16     B  initialize list length
                           FF     C  counter
GETLIST   INC D            14     D  update counter
          LD A,(HL)        7E     E  channel counter
          CP FF            FE    0F  check for end of list
                           FF    10  indicator
          JRZ DONE         28     1  if end jump to DONE
            (+14)          0C     2
          OR E             B3     3  set left nibble to F
          LD (HL),A        77     4  return to channel list
          LD C,A           4F     5  address lower byte
          LD A,(BC)        0A     6  offset of prior channel
          INC HL           23     7
          LD (HL),A        77     8  store offset
          INC HL           23     9
          CALL STMR        CD     A  waste time (126 usec)
                           76     B  to allow for next
                           13     C  conversion
          JR GETLIST       18     D  repeat for next channel
            (-16)          EE     E
DONE      LD A,(BC)        0A    1F  get last offset
          LD (LIST+1),A    32    20  store last offset at
                           05     1  beginning of list
                           1E     2
          LD A,D           7A     3  list length
          LD (LIST-1),A    32     4  store length at bottom of
                           03     5  channel order list
                           1E     6
          LD SP,1FFE       31     7  reinitialize stack pointer
                           FE     8  (this guarantees there is
                           1F     9  no creeping stack)
          LD A,00          3E     A  connect amplifiers
                           00     B
          OUT (EF),A       D3     C
                           EF     D
          CALL LTMR        CD     E  waste time to allow relays
                           57    2F  to stop bouncing and amps
                           13    30  to settle (1 sec)
          LD A,(SPSAVE)    3A     1  user input as to whether
                           08     2  to use bunched or even
                           1D     3  samples
          CP 00            FE     4  check for even sampling
                           00  1135
```

Setup for Real Time Operation:

```
          JRNZ GORTB        20  1136  if not even sampling jump
            (+53)           33     7  to setup for bunched
          LD A,(PFLAG)      3A     8  get user response to long
                            07     9  or short timer question
                            1D     A
          CP 00             FE     B  check for short timer
                            00     C
          JRNZ LONG         20     D  if not short timer jump
            (+6)            04     E  to setup for long
          LD A,STMRA        3E    3F  lower byte of short timer
                            76    40  address
          JR OUTMR          18     1
            (+4)            02     2
LONG      LD A,LTMRA        3E     3  lower byte of long timer
                            57     4  address
OUTMR     LD (10C9),A       32     5  load location in real time
                            C9     6  with proper timer address
                            10     7  lower byte
          LD A,CD           3E     8  load CALL instruction into
                            CD     9  proper location in real
          LD (10C8),A       32     A  time routine
                            C8     B
                            10     C
          LD A,13           3E     D  upper byte of timer
                            13     E  address
          LD (10CA),A       32    4F  load location in real time
                            CA    50  with timer address upper
                            10     1  byte
          EXX               D9     2
          LD DE,(HBCD)      ED     3  get timer delay length
                            5B     4  parameter to be passed to
                            05     5  the timer routine during
                            1D     6  real time operation
          LD A,D            7A     7  check for zero period
          OR E              B3     8
          JRNZ LVTMR        20     9  if not zero jump
            (+13)           0B     A
          LD A,00           3E     B  if the period is zero
                            00     C  replace the CALL to timer
          LD (10C8),A       32     D  with 3 NOPs to minimize
                            C8     E  execution time
                            10    5F
          LD (10C9),A       32    60
                            C9     1
                            10     2
          LD (10CA),A       32     3
                            CA     4
                            10     5
LVTMR     EXX               D9     6
          EI                FB  1167  enable maskable interrupt
```

Setup for Real Time Operation:

```
          JP RTE          C3  1168  jump to even spaced sampling
                          C0     9  routine
                          10     A
CORTB     LD A,(PFLAC)    3A     B  This section of code is
                          07     C  functionally identical
                          1D     D  to that given above for
          CP 00           FE     E  the even spaced sampling
                          00    6F  setup.  The only
          JRNZ CLONC      20    70  difference is the
            (+6)          04     1  addresses.
          LD A,STMRA      3E     2
                          76     3
          JP OPTMRA       18     4
            (+4)          02     5
CLONC     LD A,LTMRA      3E     6
                          57     7
OPTMRA    LD (10E7),A     32     8
                          E7     9
                          10     A
          LD A,CD         3E     B
                          CD     C
          LD (10E6),A     32     D
                          E6     E
                          10    7F
          LD A,13         3E    80
                          13     1
          LD (10E8),A     32     2
                          E8     3
                          10     4
          EXX             D9     5
          LD DE,(HBCD)    ED     6
                          5B     7
                          05     8
                          1D     9
          LD A,D          7A     A
          OR E            B3     B
          JRNZ TMROK      20     C
            (+13)         0B     D
          LD A,00         3E     E
                          00    8F
          LD (10E6),A     32    90
                          E6     1
                          10     2
          LD (10E7),A     32     3
                          E7     4
                          10     5
          LD (10E8),A     32     6
                          E8     7
                          10     8
TMROK     EXX             D9     9
          EI              FB     A
          JP RTB          C3     B  jump to bunched sampling
                          E0     C  real time routine
                          10  119D
```

AIII.10. Comments on Even or Bunched Sampling (SPACING)

The purpose of this section of code is to obtain an indication from the operator as to whether he wants to have all of the samples taken at evenly spaced intervals, or to have the samples taken as a group with a user selectable delay between successive groups. The user's input is stored for use by the setup routine. There are no input parameters, and the output parameter is the user's input.

register usage:

A - general purpose
B - length control parameter for DISPDRVR
D - return parameter from keyboard
H - memory pointer
L - memory pointer and digit return from keyboard

subroutines called:

DISPDRVR
KEBDCHK
LTMR

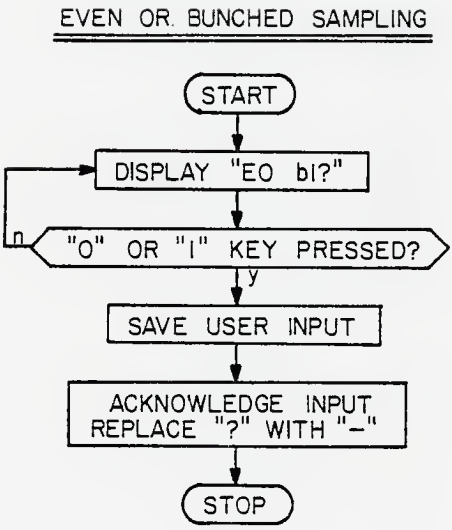


Figure AIII.4. Flowchart of Even or Bunched Sampling

Even or Bunched Sampling:

```
SPACING   LD HL,SPCDRVR     21  1217  output "E0 b1?" to display
                            3B    8
                            12    9
          LD B,05           06    A
                            05    B
          CALL DISPDRVR     CD    C
                            3C    D
                            13    E
          CALL KEBDCHK      CD   1F  check keyboard for input
                            A8   20
                            13    1
          LD A,"enter"      3E    2  check to see if input was
            (22)            22    3  digit followed by "enter"
          CP D              BA    4
          JRNZ SPACING      20    5  if not start over
            (-14)           F0    6
          LD A,L            7D    7  check to see if input was
          CP 00             FE    8 "0"
                            00    9
          JRZ ISZERO        28    A  jump if zero
            (+6)            04    B
          CP 01             FE    C  check to see if input was
                            01    D  "1"
          JRNZ SPACING      20    E  if not "0" or "1" start
            (-23)           E7   2F  over
ISZERO    LD SPSAVE,A       32   30  save user input
                            08    1
                            1D    2
          LD A,0A           3E    3  acknowledge user input by
                            0A    4  replacing "?" with "-"
          OUT (FD),A        D3    5
                            FD    6
          CALL LTMR         CD    7  delay for one second
                            57    8
                            13    9
          RET               C9    A  return to caller
SPCDRVR   "E"               30    B  prompt storage location
          "0"               40    C
          "b"               8B    D
          "1"               A1    E
          "?"               D9  123F
```

AIII.11. Comments on Get Period (GTPERIOD).

The purpose of this section of code is to obtain from the user the time interval he wishes to have between samples or groups of samples. First, the operator is asked if he wishes to use the long or short timing routines, and then he is asked to enter the period of the delay. The period is a four digit decimal number (0000 - 9999). In the short timing routine, the period will be a delay of 25 microseconds times the number entered plus one microsecond. In the long timing routine, the delay will be 0.2 seconds times the number entered. The choice of timing routines and the length of the delay are stored for use by the setup routine. There are no input parameters, and the output parameters are the user's inputs.

register usage:

A - general purpose
B - length control parameter for DISPDRVR
C - counter
D - return parameter from keyboard
H - memory pointer
L - memory pointer and digit return from keyboard
IX - storage location pointer

subroutines called:

DISPDRVR
KEBDCHK
LTMR
DTOH

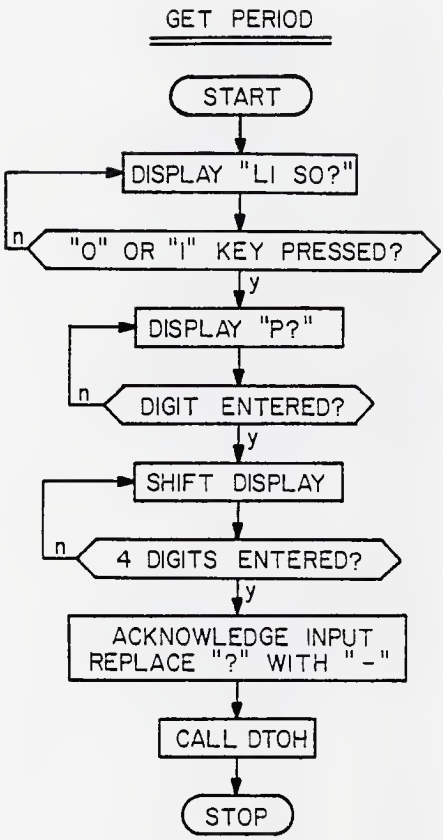


Figure AIII.5. Flowchart of Get Period

Get Period:

```
GTPERIOD  LD HL,DELAY      21  1241  output "L1 SO?" to display
                           99     2
                           12     3
          LD B,05          06     4
                           05     5
          CALL DISPDRVR    CD     6
                           3C     7
                           13     8
          CALL KEBDCHK     CD     9  check keyboard for input
                           A8     A
                           13     B
          LD A,"enter"     3E     C  check to see if input was
            (22)           22     D  digit followed by "enter"
          CP D             BA     E
          JRNZ GTPERIOD    20    4F  if not start over
            (-14)          F0    50
          LD A,L           7D     1  check to see if input was
          CP 00            FE     2  "0"
                           00     3
          JRZ ISZERO       28     4  jump if zero
            (+6)           04     5
          CP 01            FE     6  check to see if input was
                           01     7  "1"
          JRNZ GTPERIOD    20     8  if not "0" or "1" start
            (-23)          E7     9  over
ISZERO    LD (PFLAG),A     32     A  save user input
                           07     B
                           1D     C
          LD A,0A          3E     D  acknowledge user input by
                           0A     E  replacing "?" with "-"
          OUT (FD),A       D3    5F
                           FD    60
          CALL LTMR        CD     1  delay for one second
                           57     2
                           13     3
          LD HL,PERIOD     21     4  output "P?" to display
                           9E     5
                           12     6
          LD B,02          06     7
                           02     8
          CALL DISPDRVR    CD     9
                           3C     A
                           13     B
          LD C,F6          0E     C  load counter/display address
                           F6     D
          LD IX,STOR+3     DD     E  establish storage location
                           21    6F
                           03    70
                           1D  1271
```

Get Period:

```
GOGET     PUSH BC          C5  1272  save BC
          CALL KEBDCHK     CD     3  check keyboard for input
                           A8     4
                           13     5
          POP BC           C1     6  recover BC
          LD A,"enter"     3E     7  check to see if input was
            (22)           22     8  digit followed by "enter"
          CP D             BA     9
          JRNZ GOGET       20     A  if not jump back
            (-8)           F6     B
          LD (IX+0),L      DD     C  save input digit
                           75     D
                           00     E
          DEC IX           DD    7F  update pointer
                           2B    80
          OUT (C),L        ED     1  acknowledge user input by
                           69     2  shifting digit in display
          LD A,0F          3E     3  and clearing rightmost
                           0F     4  digit
          OUT (FE),A       D3     5
                           FE     6
          INC C            0C     7  update counter/display
          INC C            0C     8  address
          LD A,FE          3E     9  check to see if four
                           FE     A  digits have been entered
          CP C             B9     B
          JRNZ GOGET       20     C  if not four, then go back
            (-26)          E4     D  for more
          LD A,0A          3E     E  acknowledge entry of four
                           0A    8F  digits by replacing "?"
          OUT (F5),A       D3    90  with "-"
                           F5     1
          CALL DTOH        CD     2  convert four BCD digits
                           02     3  to two packed hex bytes
                           13     4
          CALL LTMR        CD     5  delay for one second
                           57     6
                           13     7
          RET              C9     8  return to caller
DELAY     "L"              3B     9  prompt storage location
          "1"              41     A
          "S"              85     B
          "0"              A0     C
          "?"              D9     D
PERIOD    "P"              36     E
          "?"              59  129F
```

AIII.12. Comments on Channel Order List (CHORDER)

The purpose of this section of code is to obtain from the user a list of the order in which he wants to have the channels sampled. The input list is stored for processing by the setup routine before it is used by the real time routines. The maximum length of the list is 64 channels. There are no input parameters, and the output parameters are the user inputs.

register usage:

A - general purpose
B - length control parameter for DISPDRVR
C - counter
D - return parameter from keyboard
H - memory pointer
L - memory pointer and digit return from keyboard
IX - list pointer

subroutines called:

DISPDRVR
KEBDCHK
LTMR

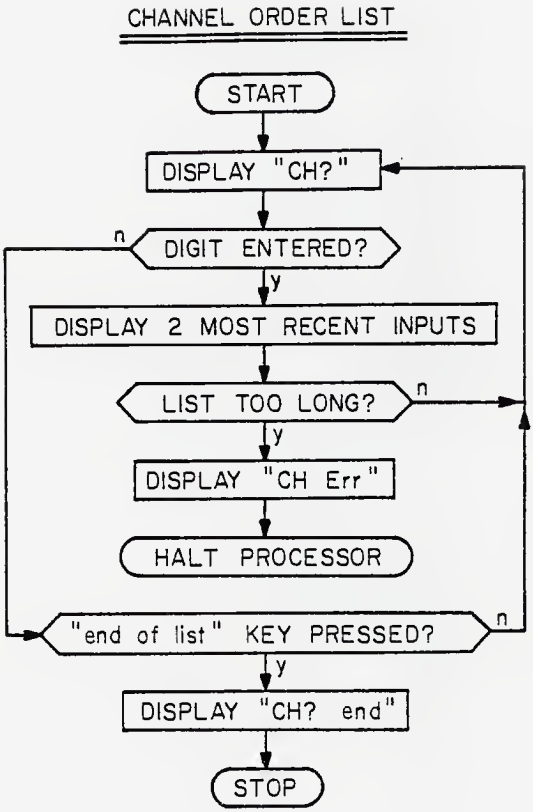


Figure AIII.6. Flowchart of Channel Order List

Channel Order List:

```
CHORDER   LD HL,CHDRVR      21  12A0  output "CH?" to display
                            F5    1
                            12    2
          LD B,03           06    3
                            03    4
          CALL DISPDRVR     CD    5
                            3C    6
                            13    7
          LD C,64           0E    8  initialize counter to
                            64    9  maximum list length
          LD IX,LIST        DD    A  establish location of
                            21    B  channel order list
                            04    C
                            1E    D
          LD (IX-2),0F      DD    E  assure that display will
                            36   AF  blank properly during
                            FE   B0  first display shift
                            0F    1
RESTART   LD A,0F           3E    2  clear rightmost digit
                            0F    3  of display
          OUT (FE),A        D3    4
                            FE    5
          PUSH BC           C5    6  save BC
          CALL KEBDCHK      CD    7  check keyboard for input
                            A8    8
                            13    9
          POP BC            C1    A  recover BC
          LD A,D            7A    B  check to see if input was
          CP "enter"        FE    C  digit followed by "enter"
            (22)            22    D
          JRZ DIGIT         28    E  if it is then jump
            (+28)           1A   BF
          CP "end of        FE   C0  check to see if input was
            list" (25)      25    1  "end of list"
          JRNZ RESTART      20    2  if not digit or "end of
            (-16)           EE    3  list" try again
          LD HL,CHDRVR      21    4  output "CH? End" to
                            F5    5  display
                            12    6
          LD B,06           06    7
                            06    8
          CALL DISPDRVR     CD    9
                            3C    A
                            13    B
          LD (IX+0),FF      DD    C  load end of list indicator
                            36    D  at end of channel order
                            00    E  list
                            FF  12CF
```

Channel Order List:

```
         LD A,(LIST)       3A  12D0  check to be sure list
                           04    1   isn't of zero length
                           1E    2
         INC A             3C    3
         JRZ CHORDER       28    4   if zero length start over
           (-52)           CA    5
         CALL LTMR         CD    6   delay for one second
                           57    7
                           13    8
         RET               C9    9   return to caller
DIGIT    LD A,L            7D    A   load most recent digit
         OUT (FC),A        D3    B   in next to rightmost
                           FC    C   display digit
         LD (IX+0),A       DD    D   save most recent digit
                           77    E
                           00   DF
         INC IX            DD   E0   update pointer
                           23    1
         INC IX            DD    2
                           23    3
         LD A,(IX-4)       DD    4   load next to most recent
                           7E    5   digit in the digit just
                           FC    6   left of the one containing
         OUT (FA),A        D3    7   most recent
                           FA    8
         DEC C             0D    9   update channel order list
         JRNZ RESTART      20    A   length counter & check for
           (-56)           C6    B   list being too long
         LD HL,CHERR       21    C   if too long output
                           FB    D   "CH Err" to display and
                           12    E   stop program execution
         LD B,05           06   EF
                           05   F0
         CALL DISPDRVR     CD    1
                           3C    2
                           13    3
         HALT              76    4
CHDRVR   "C"               2C    5   prompt storage location
         "H"               52    6
         "?"               79    7
CHEND    "E"               B0    8
         "n"               D4    9
         "d"               ED    A
CHERR    "C"               2C    B
         "H"               52    C
         "E"               90    D
         "r"               B7    E
         "r"               D7  12FF
```

AIII.13. Comments on 4 BCD Digits to 2 Hex Bytes (DTOH)

The purpose of this subroutine is to take the four BCD digits entered as the delay length between samples and convert them to four hex digits packed into two bytes. This allows the timing routines to use simple binary math. The conversion technique is performed by decrementing each BCD digit to zero and adding an appropriately weighted binary constant at each decrement to the hex equivalent. The input parameters are the four BCD digits stored in memory locations 1D00H to 1D03H. The output parameters are the two hex bytes placed in memory locations 1D05H and 1D06H.

register usage:

A - general purpose
B - decimal digit being converted
C - counter
D - generated hex output
E - ditto
H - storage location pointer
L - ditto
IX - storage location pointer

subroutines called:

no subroutines called

using DTOH:

inputs: 4 BCD digits stored in the lower nibble of memory locations 1D00 to 1D03; the least significant digit is stored in 1D00; storage location 1D04 is used by the routine

outputs: the equivalent 16 bit hexadecimal number is returned in memory locations 1D05 and 1D06

registers affected: A, B, C, D, E, H, L, IX

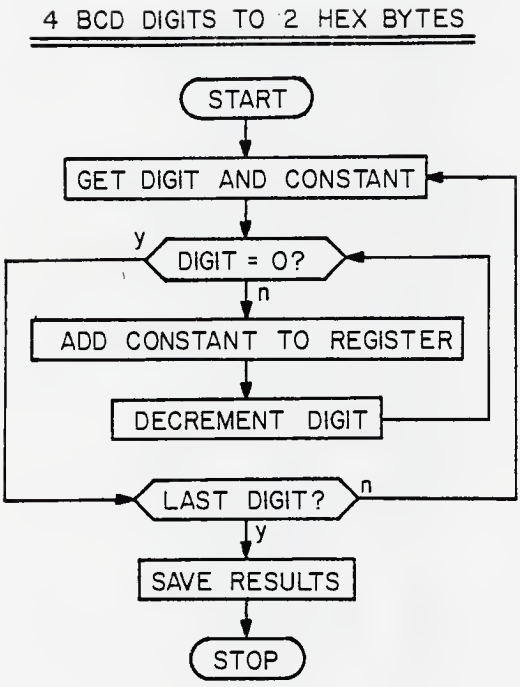


Figure AIII.7. Flowchart of 4 BCD Digits to 2 Hex Bytes

4 BCD Digits to 2 Hex Bytes:

```
DTOH    LD A,(STOR+3)     3A  1302  copy most significant
                          03    3   digit into next memory
                          1D    4   location
        LD (STOR+4),A     32    5
                          04    6
                          1D    7
        LD D,00           16    8
                          00    9
        LD IX,STOR        DD    A   set pointer to memory
                          21    B   locations containing
                          00    C   BCD digits
                          1D    D
        LD HL,LOC         21    E   set pointer to constants
                          38   0F   used for conversion
                          13   10
        LD E,(IX+0)       DD    1   get least significant
                          5E    2   digit
                          00    3
        LD C,04           0E    4   set counter
                          04    5
NEXT    INC IX            DD    6   increment to next more
                          23    7   significant digit
        XOR A             AF    8   set A to zero
        LD B,(IX+0)       DD    9   get digit
                          46    A
                          00    B
        INC B             04    C   test digit for zero
AGAN    DEC B             05    D
        JRZ EXLP          28    E   jump if zero
          (+8)            06   1F
        ADD A,(HL)        86   20   add constant to A
        JRNC AGAN         30    1   if no carry do it again
          (-4)            FA    2
        INC D             14    3   if carry increment D
        JR AGAN           18    4   and go decrement again
          (-7)            F7    5
EXLP    DEC C             0D    6   update counter
        JRZ DONE          28    7   jump if all converted
          (+10)           08    8
        INC HL            23    9   next constant
        ADD A,E           83    A   update hex
        LD E,A            5F    B
        JRNC,NEXT         30    C
          (-22)           E8    D
        INC D             14    E
        JR NEXT           18   2F
          (-25)           E5 1330
```

4 BCD Digits to 2 Hex Bytes:

```
DONE    ADD A,D          82  1331  update hex
        LD D,A           57     2
        LD (HBCD),DE     ED     3  store results
                         53     4
                         05     5
                         1D     6
        RET              C9     7  return to caller
LOC                      0A     8  conversion constants
                         64     9
                         E8     A
                         03  133B
```

```
STOR    LSD                 1D00  BCD storage
                               1
                               2
        MSD                    3
                               4  MSD copy
HBCD                           5  result
                            1D06
```

AIII.14. Comments on Display Driver (DISPDRVR)

This subroutine takes a message stored in memory and places it in the display. The message in memory should have the address of the proper display digit in the upper nibble and the character designator in the lower nibble. (The display digit addresses and character designators are tabulated in the hardware documentation for the front panel board.) The input parameters are the location and length of the message. There is no output parameter.

register usage:

A - input/output
B - message length parameter
C - display port address
H - message location pointer
L - ditto

subroutines called:

CLRDISP

using DISPDRVR:

inputs: the message to be displayed stored in RAM; the format is the display digit address in the upper nibble and the character designator in the lower nibble; the register pair HL is the address of the first byte of the output list; the register B contains the number of characters to be displayed

outputs: the displayed message

registers affected: A, B, C, H, L

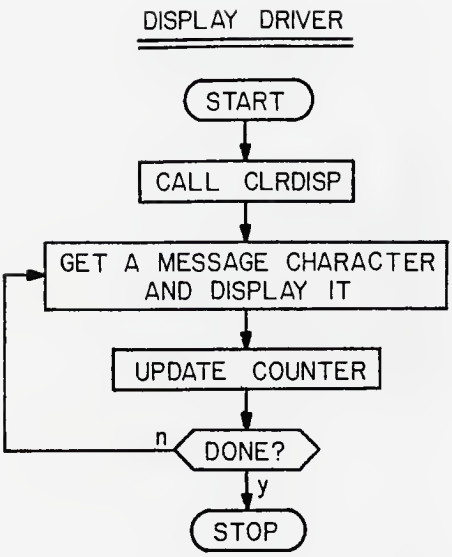


Figure AIII.8. Flowchart of Display Driver

```
DISPDRVR  CALL CLRDISP     CD  133C  clear display before
                           4D    D   writing to it
                           13    E   also set A=0F
          RRD              ED   3F   get character in A
                           67   40
          LD C,(HL)        4E    1   get port address
          OUT (C),A        ED    2   load display
                           79    3
          RRD              ED    4   restore message
                           67    5
          RRD              ED    6
                           67    7
          INC HL           23    8   next location
          DEC B            05    9   update length counter
          JRNZ -11         20    A   if not done do next
                           F3    B
          RET              C9  134C  return to caller
```

AIII.15. Comments on Clear Display (CLRDISP)

This subroutine clears the display. It has no input or output parameters.

register usage:

A - blank character
C - counter/display port address

subroutines called:

no subroutines called

using CLRDISP:

inputs: none

outputs: blank display

registers affected: A, C

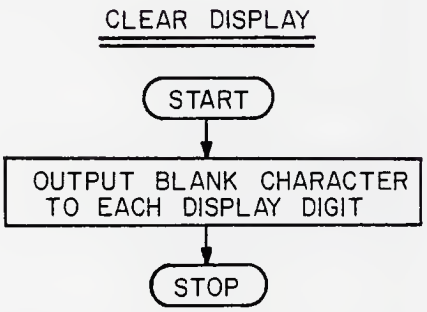


Figure AIII.9. Flowchart of Clear Display

```
CLRDISP   LD A,0F          3E  134D  blanking character
                           0F    E
          LD C,F0          0E    4F  leftmost display digit
                           F0    50
          OUT (C),A        ED     1  output blank character
                           79     2
          INC C            0C     3  next display digit
          JRNZ -3          20     4  if not done do more
                           FB     5
          RET              C9  1356  return to caller
```

AIII.16. Comments on Long Timer (LTMR) and Short Timer (STMR)

The purpose of the timers is to provide a precision delay when it is needed by other sections of the program. The short timer provides a delay equal to 25 useconds times the input delay parameter plus a constant error of 1 usecond. The long timer provides a delay equal to 0.2 seconds times the input delay parameter. The input parameter is the desired length of delay. There is no output parameter.

register usage:

A' - general purpose
B' - counter
C' - ditto
D' - delay length parameter
E' - ditto

subroutines called:

LTMR calls STMR

using LTMR or STMR:

inputs: delay parameter "P" in register pair D'E'; "P" can be any nonzero 16 bit hex number

outputs: STMR delay = Px25+1 microsecond

LTMR delay = Px0.2 second

registers affected: A', B', C'

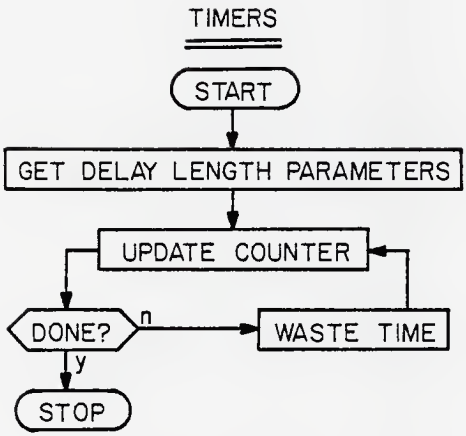


Figure AIII.10. Flowchart of Long Timer and Short Timer

Timers:

```
LTMR    EXX             D9  1357  exchange register sets
        LD B,D          42     8  get length of delay
        LD C,E          4B     9  parameters
        PUSH BC         C5     A  save registers
        PUSH DE         D5     B
        LD DE,1F3D      11     C  setup to call short
                        3D     D  timer
                        1F     E
        EXX             D9    5F
        CALL STMR       CD    60
                        76     1
                        13     2
        EXX             D9     3
        LD A,(IX+0)     DD     4  waste time
                        7E     5
                        00     6
        POP DE          D1     7  recover registers
        POP BC          C1     8
        DEC BC          0B     9  update counter
        LD A,B          78     A  check to see if done
        OR C            B1     B
        JRNZ +4         20     C  jump if not done
                        02     D
        EXX             D9     E  recover main register set
        RET             C9    6F  return to caller
        LD A,00         3E    70  waste time
                        00     1
        LD A,00         3E     2
                        00     3
        JR -26          18     4  jump back
                        E4  1375
STMR    EXX             D9  1376  exchange register sets
        LD B,D          42     7  get length of delay
        LD C,E          4B     8  parameters
        DEC BC          0B     9  update counter
        LD A,B          78     A  check to see if done
        OR C            B1     B
        JRMZ +04        20     C  jump if not done
                        02     D
        EXX             D9     E  recover main register set
        RET             C9    7F  return to caller
        JR +02          18    80  waste time
                        00     1
        JR -09          18     2  jump back
                        F5  1383
```

AIII.17. Comments on Decoder (DECODER)

This subroutine helps insure that only one key was pressed at the time the keyboard was checked, and returns a unique value for each key by indicating its row and column number. The input parameter is the keyboard input of a one-of-eight column input. The output parameter is a concatenation of the row and column numbers of the key.

register usage:

A - general purpose
B - mask/transfer from KEBDCHK/input register
C - keyboard port address
D - row and column number of key

subroutines called:

no subroutines called

using DECODER:

this subroutine should only be accessed from the subroutine KEBDCHK

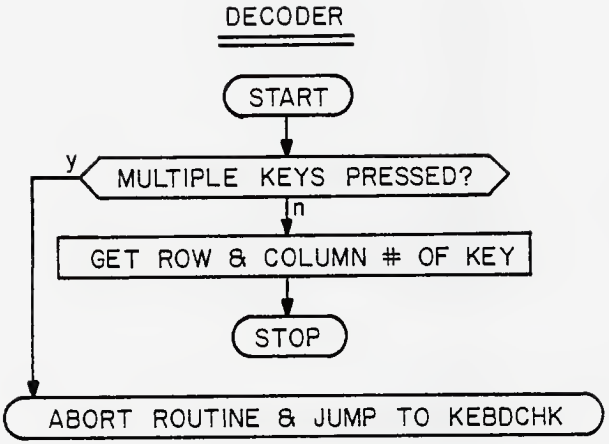


Figure AIII.11. Flowchart of Decoder

Decoder:

```
DECODER   LD A,B          78  1384  get current mask
          RLCA            07     5  check next two rows to
          OUT (C),A       ED     6  be sure there isn't more
                          79     7  than one key depressed.
          IN B,(C)        ED     8  if more than one key is
                          40     9  depressed then pop return
          JRNZ BADKEY     20     A  address from stack and
            (+26)         24     B  jump into KEBDCHK at
          RLCA            07     C  beginning
          OUT (C),A       ED     D
                          79     E
          IN B,(C)        ED    8F
                          40    90
          JRNZ BADKEY     20     1
            (+19)         17     2
          RLCA            07     3
          RLCA            07     4
          CPL             2F     5
          LD D,A          57     6  row containing pushed key
          XOR A           AF     7  set A to zero
          RRH             CB     8  determine which column the
                          1C     9  pushed key is on
          JRC +5          38     A
                          03     B
          INC A           3C     C
          JR -5           18     D
                          F9     E
          JRNZ BADKEY     20    9F
            (+5)          03    A0
          ADD A,D         82     1  assign unique value to D
          LD D,A          57     2
          RET             C9     3  return to caller
BADKEY    POP AF          F1     4  abort DECODER and jump
          JP RLOAD        C3     5  to KEBDCHK
                          B0     6
                          13  13A7
```

A III.18. Comments on Keyboard Checker (KEBDCHK)

This subroutine checks the keyboard for keys which have been pushed, insures that only one key was pushed at the time the key was spotted, and then returns a unique code to the caller for each key. In the case of a digit key, the value of that digit is returned. There is no input parameter. The output parameter is the indication of which key was pressed and the digit value in the case of a digit key.

register usage:

A - general purpose
B - keyboard row mask
C - keyboard port address
D - return parameter from keyboard
E - digit flag
H - input from keyboard
L - digit return parameter
D' - timer delay length parameter
E' - ditto

subroutines called:

STMR
DECODER

using KEBDCHK:

inputs: none

outputs: function key code in register D; digit code in register L if function key pressed was "enter"

registers affected: A, B, C, D, E, H, L, D', E'

keyboard return codes (in hex):

"enter"	22	for digit keys the digit
"run"	23	itself is returned in L
"?"	24	
"end of list"	25	
"cont"	26	

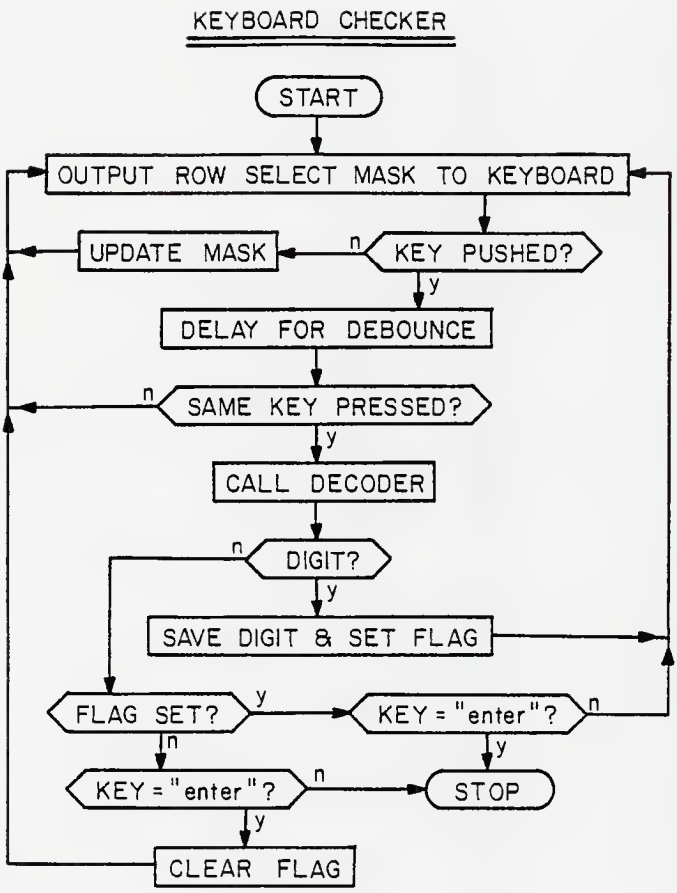


Figure AIII.12. Flowchart of Keyboard Checker

Keyboard Checker:

```
KEBDCHK   EXX              D9  13A8  set timer parameters
          LD DE,1000       11    9   to waste approximately
                           00    A   0.1 sec in the short
                           10    B   timer
          EXX              D9    C
BEGIN     LD E,00          1E    D   initialize
                           00    E
          LD C,E           4B   AF
RLOAD     LD B,FE          06   B0   load keyboard mask
                           FE    1
OUT       OUT (C),B        ED    2   output mask to keyboard
                           41    3
          IN A,(C)         ED    4   read data from keyboard
                           78    5
          JRNZ VDATA       20    6   jump if a key is pushed
            (+11)          09    7
          RLC B            CB    8   change keyboard mask
                           00    9
          JRC OUT          38    A   try all mask possibilities
            (-8)           F6    B   and then delay before
          CALL STMR        CD    C   starting over
                           76    D
                           13    E
          JR RLOAD         18   BF   check keyboard again
            (-15)          EF   C0
VDATA     PUSH AF          F5    1   save input
          CALL STMR        CD    2   delay before checking again
                           76    3   this is for debouncing
                           13    4
          POP AF           F1    5   recover input
          IN H,(C)         ED    6   read keyboard again
                           60    7
          CP H             BC    8   check to see if same
          JRNZ RLOAD       20    9   if not try again
            (-25)          E5    A
          CALL DECODER     CD    B   decode key and check for
                           84    C   multiple keys
                           13    D
          CP 20            FE    E   check for digit 8
                           20   CF
          JRZ IS8          28   D0   if key is 8 then jump
            (+6)           04    1
          CP 21            FE    2   check for digit 9
                           21    3
          JRNZ ISNT9       20    4   jump if not 9
            (+11)          09    5
IS8       ADD E8           C6    6   set A to actual digit
                           E8  13D7
```

Keyboard Checker:

```
ISDIGIT   LD E,FF          1E  13D8  set digit flag
                           FF    9
          OUT (FE),A       D3    A  load digit in rightmost
                           FE    B  display digit
          LD L,A           6F    C  save digit
          JR RLOAD         18    D  go back and wait for next
            (-45)          D1    E  key
ISNT9     AND E0           E6   DF  check for digits 0-7
                           E0   E0
          JRNZ ISNTDIG     20    1  jump if not digit
            (+7)           05    2
          LD A,D           7A    3  get digit code
          AND 0F           E6    4  set A to actual digit
                           0F    5
          JR ISDIGIT       18    6  jump and output digit
            (-14)          F0    7
ISNTDIG   XOR A            AF    8  set A to zero
          CP E             BB    9  check for digit flag set
          LD A,22          3E    A  load A for check for
                           22    B  "enter"
          JRNZ TEST        20    C  jump if digit flag set
            (+11)          09    D
          CP D             BA    E  if "enter" key pushed
          JRNZ RLOAD       28   EF  without digit, ignore it
            (-63)          BF   F0
RETURN    EXX              D9    1  reset timer parameters for
          LD DE,0005       11    2  1 second delay in long
                           05    3  timer
                           00    4
          EXX              D9    5
          RET              C9    6  return to caller
TEST      CP D             BA    7  check for "enter"
          JRZ RETURN       28    8  if "enter" (after digit)
            (-7)           F7    9  return to caller
          LD A,0F          3E    A  for non-"enter" key after
                           0F    B  digit clear rightmost
          OUT (FE),A       D3    C  display digit and ignore
                           FE    D  both keys
          JR BEGIN         18    E
            (-81)          AD  13FF
```

AIII.19. Comments on Test Routines

The following pages contain several test routines which are useful in debugging hardware problems. They assume a working processor board. Faulty processor boards are most easily debugged in another, known good, system. If the routines are used in the order presented, they allow the user to debug one section of hardware at a time, often using that section to help test other sections later. Extensive documentation is not provided, and it is recommended that only persons familiar with the hardware and Z-80 programming attempt to debug the system.

Semi-exhaustive Diaplay Test:

This routine tests the display by writing all possible characters to each digit.

```
LD SP,1FFE      31
                FE
                1F
LD A,0F         3E
                0F
LD C,F0         0E
                F0
OUT (C),A       ED
                79
INC C           0C
JRNZ -3         20
                FB
LD C,F2         0E
                F2
LD B,F0         06
                F0
OUT (C),B       ED
                41
LD DE,ACE0      11
                E0
                AC
PUSH DE         D5
POP DE          D1
INC DE          13
LD A,D          7A
OR E            B3
JRNZ -5         20
                F9
INC B           04
JRNZ -13        20
                F1
INC C           0C
JRNZ -18        20
                EC
LD C,F0         0E
                F0
LD A,01         3E
                01
INC C           0C
INC C           0C
OUT (C),A       ED
                79
INC A           3C
DEC C           0D
INC C           0C
JRNZ -7         20
                F7
HALT            76
```

Display and Keyboard Test:

This routine reads one key from the keyboard and displays the column number of the key.

```
          IM 1              ED
                            5E
          EI                FB
          LD A,06           3E
                            06
          OUT (00),A        D3
                            00
          NOP               00
          LD A,FF           3E
                            FF
          LD C,F0           0E
                            F0
CONT      OUT (C),A         ED
                            79
          INC C             0C
          JRNZ CONT         C2
                            XX
                            XX
          IN (00),A         DB
                            00
          OUT (FE),A        D3
                            FE
          HALT              76
```

A/D Board Test (MP-20 Module):

This routine tests the MP-20 module for proper operation. The test first sets up the module and halts. After an interrupt occurs, the system starts sampling channel 0 and displaying the results.

IM 1	ED	22
	56	23
EI	FB	24
LD SP,1FFE	31	25
	FE	26
	1F	27
LD HL,100F	21	28
	0F	29
	10	2A
LD DE,8000	11	2B
	00	2C
	80	2D
LD C,F0	0E	2E
	F0	2F
LD A,0F	3E	30
	0F	31
OUT (C),A	ED	32
	79	33
INC C	0C	34
JRNZ -3	20	35
	FB	36
HALT	76	37
LD BC,0000	01	38
	00	39
	00	3A
PUSH DE	D5	3B
PUSH HL	E5	3C
PUSH BC	C5	3D
POP BC	C1	3E
POP HL	E1	3F
POP DE	D1	40
INC BC	03	41
LD A,B	78	42
OR C	B1	43
JRNZ -9	20	44
	F5	45
LD A,00	3E	46
	00	47
LD (HL),A	77	48
LD A,(DE)	1A	49
RLD	ED	4A
	6F	4B

```
RRCA              0F      4C
RRCA              0F      4D
RRCA              0F      4E
RRCA              0F      4F
OUT (FC),A        D3      50
                  FC      51
LD A,(HL)         7E      52
OUT (FE),A        D3      53
                  FE      54
JR -29            18      55
                  E1      56
```

Test Digital Output Board and D/A"

This routine tests the digital output board and D/A by reading channel 0 and listing the data to the digital output board and the D/A.

```
LD SP,1FFE      31
                FE
                1F
LD A,00         3E
                00
OUT (EF),A      D3
                EF
LD A,0F         3E
                0F
LD C,F0         0E
                F0
OUT (C),A       ED
                79
INC C           0C
JRNZ -3         20
                FB
LD HL,8000      21
                00
                80
LD DE           11
                E0
                AC
PUSH DE         D5
POP DE          D1
INC DE          13
LD A,D          7A
OR E            B3
JRNZ -5         20
                F9
LD A,(HL)       7E
OUT (DF),A      D3
                DF
OUT (30),A      D3
                30
JR -15          18
                EF
HALT            76
```

A VERSATILE MICROPROCESSOR BASED DATA ACQUISITION SYSTEM FOR A BIOENGINEERING INSTRUMENTATION LABORATORY

by

PHILIP NOLAN KING

B. S., Kansas State University, 1976

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

ABSTRACT

A versatile microprocessor controlled data acquisition system was designed and fabricated for use in a bioengineering instrumentation laboratory. It provides for up to eight channels of analog input signals ranging from 500 microvolts to 10 volts peak-to-peak. The maximum sampling rate of 19,000 samples per second can be increased to 80,000 samples per second with minor software modifications. Formatted data is transferred to a computer for processing and storage using either of two digital interfaces. The operating system was designed for flexibility and ease of use.