A microprocessor-controlled vectorcardiograph

ISU
1982
Sm67
c. 3

nw

by

Therese Mary Smith

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major: Biomedical Engineering

Approved:

Iowa State University
Ames, Iowa

1982

TABLE OF CONTENTS

DEDICATION

To Alan John Luse

INTRODUCTION

Researchers are currently determining diagnostic criteria from vectorcardiograms. Non-invasive detection of infarct location, size and presence of a second infarct near an older one can be accomplished. The measurements and calculations required in the comparison of a vectorcardiogram with these criteria can be done by computer; thus, relieving physicians of much tedium, enabling them to spend a greater amount of time on higher professional skills. In 1970,

> In general, there is no difficulty in formulating an adequate algorithm for the diagnostic evaluation of vectorcardiograms to be followed by a computer. However, an algorithm for the pattern detection of vectorcardiograms cannot be adequately formulated at the present time...consequently in many cases the value of computer analysis is seriously limited.

In 1978, the work of Gustafson, Willsky, Wang, Lancaster and Triebwasser (14,15) was published, describing computerized statistical techniques for detection of arrythmias based on R-R intervals. In 1971, at the second International Federation for Information Processing, Technical Committee on Information Processing in Medicine, (IFIP TC-4) Working Conference on Computer Application on ECG and VCG Analysis, Wolf, MacFarlane, Friedricks, Duicmetiere, Werhrer, Van Bemmel, and Smith each discussed programs implemented for wave recognition (47). Others, e.g., Pipberger, et al. (30), have produced computer programs for VCG waveform analysis. In order to provide this diagnostic capability to the practicing clinician, it would be desirable to develop an inexpensive instrument which would be straightforward for a technician to operate.

Microprocessor technology, including compatible analog-to-digital (A/D) and digital-to-analog (D/A) converters, may be exploited to this end. This thesis describes the design and implementation of a microprocessor-based vectorcardiographic data acquisition system. The controlling firmware is written to provide support for modular algorithms for analysis of VCG waveforms.

A design for a microprocessor-based vectorcardiograph was proposed by Anil Sahai in his doctoral dissertation: The Design of a Microprocessor-Controlled Vectorcardiographic System (34). Due to the introduction of improved and less expensive parts, the hardware and software have been redesigned. The design and implementation are presented herein.

REVIEW OF LITERATURE

Vectorcardiography

The vectorcardiogram is a planar projection of the heart dipole. Frontal, left or right sagittal and transverse (horizontal) views are taken. These displays are constructed by plotting one electrocardiographic lead versus another instead of versus time. The passage of time would not be evident in the simple two-dimensional picture, therefore, the line is modulated in some way to indicate the direction and duration of time corresponding to a particular segment. Dashes and teardrops shapes are used. In a more sophisticated presentation suggested by Brinberg (cited in Wartak (43)), cones are drawn depicting the direction by their orientation and extent of time by spacing. A recent technique employs color to clarify the pattern (20).

The arrangement of electrodes used most frequently is the Frank lead system (8), which is based on a homogeneous torso model (10). There is some controversy over which lead system is best (see, for example, Zywietz and Schneider (47)), and much of the use of Frank leads involves modification of the electrode placement (32). It is not necessary to change the lead placement to compensate for the position of the heart in the individual patient, as an electrical circuit, called a resolver, exists which can perform rotations of the vectorcardiogram (43). A discussion of the sensitivity of diagnostic programs using the Frank lead system to mispositioning of leads is given in Greiser and Freidricks (12).

A representation of the Frank lead system is shown in Figure 1. The resistors perform a linear combination of the body-surface potentials, producing the three leads--X, Y, and Z. The three planes are developed from the leads as follows: Transverse: X vs Z; sagittal: Z vs Y; and frontal: X vs Y. Figure 2, also after Wartak (43), shows a normal vectorcardiogram. Different segments of the waveforms provide information of diagnostic value. The section of the pattern from the isoelectric or E-point to the O-point (see Figure 2) is called the P-loop and is produced by atrial depolarization. As atrial repolarization is not completed before ventricular depolarization begins, the P-loop is not closed, i.e., the O-point is not equivalent to the E-point. Ventricular depolarization produces the QRS-loop, the largest in area of the three loops. The last excursion of the curve is called the T-loop. It is produced by ventricular repolarization. The characteristics of the loops, the separation of their endpoints, and the instantaneous position of the vector are all used in diagnosis. For a thorough introductory description of vectorcardiography, see Wartak, Simplified Vectorcardiography (43).

Normal values and variation in vectorcardiograms are being obtained as a function of age (9,13), body size, sex (45) and race (2), for the purpose of diagnosis of abnormal patterns. Diagnostic criteria are being determined in many disease states, including ventricular (1,39,41) and atrial hypertrophy (9,46), atrial flutter (46), conduction defects--Wolf-Parkinson-White syndrome (40), sick sinus panconduction (31), and blocks (4). Vectorcardiographs give information about old infarcts (22), localization (27, 28, 33) and size (44) of infarcts, perioperative infarcts (24), diaphragmatic myocardial infarcts (37) and especially posterior infarcts, which can only be determined by vectorcardiography (6). Ischemia (7), hypertension (2), aortic regurgitation (41) and coronary occlusion (16,35) have known effects on vectorcardiograms. The vectorcardiogram can be used to measure pulmonary arterial pressure in the presence of chronic rheumatic mitral valve disease (36), to detect aberrant chromosome types (23) and to monitor the extent of damage to the heart in Fabry's disease (25). Various features of the vector pattern are utilized in diagnosis: the three planar projections (27); instantaneous values of magnitude and direction of the vector in the planes (33); spatial magnitude (from the Pythagorean theorem); and angle (41), ratios of magnitudes, and distances between points. The difference between features from a prior pattern to a present one are used (33), and at least one group of researchers has suggested that the variability of the pattern over time in one individual may be significant (47).

Some researchers obtain orthogonal potentials by means of ECG electrodes connected through a resistor scaling network to Grass recorders. Magnetic tape recordings of analog signals are later fed to A/D converters attached to minicomputers for data analysis (26,47). Others have used commercial vectorcardiograph units (27,46). At least one of these units is no longer available for sale[1]. The functions required of such a unit are diverse.

The measurements and computations required can be accomplished more efficiently by computer if the computer can be instructed to recognize the waveform and to perform the analysis (43). Many programs are already available. In Computer Application of ECG and VCG Analysis (47), specific programs are described. For example, in one article, "Conduction Defects with and without Myocardial Infarction", by J. Gelin (11), a program is described which can give diagnoses for complete left bundle branch block, complete right bundle branch block, left hemiblocks, and conduction defects with myocardial infarction. Several arrythmia detection programs are cited in the two articles by Gustafson, Willsky, et al. (14, 15).

In Friedman, Diagnostic ECG and VCG, criteria are given for the diagnosis of left and right ventricular hypertrophy or enlargement, Wolf-Parkinson-White syndrome, (based on individual VCG patterns), and a host of associations of disease conditions with arrythmias. This book contains many references to special articles where diagnostic criteria may be found. Wartak, in Simplified Vectorcardiography gives decision trees (the semantic content of a program, rather than the instructions themselves) for diagnosis of left ventricular hypertrophy, right ventricular hypertrophy, biventricular hypertrophy, and Wolf-Parkinson-White syndrome.

[1]Personal communication with Mike Kellen, Medical Sales Engineer, Hewlett Packard Co., Iowa City, Iowa.

Microprocessors in Instrumentation

While the computer systems on which these programs run may be more expensive than is feasible for some situations, the use of microprocessors may bring the cost down sufficiently to make these diagnostic programs more available (18). For example, a recent business reply mail advertisement from Abbott Medical Electronics states:

> We've eliminated the computer from our new computerized ACS Arrythmia Detection System, using new microprocessor technology. And we've cut our selling price nearly in half while increasing the system's clinical usefulness.

It is also important that an instrument be simple to learn and use (42). By using microprocessor components, instrumentation can be designed which is tailored to a particular application, i.e., many features which are not of interest can be left out, and those which are desirable incorporated without significant impact on price. Peatman describes how this is done in Microcomputer-based Design (29). This book provides an introduction to design with microprocessors, but is written for those who intend to design systems at the component level. An introduction which is more appealing, and has the additional benefit of hands-on experience is use of the Heathkit Microprocessor Trainer (E2). The educational literature associated with the kit provides an elementary but thorough description of microprocessors. Experiments with the hardware are clearly spelled out, so that the user is painlessly exposed to the use of the microprocessor. The kit with microprocessor, associated components, and educational literature costs $300 as of June, 1981. Korn's book, Microprocessors and Small Digital Computer

Systems for Engineers and Scientists (21), presents microprocessor-based design in perspective with other computer-based laboratory instrumentation. This book is very well-organized, presenting the wealth of detail necessary to begin a design in such a fashion that the sought-for information is readily found. Finally, further references are given after each section. This book seems to reach its goal, to

> help readers understand the manufacturers' literature and to make informed choices of microprocessors, minicomputers, peripherals.

An eminently satisfying book is Microprocessors in Instruments and Control by Bibbero (3). This book reviews fundamentals of data acquisition and control then digital logic before describing microprocessors. Ample references are given.

Last, for articles describing implementation, design and debugging specifics, it may be helpful to look through Applying Microprocessors from the Electronics Magazine Book Series, Altman and Scrupski (eds.) (1).

There are many features of the vectorcardiogram which may be of diagnostic value. Provision of all of these capabilities in one device may lead to a very complicated and/or expensive piece of equipment. If a family of instruments could be developed, each member of which answers the needs of some particular investigators, without incorporating all features, a reduction in cost and complexity could result. This set of related instruments could be developed by designing hardware to acquire raw vector data, and modular software routines to analyze it. Selective insertion of the program modules into the basic vectorcardiograph, which microprocessor technology makes possible, could accomplish this end.

PROBLEM DESIGN REQUIREMENTS

The design requirements are divided into five areas: quality of signal measured, safety of instrument, quality of display, simplicity of use, and ease of improvement.

To be of diagnostic value, the Frank lead signals must be accurately represented from 0.2 Hertz to 200 Hertz (43). To satisfy the Nyquist criterion, then, a sampling (and conversion) rate of 400 Hertz per lead is required of the A/D converter and multiplexer. Real-time processing of the signal implies a conversion turn-around time of less than or equal to 0.8 milliseconds or 800 microseconds. Turn-around time also includes software execution time for servicing interrupt requests. The time required for a conversion can be measured using a triggerable logic analyzer with interval timing capability. Measurements of voltage on each lead must be made at the same time (47).

A high input impedance on the leads is required both to improve a signal quality and to minimize the current drawn from the patient.

The bandwidth and resolution of the display must be sufficient for the waveforms' highest frequency components to be seen. Resolution can be estimated by comparing the ratio of the diameter of the illuminated spot to the linear dimension of the display, with the ratio of the change in voltage to be discerned to the voltage excursion of the segment containing the voltage change.

The controls of the device must be readily understood from the user's viewpoint. There should be no settings of the input devices which are meaningless or ambiguous.

The design should be modular, so that improvements can be readily implemented. Thorough documentation, obviously, is necessary for both hardware and software.

SYSTEM DESCRIPTION

The function of the instrument can be subdivided as follows:

1. acquisition of analog ECGs including amplification
2. filtering and sampling of the data
3. digitizing
4. storage in memory
5. control panel
6. retrieval from memory
7. digital manipulation (i.e., computation)
8. display

These functions are implemented within the instrument. A block diagram showing the relationship of these functions is given in Figure 3.

Acquisition and Amplification of ECGs

The Frank lead system, the most widely used in clinical instruments as described in Biophysical Measurements (38), is employed. The leads are buffered to provide high input impedance (for electric shock reduction and signal improvement). The buffering is followed by the scaling network of the Frank lead system. The X, Y, and Z leads are then amplified by instrumentation amplifiers, filtered by second-order Butterworth filters to remove aliasing, which would otherwise be introduced by subsequent sampling, and the signal raised to an offset of 2.5V, to center the signal in the range of the input of the A/D converter. The lead signals are then coupled to the multiplexer inputs.

In van Bemmel et al. (42), a summary of filter cutoffs used by researchers is given. Most are using approximately 200 Hertz, and only one is using above 500 Hertz. The X, Y, and Z leads are sampled sequentially by a multiplexer, which feeds the selected analog channel to a sample/hold amplifier. The sample/hold supplies the SAR (successive approximation register) A/D (analog-to-digital) converter which provides digital data to the microprocessor via the data bus.

The analog channel switched to the sample/hold also drives the display circuitry so that the data being gathered are displayed in real time. Frank leads can be monitored in real time by this connection. This is designated Real Time Mode. In Stored Data Mode, the output of the sample/hold is input to the A/D converter, which outputs digital data on the microprocessor data bus. The A/D converter interrupts the microprocessor with an "end-of-conversion" signal, at which time the processor stores the data in memory. The data can then be retrieved by the microprocessor and sent to a D/A (digital-to-analog) converter, where they are converted to a signal for driving the cathode ray tube. The output of the D/A is routed to one of the channels of the analog mutliplexer used in data acquisition, so the same multiplexer (through which real time data are sent to the CRT) can select the previously stored data.

The display is a low persistence CRT X-Y monitor, with a bandwidth of 4 kilohertz (E13). The functions of the vectorcardiograph are controlled from a front panel. ECG or VCG display can be selected, either in real time or from stored data. When ECG data are displayed, the horizontal scan is supplied from a sawtooth generator circuit (selected through the multiplexer). Presently the X, Y and Z leads are available, but simple modification will allow for the bipolar leads I, II and III to be observed as well. Any of the three VCG planes may be selected, either in real time or stored.

When Stored Data Mode is selected, two sets of 3-digit BCD switches are used as inputs to the processor to define the range of samples of the VCG pattern to be displayed. For example, if it were desired to examine the T-loop without interference from an overlapping P-loop, the switches could be set to display only the desired portion of the waveform. The output is updated as the switches are rotated, so that lengthening or shortening either end of the waveform can be done with rapid visual feedback. The resolution of the switches is one sample per least significant digit, as the waveform is made up of 1000 samples per lead. Should a range of points be selected such that the starting point is higher that the stopping point, a range error light is illuminated.

In summary, the ECG data are conditioned and input into a multiplexer and can be displayed in real time or stored and then displayed with adjustable blanking. X, Y, Z, X-Z, X-Y and Y-Z waveforms can be shown.

Patient Interface

The patient interface is that part of the circuit which touches the patient or whose design is influenced directly by the requirements of the patient.

In the vectorcardiograph, the interface comprises the electrodes, the leads, buffer amplifiers, resistor network for scaling and combining the input (Frank lead system), instrumentation amplifiers, the high-pass filters which prevent formation of sum and difference frequencies which come about due to sampling by the multiplexer, and the conditioning of the signal from a bipolar signal to one varying from 0 to 5 volts. Commercially available electrodes are used. Buffering is presently accomplished by a 741 operational amplifier non-inverting follower configuration which provides a higher impedance to the device than it would otherwise have. An input impedance of two megohms is supplied by the 741.

To prevent leakage current entering the patient in the event of device abuse or failure, optoelectronic couplers such as the MCT2 by General Instrument or the 4N26 by Hewlett-Packard or Motorola could be placed in series with the operational amplifier. These operational amplifiers and the patient side of the optoelectronic isolator could be battery powered for isolation of the patient from the AC power mains. Optocouplers are available with insulation voltages of 3000 volts (E3).

The output of the optocoupler, if used, is scaled by the Frank lead system resistor network. The network used was taken from Biophysical Measurements (38) and is shown in Figure 1. After scaling, the X, Y, and Z leads are available as differential signals. These are amplified by instrumentation amplifiers tailored for a gain of 1000 (E1).

The amplified signal is then filtered by a second-order Butterworth filter. The cutoff frequency is 500 Hertz, and the response is flat out to beyond 200 Hertz which is the high-frequency cutoff required for diagnostic measurements (43).

The multiplexer will handle inputs in a range centered at 2.5 volts. A range of 0 to 5 volts was chosen. The output of the filters is bipolar; therefore it is followed by a summing amplifier configuration with a potentiometer to control the voltage added. As the patient requirements have been met and the signal has been conditioned to the requirements of the multiplexer, this completes the design of the patient interface.

Front Panel

The front panel is an interface between the device and the hospital or research personnel using the instrument. The front panel must provide the means of selecting the options of which the instrument is capable, without being so cluttered that it is too difficult to understand, or unwieldy to use in a crisis situation. It is an advantage to have a "soft" front panel, i.e., one for which the microprocessor can control the settings of the switches. For example, if one were to select VCG mode and X-Y plane, and then select X-Z plane, on a soft front panel the microprocessor could automatically deselect X-Y plane perhaps displaying the current state of the instruments via light emitting diodes (LEDs). Another method of preventing selection of incompatible features is the multiposition rotary switch, but this does not have the advantage of software selection of options on startup or reset.

The front panel used in this design is extremely simple, requiring only two PIAs (peripheral interface adapters) to interface it to the microprocessor. Toggle switches were chosen to select among the different options. Soft switches could be implemented as shown in Figure 4, requiring significantly more parts and supply current.

The front panel includes a momentary switch labelled "Record" which sends a non-maskable interrupt (NMI) to the processor. Use of the NMI pin for this function reflects a choice of priority for the different functions. Beginning a record of data is regarded as the highest priority function of the instrument. It is envisioned that the physician may see an interesting waveform on the real time display and decide to record the ECG signals immediately. Use of a first-in-first-out (FIFO) buffer at the output of the A/D would expand the functionality of the instrument to include a record of information from the immediate past.

The interrupt, which is of lesser priority, is used to signal end-of-conversion from the analog-to-digital converter. A standby switch to inhibit any display while the switches are being set up, and a light emitting diode (LED) indicator which lights for non-meaningful combinations are also provided.

Internal Design

Interfacing the A/D, D/A, PIAs and memory to the microprocessor was made relatively easy by the availability of parts designed for microprocessor bus architecture.

For the analog/digital conversion, a National ADC0817 was chosen (E10). Integrated injection logic technology permits fabrication of analog and digital parts on the same substrate and this chip combines a 16-channel analog multiplexer with an 8-bit successive approximation register. Bus compatible address pins are provided for channel selection. The chosen input is brought out to a common pin to enable the designer to save parts by processing only the chosen signal. This circuit places a sample/hold amplifier at the common output (E9). The output of the sample/hold is fed into the analog-to-digital conversion portion of the chip. The microprocessor supplies the start pulse to the converter, and the latch control to the sample/hold. The conversion takes approximately 100 μsec, during which the hold value changes by 10 microvolts, or about 1%. End-of-conversion is signalled to the microprocessor via CA1 input of a PIA, which pulls the lesser priority interrupt line low. The interrupt is handled by a routine which stores the data in memory. The A/D latches the value onto the data bus for the microprocessor to load (read), and the microprocessor load (read) instruction to the A/D address initiates another conversion. In between the interrupts, the microprocessor is free to execute any algorithms, such as described by Gustafson et al. (14, 15). In the current design, no processing of the data is done.

One thousand data points per lead are gathered and stored in memory. On retrieval, the data are accessed as directed by the decimal digit switches. Three digits identify the first sample point to display and the other three digits indicate the last piece of data. These switches are interfaced to the microprocessor through two PIAs. PIAs are part of the Motorola 6800 family of parts, and can be connected directly to the microprocessor buses. They provide two 8-bit wide programmable I/O (input/output) ports, interrupt lines to and from peripherals, and interrupt lines to the processor. The BCD switches provide a 4-bit binary output for each digit.

Display

The D/As used are Signetics SE 5018's which are designed for microprocessor compatibility (E11). The input data are latched. The device provides a voltage output (0-10 volts), which is routed to the analog multiplexer to be displayed. Before the signal is connected to multiplexer it must be conditioned. The voltage is divided in two by a voltage divider so that the maximum is 5 volts and filters (low-pass Butterworth) are used for deglitching.

Bus connections between boards are made using 12" ribbon cable, with alternate leads grounded, to reduce coupling of one signal lead to the next. Power was transmitted to the boards on separate wires. The bus transceivers used were Signetics 8T26s (E12). These chips can drive 18" of unterminated line, provide 40 milliamperes of current sink on outputs, and require 25 microamps for all inputs. All the bus transceivers' supplies were bypassed with 0.1 microfarad ceramic capacitors.

The use of high current drivers and microprocessor compatible parts simplified the design considerably.

SYSTEM DESIGN DETAIL

Patient Interface

Please refer to Figure 1 which shows the buffer amplifiers and scaling network. Many kinds of patient electrodes are commercially available. Therefore, no development of electrodes was undertaken. Leads from the electrodes are buffered by 741 monolithic integrated circuit (IC) op amps (operational amplifiers). The operational amplifiers provide a high-impedance input (two megohms) on the patient side, and a low-impedance (much less than 75 ohms) on the resistor network side. The 741s may be expected to draw 0.001 microamps from the patient, which is not expected to produce any shock (personal communication, Dr. Curran Swift). The operational amplifier is connected in a unity-gain follower configuration; therefore, its output is expected to be bipolar with respect to the right leg reference, of millivolts in amplitude (41), and will roll-off at 20 dB/decade, with a cutoff frequency of approximately 700 kilohertz. The outputs of the buffer amplifiers (741s) are connected to the Frank lead system scaling network (resistance values are noted in Figure 1) as described in Biophysical Measurements (38). A value of 10 kilohms was used for R. The output of the scaling network comprises the three Frank leads: Vx, Vy, and Vz.

ECG Signal Acquisition and Amplification

The three Frank leads carry bipolar differential signals on the order of millivolts. It is desirable to condition these signals for the input to the analog-to-digital converter (A/D). The circuit from the differential signals up to the A/D input is given in Figure 5. This circuit comprises a Burr-Brown instrumentation amplifier, (either model 3660 or 3662) set for a gain of 250. This is followed by a low-pass filter designed for a high-frequency cutoff of 500 Hertz. The roll-off at high frequency is 40 dB/decade. The filter is a second-order Butterworth, which is used by several research groups (45). This particular implementation of a second-order Butterworth active filter was obtained from Rapid Practical Design of Active Filters (17). The function of this filter is to attenuate frequencies of 500 Hertz or higher. Later in the signal processing, sampling will occur, which will fold back signals at a given frequency into sum and difference frequencies. This is called aliasing. For example, for a sampling frequency of 1 kilohertz, a signal component of 800 Hertz will appear at 1 kiloHertz - 800 Hertz = 200 Hertz, and at 1800 Hertz (1 kiloHertz + 800 Hertz). If we are concerned about signal accuracy around 200 Hertz, and not around 800 Hertz, we can attenuate any signal around 800 Hertz, so that when it is folded back, its contribution will be small compared to the signal of interest. Most researchers find no useful information above approximately 200 Hertz; they use a low-pass filter whose cutoff is 300 Hertz or less (47). One researcher uses 50 Hertz (47). This circuit uses 500 Hertz; therefore, frequency response should be more than adequate. A voltage greater than the maximum expected QRS peak (10 millivolts) was amplified to 2.5 volts, leading to a gain factor of 250.

Sampling and Conversion

The sampling and conversion is accomplished by the circuit shown in Figure 6. The inputs are from the band-pass filters described above. The National ADC0817 is a combination 16-channel multiplexer and successive approximation A/D converter. Four address lines are used to select the input channel, which becomes connected to the so-called common output. At this output, any circuitry which it may be desirable to place before the converter is situated. In this design, a sample/hold circuit is shared by each analog channel. The sample/hold follows its input signal until the control input is brought low to cause it to retain data. The circuit for controlling the sample/hold was obtained from a National ADC0817 Application Note (E10).

The address lines are decoded to detect the address of the A/D which is sent out on the address lines by the microprocessor in response to either a read instruction or a write instruction. The R/W control line is used to signal the A/D to start a conversion and to cause the sample/hold to hold. The end of conversion output (EOC) is sent to a digital input of a peripheral interface adapter (PIA). The PIA input can be conditioned by the software to respond to a falling edge of a signal, which is appropriate in this case. Upon detection of the EOC signal, an interrupt is sent to the microprocessor. The interrupt handling subroutine reads the data from the A/D data output leads, which are connected via bus buffers (discussed later) to the microprocessor data inputs.

At the conclusion of the data gathering routine, the sample/hold is allowed to sample, the address of the analog channel is changed, and a new conversion is initiated. The sample/hold biasing circuit was obtained from National's Linear Data Book (E9). Both static and dynamic zeroing are made available to service personnel. Static zeroing is correction of output offset, as in an operational amplifier. Dynamic zeroing refers to removal of a step change in the output when the sample/hold control input makes a transition to low voltage.

Front Panel

The front panel makes available several toggle switches (five of which are currently handled by the software), six digits of BCD switches for selecting intervals of the VCG loops, a push button switch for recording (storing) of data, and an error light. The front panel interfaces to the VCG are shown in Figure 7.

The components labelled PIA are peripheral interface adapters. These digital interface units are members of the Motorola 6800 family of chips, that is, they are designed by Motorola to attach directly to the microprocessor address, data and control buses. They include internal registers which specify the mode of operation. For example, in the case of the EOC signal from the A/D converter, a pin is configured as an input which will respond to a falling edge by signalling the microprocessor on the interrupt line (discussed later). The PIA has two separate 8-bit peripheral ports. The 16-bit lines may be individually programmed as an input or an output. This design calls for one output line to control the error light. The remaining lines are configured (by the software) to act as inputs.

Each front panel toggle switch provides either a high (+5 volts) or low (0 volts) signal to its peripheral port pin. The data are read by the program as if the PIA were a memory location. The combination of the five-switch settings specifies a particular subroutine to the controller program, for example, the switch setting: Real Time, VCG, X-Y (high), Y-Z (low), X-Z (low) causes the program to invoke a subroutine to display the real-time frontal plane vectorcardiogram. Use of a three-position selector switch would have been superior to three toggles, as it would eliminate meaningless switch combinations, i.e., any with more than one pattern selected, e.g., X-Y (high) and Y-Z (high). The BCD switches supply four bits of binary data for each decimal digit displayed on the front panel. The logic is inverted, i.e., 0 is represented by 1111, 5 is represented by 1010. The program inverts the input from the BCD switches before converting the concatenation of three BCD numbers to a binary number.

The last input on the front panel is the Record button, which is connected to a microprocessor input named NMI. This active-low signal is held high by a pullup resistor, until brought low by pressing the switch. This input signals the microprocessor to suspend its current task, execute a subroutine specific to the NMI low interrupt, then return to the previous operation. In the present design, the NMI subroutine reads data from one of the A/D converters, until 1000 data points are collected from each lead, and stores the values in memory. The NMI processor pin is different from the IRQ processor pin in that its request cannot be ignored by the microprocessor (See Figure 8). By means of a mask bit, IRQ signals can be prevented from disturbing the program flow.

Memory

The memory is divided into two parts: program memory and data memory. The program memory does not change. It contains the instructions which control the instrument; these are retained in a 1024 x 8-bit programmable read only memory, an Intel 2708 (E4). The 2708 is UV (ultraviolet) erasable, as a convenience for program development. It has been superseded by the Intel 2758, which does not require the -5 volts and +12 volts supplies needed for the 2708. PROM (programmable read only memory) programmers for the 2758 are easier to build, as well.

The data memory must be changeable. It has been implemented by Intel 2114s. To provide 4K x 8 of read/write memory (RAM, or random access memory), eight 2114s are required. As the 2114s have bi-directional input/output pints, and respond to the read and write signals fast enough, they can be connected to the microprocessor address, control and data buses very simply.

Selection of a particular chip within a memory array is made by means of address bus decoding. A 74138 (3 bit to 1 of 8 line converter) is employed to interpret microprocessor address lines A14, A13, and A12. Different patterns of these three bits represent addresses of different parts of the circuit. The eight output lines are used as enables for these parts, hence the enables are named "PROM", "RAM", "PIA", etc. Selection of a particular chip within RAM memory is made by another 74138, which currently decodes only address lines A10 and A11.

The outputs of the 74138 which decodes A10 and A11 (RAM selector) are combined with the "RAM" select line and an inverted valid memory address (VMA) signal to form a set of chip select lines for the individual 2114s making up the 4K x 8 read/write memory. This combination is accomplished using 74LS27s, which are three-input NOR gates, followed by inverters (7404s). Only if RAM is in its active low state, the output from the RAM selector is in its active low state, and the VMA is in its active low state will the NOR output become high. As active low signals are desired for the RAM chips, the NOR gates are followed by inverters. An improved design would replace the NORs and inverters by using the additional enable lines to the 74138 RAM selector. Chip selection can be done without use of address bus decoders, when the entire address space is not required. To see how to do this, refer to Peatman (29), Chapter 3.

The 2708 is enabled by the "PROM" enable, described above. The address chosen for the PROM corresponds to A15, A13, and A12 all high, because on reset or power-up the microprocessor sends two addresses out on the data bus. The two bytes of data it receives from memory are concatenated to produce the address of the first instruction, which the microprocessor then fetches. The two addresses sent out on reset have A14, A13, and A12 high, and as it is desirable to store the first instruction address in the PROM, the PROM is enabled for A14, A13, and A12 high. There is an advantage in locating RAM in low addresses, i.e., A14, A13, and A12 all low. The advantage comes about via specialized instructions for accessing low-address memory. These instructions are faster to execute and take up less space in the PROM. To use these instructions with the RAM memory, the "RAM" enable corresponds to A14, A13, and A12 low.

The memory is located physically on a separate board from the microprocessor. To drive the signal lines between memory and microprocessor, high current capability bus transceivers were employed. Signetics 8T26s (quad bus transceivers) were used on both address and data lines for simplicity in stocking of parts (E12). These bus transceivers have tristate outputs, enabling the designer to place more than two components on a signal lead. By controlling each device connected to the bus, i.e., by forcing all but one device into the high-impedance (third) state, the remaining component can actively pull the signal lead high or low without competition from another part.

The microprocessor supplies control lines, such as $R/\overline{W}$ (read, not write) and Ø2 (phase 2 clock; also called data bus enable in this circuit) which are used in controlling the enables of the data bus transceivers. For example, the memory card should drive the data bus if either RAM or PROM is enabled, (i.e., the memory address is valid and address corresponds to RAM or PROM) and if a READ instruction is being executed ($R/\overline{W}$ is high). The memory board transceiver drive enable circuit is shown in Figure 9. The output of the second NAND is active low for (PROM or RAM) and READ. The signal is buffered by the transceiver which sends it to the memory board. As the transceivers are inverting, the signal arrives at the memory board active high, and there enables the data bus drivers for the memory board.

Microprocessor

The microprocessor board holds the microprocessor chip (Motorola 6800, see Figure 8), the clock chip (Motorola 6875), the address decoding circuits, some bus transceivers, and the manual reset pushbutton switch. The Motorola 6800 was selected as the microprocessor to use for this project because it and some other members of the microprocessor chip set were already owned by the department, and a cross-assembler (MSAM from Motorola) for 6800 assembly language was available at the I.S.U. Computation Center. Subsequent purchase of a 6800-based microprocessor trainer (E2) by the department facilitated testing of the individual components, by substitution (6800) or by use of the breadboards (2114, 6820, or 6821s). Software modules were also debugged using the trainer.

The 6875 clock chip produces three clock signals: Ø1 for the microprocessor, Ø2 for the microprocessor, and a higher drive capability Ø2 or enable for the peripherals. The frequency of the clock signals can be controlled by an RC circuit, as explained in the 6875 application note (E8). The 6875 also interfaces the reset pushbutton to the microprocessor, supplying the fast rise time required by the 6800, and making possible a simple input switch circuit. Clock frequency generation and input switch circuits are shown in Figure 10. A clock frequency of 0.8 megahertz is used in this circuit which allows straightforward use of the 2708 and the slowest (least expensive) version of the 2114s.

Cables

The address, control and data buses are delivered from the microprocessor board to the memory board and to the I/O board by means of ribbon cables. The ribbon cables are 16 lines wide; the address bus (A0 - A9) is delivered to the I/O board with some control bus signals on one cable. Capacitive coupling can occur, however. The ribbon cable carrying the Ø2 clock has every other conductor connected to ground on the microprocessor side. The cable from the data bus to the I/O board has alternate grounding as well.

Interface of Internal Parts to Input/Output

The interface to the front panel has been described. The PIAs receive the microprocessor address and control signals as delivered to the I/O board via a pair of bus transceivers; the PIAs also connect to the data bus. In the next section the digital-to-analog (D/A) converter circuit will be given. The D/As interface to the data bus lines as well. They are controlled by a signal "D/A" derived from the address lines as "PIA" and "PROM" were. There is a circuit which controls the latch for data on the data bus shown in Figure 11. Q is set to low on reset, so the data on the data bus are presented through transparent latches to the D/As. Should the D/A signal indicate one of the D/As is being addressed, at the falling edge of Ø2, the data on the data bus will be latched into the D/A which corresponds to the A0 address line. If A0 is high, the vertical output D/A is being addressed, and its latch will be brought high. The 7432 is a quad 2-input NOR, so the inversion of RESET and A0 is accomplished using spare gates from this chip.

Sawtooth Generator

The remaining internal circuitry consists of a sawtooth generator, which uses the circuit given in Figure 12. As indicated, different component values cause different slopes and final values of the ramps. The ramp is used to sweep the horizontal input of the CRT when ECG traces are being displayed. A sawtooth waveform suitable for intensity modulation of the CRT trace to provide the teardrop pattern which indicates direction and rate of inscription could be implemented with this circuit. The ramp circuit may be divided into two parts: the astable oscillator which supplies a ramp interval and a recovery interval, and the ramp generator. The oscillator circuit is taken from Coughlin and Driscoll (5), p. 241. A thorough explanation of the circuit is given therein. The low-frequency sweep generator is described in the <u>IC Op Amp Cookbook</u> (19), p. 389.

Display

The display is produced on a Telonic-Altair Model 4060 X-Y display. This inexpensive CRT monitor has an intensity control and a bandwidth of 15 kilohertz for vertical signals and a 4 kilohertz for horizontal. Grey-scale modulation can be accomplished by varying a voltage from between 0 to -100 volts, although from 0 to -20 volts is usually adequate (E13). The X and Y inputs to the monitor are obtained at the common output of the A/D converter, as shown in Figure 3. The output is amplified by a non-inverting configuration of an operational amplifier to allow for expansion of the selected segment. In _Real Time Mode_ the vertical signal comes from one of the Frank leads, through the "vertical" multiplexer, and the horizontal lead is selected by the other multiplexer. If a real-time VCG is being displayed, the horizontal multiplexer must present another Frank lead. For ECGs, the horizontal sweep ramp described above is selected through the horizontal multiplexer. For _Stored Mode_, the Frank lead digitized data are reconverted (by the D/As) to analog signals. These analog signals are presented to the same multiplexer accepting the original Frank lead input (but on separate channels), and through the multiplexer they can be displayed on the CRT.

The ramp generating the teardrop intensity pattern could be switched onto the intensity modulation input by the ECG/VCG front panel switch.

Firmware

Figures 13 and 14 are block diagrams illustrating the program structure. The program is written in modules or blocks. After the organization at the block level is discussed, a detailed description of each module follows.

Firmware Organization

The firmware is divided into three parts. The first is initiated by power-up or by activation of the Reset switch. Similarly, the second is invoked with the Record button, and the third by the EOC pulse output of the A/D converter. These three parts are illustrated in Figures 13 and 14. The section of the program initiated by start-up or reset initializes the PIAs; that is, it causes the peripheral ports to be used as inputs, with the exception of the pin used to drive the LED error signal, which is configured as an output. Then the peripheral port which is attached to the front panel toggle switches is read. The settings of the switches determine the procedure the microprocessor should follow. The code describing the specific sequence the microprocessor should follow in response to the switch settings is then executed. If the switch settings are not meaningful, the error light is lit. There are some instructions, e.g., those for loading an address into the multiplexer, which are executed for any valid switch setting. These are performed after the setting is decoded. After the low-order bits corresponding to the specific channel of the multiplexer are determined, the address must be latched in and conversion initiated.

If <u>Real Time Mode</u> has been selected, latching of the ECG or ramp leads into the multiplexers is all that is required to get a display on the screen, so the flow of control returns to checking the front panel. <u>Stored Data Mode</u> has additional requirements, namely the determination of the segment of the VCG/ECG pattern which is to be displayed. The PIA ports connected to the BCD switches are read, the binary values of the switches determined, and then compared. If the value indicating the beginning of the range exceeds the value for the end, the error light is lit. A spare PIA pin exists, to support separate error lights if desired. (See the discussion of suggested improvements.) If the range specified is valid, memory pointers corresponding to the end points of the desired display are set up, and data from this area of memory are written to the D/A converters. Then the flow of control returns to reading the front panel.

Upon depression of the Record button, the sequence which the microcomputer was currently executing is interrupted, and the second major division is entered. Recording data involves initializing a pointer to memory for data storage, latching the appropriate address into the multiplexer for the A/D converter, initializing conversion, reading the data when they are available, storing it in memory, updating the memory pointer, checking for completion, and either returning to gather more data or reestablishing the previously interrupted task. Responsibility for the listed tasks is divided between the second and third sections of the program. The third section reads the A/D converter output and stores data in the current memory slot. This particular division of labor takes advantage of the EOC pulse from the A/D converter to read the digital data as soon as they are available. The second part executes the remainder of the items listed.

Initialization of PIAs

The actual (hexadecimal) addresses of the PIA registers are in Figure 15a. The instructions for PIA configuration are given in Figure 15b. The addresses of the particular registers are determined by the wiring of the address lines to the PIA inputs. See Figure 7 for the PIA wiring. The 4 in the fourth hexadecimal digit causes the PIA enable to go low. Address bit A2 differentiates between the PIAs; it is low for PIA 1 and high for PIA 2. A2 is connected to an active high chip enable on PIA 1, and though an inverter to the same chip enable on PIA 2. Once the addresses are determined, they can be named as shown in the assembler directive EQU (equate) statements. If the names are used throughout the program, any hardware changes to the addresses can be conveniently propagated through the software by a change in the EQU directive, with subsequent assembly. The comments after the assembly language instructions explain their function. The particular numbers to be loaded into the control registers are determined from the 6820 (PIA) specification sheet (E6). Initialization of the PIAs consists of setting the data direction of the peripheral ports to inputs, with the exception of one bit, which controls the error light.

Determination of front panel settings:

Specific response

The switches on the front panel are connected to the PIA port. Their status is determined by reading the peripheral port, i.e., executing a load accumulator instruction using the port's address. The value appearing in the accumulator is used to call a subroutine specific to the front panel setting. These instructions are shown in Figure 16a.

The first two instructions initialize the stack to $08FF, which is located in RAM. The RAM begins at $0000 and extends to $3FFF. The lowest addresses in the RAM are used for variable storage. The ECG lead data are stored beginning at $0100. One thousand samples of each of three leads require 3000 spaces, or up to $0BB7. Placing the stack at $1000 allows it to grow (toward lower addresses) to more than adequate depth. The stack holds data after a PSH (push onto stack) instruction has been executed. If a subroutine is called or an interrupt handled, the stack holds the return address of the interrupted process. The stack pointer must be set to a location in RAM, so that data may be written into memory.

There is a table beginning at location $7000, which contains a pair of instructions for each of the 32 combinations of the five toggle switches. The number represented by the settings of the switches is used to select one pair. Each pair of instructions consists of a jump instruction to a specific routine and a return from subroutine (see Appendix B). The pair of instructions requires four bytes to express.

If the switch settings are considered as a binary number, then by adding four times that binary number to the address of the table, a specific pair is selected. This is done in the program by use of the ASL instruction (arithmetic shift left), which multiplies a binary number by two. The position of the switches in the peripheral port supplies the other factor of two.

The SCAN routine preloads an address, SWITCH, with the beginning address of the above table, then calls LITEOF (Figure 16 (a)). LITEOF turns off the LED, by forcing bit 0 to a 1 (see Figure 17 (a)). Then it masks out those bits which are not from the front panel switches, and multiplies the resulting binary number by two, as explained above. This computes the low-order byte of the address into the "table of subroutine calls" described above.

An example of a subroutine to handle the selection of stored (rather than real-time), vectorcardiogram for X-Y plane is shown in Figure 16 (b). The low-order bytes of two addresses are set. HMUX is an address specifying a multiplexer channel in the multiplexer whose common output drives the horizontal deflection of the CRT. VMUX is the same for the multiplexer providing the vertical drive to the trace. For stored data, the channels connected to the D/As must be addressed. OFST2 and OFST3 are variables containing an offset to be added to the base address of the table of ECG lead data. The table contains the X, Y and Z lead data, interleaved; X lead data has an offset of 0 bytes, Y of 1 byte and Z data, 2 bytes. OFST3 is used when VCG patterns are to be displayed. It holds the difference in location between the vertical axis data and that of the horizontal axis. OFST3 is added to OFST2 to determine the location of the data plotted in horizontal direction. Meaningless switch settings cause the error routine to be executed.

Determination of front panel settings:

Common response

Figure 17 contains some of the instructions which are executed for any front-panel setting. SETMUX is a subroutine which latches the channel addresses into the multiplexers by sending these addresses out on the address bus. The A/D1 and A/D2 pulses thus produced latch the four bits of channel address into the respective multiplexers.

Then the front panel switches are read once more, to determine the status of the Real-Time/Stored Mode switch. For Real-Time Mode, everything required for the display is accomplished, and the flow of control returns to scanning the front panel.

Display of stored data - determination of range

See Figure 18 for the instructions followed in the determination of the range for display of lead data which has been stored in memory.

First the rotary binary-coded-decimal switches used for specifying the range of stored data to be displayed must be read from the peripheral ports. The negative logic BCD representations are inverted by the complement instruction. Each 8-bit pattern is separated into two sets of four bits for each switch. Each set of four bits is stored right justified in an 8-bit byte. The three bytes representing the starting value are then loaded onto the stack. A subroutine which converts the three decimal digits to a 16-bit binary representation is called. The subroutine leaves the result on the stack. The binary representation is then pulled off the stack and stored. The same conversion procedure is followed for the three decimal digits specifying the end of the range of display.

After the start and stop values for the display have been determined, they are compared to ensure the stop value equals or exceeds the start value. If this condition is not met, the error routine is entered. Otherwise conversion and display are initiated.

Figure 19 shows the instructions used in the determination of the absolute address of the stored data. First, the starting address of the table is pushed on the stack, then the index of the entry required in the table, then the number identifying in which of the interleaved layers the entry sought is to be found. A 16-bit address is returned and stored. After the address of the vertical data starting point is determined, the address of the horizontal data is obtained.

The sequence of instructions which determines the address is shown in Figure 19b. It is a general purpose table-handling routine.

Once the absolute address of the ECG table as specified by the range switches is known, the data can be converted back to analog form for the display. The output routine is shown in Figure 20.

The byte to be converted for vertical output is loaded into the accumulator and written to the D/A connected to the vertical drive on the display. Then the process is repeated for the horizontal output. The addresses of the next horizontal and vertical data bytes are calculated. Then this current address for the vertical output is compared with the address for the last data byte to drive the vertical output. Until the current address exceeds the endpoint, data are converted and displayed. The D/A signal latches the data on the data bus into the converter, which holds the analog output until a new byte is latched in.

Recording of ECG lead data

The Record button brings the NMI input of the processor low, causing it to fetch the address of a routine from the PROM. See Figure 21. This routine is then executed in response to the NMI interrupt. The routine begins by clearing the interrupt mask, thereby enabling the IRQ interrupt line. This line will go low when the A/D produces an EOC pulse. The next instructions initialize the pointer to the next location in memory into which data should be written. The next instruction causes the sample/hold amplifier to hold by causing the $R/\overline{W}$ line to be high and the A/D enable to go low which triggers the circuit driving the sample/hold input of the amplifier. The STA A XLEADV initiates the A/D conversion by causing the $R/\overline{W}$ line to go low while the A/D enable is low. This triggers the circuit driving the START CONVERSION pin on the A/D.

The next pair of instructions set a flag, which is reset only when data have been loaded into the ECG lead table. After the data have been secured, the table pointer is incremented, and the procedure repeated for the next lead. After the Z-lead has been sampled, and the table pointer updated, the value of the pointer is compared with the location of the end of the table. When the table has been filled, the interrupt mask is set, and control returns to the routine which was interrupted.

The routine which responds to the IRQ interrupt reads the A/D converter output, stores that data in the location pointed to by the table pointer, and then resets the waiting for data flag. See Figure 22.

RESULTS AND DISCUSSION

The results will be discussed with respect to the same breakdown of functions as the circuit was described, first with respect to hardware, then firmware.

Hardware

Acquisition of the analog ECG was demonstrated to work via observation of the display when the instrument was functioning in Real Time Mode. Input was obtained from a Waveforma signal generator (E15). In order for the proper signal to be displayed, many parts of the instrument were required to be functioning properly: The scaling and translation of the analog signal, presampling filtering, front panel decoding software, address decoding and latching for the multiplexer and output drive for the display.

The display presented both ECG and VCG waveforms when its inputs were driven by the Waveforma.

The interdependence of the various subsystems of this circuit should have been reduced, both to facilitate testing, and to increase the probability that the device would recover from transient disturbances, such as the relocation of an electrode.

Filtering was tested separately from operation in the circuit, by the conventional means of single sine wave input, varied over frequency, with observation of ratios of amplitude of the input and output waveforms. The filters' responses were as designed (17).

Sampling was tested in two parts. The output of the sample/hold amplifier was tested by observation with the oscilloscope, and compared with its input. The hold step, (step change in output voltage on transition of sample/hold control input) was zeroed, by observing the output on the oscilloscope, with manual switching of the control input. The adjustment is referred to as dynamic zeroing. The decay of the output voltage with time (called droop rate) was observed to be within specification.

Two functions of the A/D converter were tested. Conversion of static analog inputs worked, and the device responded to a request for conversion with an end-of-conversion signal, after approximately 80 microseconds.

The memory functions were demonstrated as follows. Retrieval of data in the PROM was shown by proper execution of the firmware. Storage and retrieval of data in the RAM was demonstrated by proper return from subroutines and interrupt handlers (the associated jump instructions store return vectors on the stack, implemented in RAM). Storage of the digitized ECG was not tested specifically. A self-test mode, or use of in-circuit-emulation would have been helpful to evaluate this aspect of performance, because it would have been possible to halt execution of the program and examine the values stored in the RAM.

The control panel functioned properly, as shown by the proper selection of the multiplexer inputs, correct software response to switch settings, including decoding of the BCD switches, and illumination of the error light. Different length segments of data were returned for different switch settings. The character of the displayed data changed somewhat with different settings due to differing software execution times; however, it was possible to show that segment selection worked as desired.

The display was driven adequately, as shown by the Real Time Mode display of ECG waveforms from the Waveforma.

Despite attention to the subsystems as described above, the waveform displayed in Stored Data Mode was not satisfactory, although the implementation described seemed to display VCG stored patterns in one dimension if the other D/A was disabled. The waveform displayed had the character of a projection of a VCG.

Several factors may have contributed to this. One is timing. Only 1000 samples of the waveform were made, and each of these samples took approximately 80 microseconds to complete. The total sampling time amounted to 240 milliseconds, which would not cover an entire cardiac cycle of a subject with a heart rate of less than 4 Hertz. As no pattern recognition was attempted, there was no synchronization to the QRS complex, and in some cases it may have been missed entirely. This topic is discussed further under Firmware. Another factor is the difference between the frequency content of the input waveform and that of the output waveform. Deglitching (desampling) filters were used on the outputs of the D/A converters. The cutoff frequency was designed for the input waveform, i.e., no consideration was given to the increased rate of output of the waveform from the circuit as compared with the rate of production at the source. A higher cutoff frequency should have been used.

Use of a flash A/D converter would have eliminated any problems occurring due to the sample/hold amplifier, but flash converters cost approximately $100 per chip, compared with less than $15 for the CMOS converter that was used.

It is well-known (47) that each lead must be measured at the same time to produce output suitable for use in diagnosis. This would require three sample/hold circuits if a converter is multiplexed.

Use of a single sample/hold in the instrument provided experience with this component without the cost of three units.

A proper test of input bandwidth would have involved halting of the microprocessor and examination of the memory, perhaps by in-circuit emulation, or use of the Motorola MIKBUG debugging program available in ROM. The selection of inputs worked when the inputs were in the specified range for the device and the power supply to the multiplexer was stable. Display of real-time data was obtained, but the circuit was extremely sensitive to offset voltages such as those obtained from corroded electrodes. The 741 buffer amplifiers had to be selected for sufficiently small output offset bias (after zeroing), because the high gain of the instrumentation amplifiers would produce a signal out of range for the multiplexer inputs. Some means of protecting the multiplexer inputs from out-of-range signals must be developed in order to make debugging of the circuit feasible. A 5-volt Zener diode in parallel with a germanium diode across the input might accomplish this.

The data were gathered at the rate of 1000 samples per lead in approximately 240 milliseconds. The A/D and D/A chips were tested with steady-state inputs, and shown to convert properly. The sample/hold circuit output was examined with the oscilloscope and appeared to hold for 'hold' and follow for 'sample' mode. One test for proper overall conversion would have been the use of a reference input such as a sine wave, and examination of the output signal, probably with a dual-trace oscilloscope.

Segment selection by means of the BCD input switches appeared to work.

The effective bandwidth of the device should have been tested as described for conversion. This test would not have given much information about which part of the circuit was functioning incorrectly, but would have demonstrated proper operation. It would be interesting to determine the amount of time available for real time analysis of the incoming lead data. One technique for measuring this would be the use of a logic analyzer with interval timing features, such as the Hewlett-Packard 1615A.

Debugging would have been simpler with a standard bus between boards, even though this would have involved delivery of signals to boards not requiring them. Use of better wiring technique would have reduced the debugging task significantly, especially attention to color coding for address bus, data bus, power and ground. Use of clips for routing of wires away from corners of IC sockets would have prevented many intermittents.

Firmware

The discussion of the results of firmware development will follow the same outline given in the system description.

The firmware controlling acquisition of the ECG addresses the multiplexer. The presence of the specific pattern on the address bus caused the address latch enable circuit to latch the four lower-order bits of address into the multiplexer, thereby selecting a channel. Selection of input channels covers the firmware required in <u>Real Time Mode</u>.

Sampling was required in <u>Stored Data Mode</u>. Firmware control of sampling was divided into two parts: the initiation of a conversion and the reading of the converted data. Due to the length of time required for a conversion (100 microseconds, maximum, at the 1 megahertz clock rate), the data were read in an interrupt handler subroutine. The conversions were initiated upon return from the interrupt handler. The firmware for requesting a sample is currently followed by a wait loop. It is in this position of the code that any analysis routines would execute. Execution of another sample request currently follows the wait loop, so that the rate of sampling depends upon the rate of conversion. This is an undesirable feature. Furthermore, simple installation of a delay prior to the next conversion request would require a different delay to accommodate each converter. One method of tying the conversion requests to a regular schedule would require a real-time clock. In this scheme, the interrupt handler for the real-time clock interrupt would disable its own interrupt enable, and then enable the interrupt from the end-of-conversion output of the A/D converter, after requesting a conversion. The interrupt handler for the end-of-conversion interrupt would disable its own interrupt, store the retrieved data (converted value), and reenable the real-time clock interrupt.

The code for digitizing and storing in memory the ECG input executed correctly and took very few instructions.

The computer instructions related to reading and responding to the control panel were not optimized for minimum storage requirements, but for maximum modularity. A jump table, as described by Peatman (29) was addressed by the settings of the panel grouped as fields. This technique is very simple to write; it worked immediately.

Use of memory mapping of the peripherals simplified input and output software. All I/O (input/output) instructions are either reads or writes, exactly as used with memory.

The entire program for this device worked, and is presented in the Appendix. It fit, with some space left over, in 1K (1K = 1024) bytes of memory. The output routine, for updating the display in Stored Data Mode, read the BCD switches to determine the beginning and ending points of data to display. The loop for displaying a point was executed a variable number of times, the number of repetitions depending upon the segment length.

Thus, the number of times a selected segment was traced on the screen per unit time (screen refresh rate) depended upon the length of the segment. Shorter segments (fewer than 500 points) were noticeably more pleasant to view. Therefore, more rapidly executing output code should be written.

CONCLUSIONS AND SUGGESTIONS

Despite remaining problems with the current implementation of the microprocessor-controlled vectorcardiograph, it seems feasible to develop a machine capable of producing diagnostic quality VCGs, which would support special purpose routines for specific diseases, or other investigations. References to literature containing diagnostic programs were given earlier in this thesis. One further function of this instrument would be the capability of uploading the data to a larger machine for more sophisticated analysis, or for storage. Motorola provides compatible chips for Direct Memory Access, which would probably be the preferred method of transfer.

Some other improvements include addition of three operational amplifiers to produce leads I, II, and III and the addition of another sawtooth generator to modulate the intensity of the display. The front panel could be improved by the use of soft keys as explained previously. Improvements related to the display might include a linear sweep, a software sweep, adjustment of the recovery time (time between end of one sweep and beginning of the next) especially for Stored Data Mode, interpolation of the data for Stored Data Mode and magnification for smaller segments. A D/A converter with higher precision, would allow more ECG traces across the screen, without reducing horizontal resolution. The use of sample/hold amplifiers on each of the input channels, rather than one shared by the channels could eliminate the difference in timing of measurement of the three leads, if the sample/hold control was a signal to each of the leads' amplifiers. Separate error lights for the different error conditions might clarify improper operation of the device. Some calibration of the display, and a trace of longer persistence may improve the usefulness of the output.

BIBLIOGRAPHY OF ELECTRONICS COMPONENTS LITERATURE

E1. Burr-Brown Model 3660 Low Drift Instrumentation Amplifier and Model 3662 Instrumentation Amplifier. 1975. Burr-Brown Research Corporation, Tucson, Arizona.

E2. Heath Co. Model ET-3400 microprocessor trainer. 1978. Heath Co., Benton Harbor, Michigan.

E3. Hewlett-Packard. 1978. Optoelectronics Designer's Catalog. Hewlett-Packard Components, Palo Alto, California.

E4. Intel Memory Design Handbook. 1977. Intel Corporation, Santa Clara, California.

E5. Intel 2114 1024 x 4 bit Static RAM. 1977. Intel Corporation, Santa Clara, California.

E6. M6800 Microcomputer System Design Data. 1976. Motorola Semiconductor Products, Inc., Phoenix, Arizona.

E7. M6800 Microprocessor Applications Manual. 1975. Microcomputer Applications Engineering. Motorola, Inc., Phoenix, Arizona.

E8. MC6875 Specifications and Applications Information. 1978. Motorola Semiconductors, Phoenix, Arizona.

E9. National Linear Data Book. 1976. National Semiconductor Corp., Santa Clara, California.

E10. National Semiconductor ADC 0816/0817 Single Chip Data Acquisition System. 1977. National Semiconductor Corp., Santa Clara, California.

E11. Signetics Analog Data Manual. 1977. Signetics, Sunnyvale, California.

E12. Signetics Digital Linear MOS. 1972. Signetics, Sunnyvale, California.

E13. Telonic Altair Model 4060 X-Y Monitor. 197x. Telonic Altair Laguna Beach, California.

E14. The TTL Data Book for Design Engineers. Second ed. 1976. Texas Instruments. Inc., Dallas, Texas.

E15. Waveforma Operating Instructions. 1972. General Industries, Inc. Model GI-1017-1. Santa Barbara, California.

BIBLIOGRAPHY

1. Altman, L. and S. Scrupski, eds. 1976. Applying Microprocessors. Electronics Magazine Book Series. McGraw-Hill, New York. 191 pp.

2. Arsenescu, G., M. Sabau, G. Badiu, T. Damsa, A. Voicvlescu, and I. Arsenescu. 1977. Computer Analysis of Orthogonal Electrocardiogram and Vectorcardiogram in Patients with Hypertension. Adv. Cardiol. 19:1973-1975.

3. Bibbero, R. 1977. Microprocessors in Instruments and Control. John Wiley & Sons, New York. 301 pp.

4. Brohet, C., M. Styns, P. Arnaud, and L. Brasseur. 1978. Vectorcardiographic Diagnosis of Right Ventricular Hypertrophy in the Presence of Right Bundle Branch Block in Young Subjects. Am. J. Cardiol. 42(4):602-612.

5. Coughlin, R., and R. Driscoll. 1977. Operational Amplifiers and Linear Integrated Circuits. Prentice Hall, Inc., Englewood Cliffs, N. J. 312 pp.

6. Dudeck, J., and J. Michaelis. 1971. The diagnostic process in electrocardiography. In Chr. Zywietz and B. Schneider, eds. Computer Application on ECG and VCG Analysis. North-Holland Publishing Company, Amsterdam.

7. Foerster, M., Z. Vera, D. A. Janzen, S. J. Foerster, and D. T. Mason. 1977. Evaluation of Precordial Orthogonal Vectorcardiographic Lead ST-segment Magnitude in the Assessment of Myocardial Ischemic Injury. Circulation 55(5):728-732.

8. Frank, E. 1956. An Accurate clinically practical system for spatial vectorcardiography. Circulation. 13:737-749.

9. Friedman, H. 1971. Diagnostic Electrocardiology and Vectorcardiography. McGraw-Hill Book Company, New York. 486 pp-

10. Geselowitz, D. 1971. Some Comments on the Status of Electrocardiographic Lead Systems. In Chr. Zywietz and B. Schneider, eds. Computer Application on ECG and VCG Analysis. North-Holland Publishing Company, Amsterdam.

11. Gelin, J. 1971. Conduction defects with and without myocardial infarction In Chr. Zywietz and B. Schneider, eds. Computer Application on ECG and VCG Analysis. North-Holland Publishing Company, Amsterdam.

12. Greiser, E. and P. Freidricks. 1971. Differences in vectorcardiographic parameters in the Frank lead system caused by systematic dislocation. In Chr. Zywietz and B. Schneider, eds. Computer Application on ECG and VCG Analysis. North-Holland Publishing Company, Amsterdam.

13. Guller, B., F. Lau, R. Dunn, H. Pipberger. 1977. Computer Analysis of Changes in Frank Vectorcardiograms of 666 Normal Infants in the First 72 Hours of Life. J. Electrocardiology 10(1):19-26.

14. Gustafson, D., A. Willsky, J-Y. Wang, M. Lancaster, and J. Triebwasser. 1978. ECG/VCG Rhythm Diagnosis Using Statistical Signal Analysis - I. Identification of Persistent Rhythms. IEEE Transactions on Biomedical Engineering BME 25(4):344-352.

15. Gustafson, D., A. Willsky, J-Y. Wang, M. Lancaster, and J. Triebwasser. 1978. ECG/VCG Rhythm Diagnosis Using Statistical Signal Analysis - II. Identification of Transient Rhythms. IEEE Transactions on Biomedical Engineering BME 25(4): 353-361.

16. Howard, P., A. Benchimol, K. Desser, F. Reich, and C. Graves. 1976. Correlation of Electrocardiogram and Vectorcardiogram with Coronary Occlusion and Myocardial Contraction Abnormality. Am. J. Cardiol. 38(5):582-7.

17. Johnson, D., and J. Hilburn. 1975. Rapid Practical Design of Active Filters. John Wiley and Sons, New York. 264 pp.

18. Johnson, F., and S. Gibbons. 1976. Microprocessors in health care: panacea or more different technology? Biomedical Engineering 11(4):132-136.

19. Jung, W. 1974. IC Op-Amp Cookbook. Howard W. Sams and Co., Inc., Indianapolis, Indiana. 591 pp.

20. Kinoshita, S., K. Suzuki, H. Hiraki and S. Ito. 1977. Direct-writing Color Vectorcardiograph. J. Electrocardiology 10(4):393-395.

21. Korn, G. 1977. Microprocessors and Small Digital Systems for Engineers and Scientists. McGraw-Hill Book Company, New York. 390 pp.

22. Madias, J. 1978. Vectorcardiography and Body Surface Isopentential Mapping to Detect Old Myocardial Infarction (letter). Am. J. Cardiol. 41(2):349.

23. Mason, D., W. Price, J. Wilson, and I. Lauder. 1977. The Use of a Semi-automatic Computerized Procedure for the Study of Genetically Determined Variation in the Vectorcardiogram. Comput. Biol. Med. 7(1):27-33.

24. McNeill, G., N. Poirier, J. Morin, and G. Klassen. 1977. Myocardial Infarction after aortocoronary saphenous vein bypass. Vectorcardiographic study. Br. Heart J. 39(8):903-6.

25. Mehta, J., N. Tuna, J. Moller, and R. Desnick. 1977. Electrocardiographic and vectorcardiographic abnormalities in Fabry's disease. Am. Heart J. 93(6):699-705.

26. Millard, R., B. Hodgkin and C. Nelson. 1978. Effect of ventricular enddiastolic volume on vectorcardiographic potentials of the pig. Am. J. Physiol. 235(2): 182-H 187.

27. Mimbs, J. W., V. deMello, and R. Roberts. 1977. The effect of respiration on normal and abnormal Q waves. Am. Heart J. 94(5):579-582.

28. Nelson, C. V., B. C. Hodgkin, R. A. Bonner, and E. T. Angelakos. 1978. The difference vector: Assessment of effects of changes or interventions. Am. Heart J. 95(2):220-227.

29. Peatman, J. 1977. Microcomputer-based Design. McGraw-Hill, New York. 590 pp.

30. Pipberger, H., R. Dunn, and J. Cornfield. 1971. First and second generation computer programs for diagnostic ECG and VCG classifications, paper presented at 12 Intern. Colloq. Vectorcardiogr. (August, Brussels).

31. Raizada, V., A. Benchimol, and K. B. Desser. 1976. Demonstration of Diffuse Conduction Disturbance in Sick Sinus Syndrome Utilizing Simultaneous His Bundle Electrogram and Timed Vectorcardiogram. Chest 69:416-417.

32. Rautaharu, P. 1977. Toward Standardized VCG Systems. Circulation 55(3):556-557.

33. Ruttkay-Nedecky, I., V. Szathmary, and S. Cagan. 1977. Algorithm for Computer Recognition of Infarction Localizations. Adv. Cardiol. 19:185-186.

34. Sahai, A. 1976. The design of a microprocessor-controlled vectorcardiographic system. Ph.D. Thesis, Iowa State University, Ames, Iowa.

35. Schneider, J., H. Thomas, Jr., and W. Kannel. 1977. Precordial T Wave vectors in the detection of coronary heart disease. The Framingham Study. Am. Heart J. 94(5):578-572.

36. Shakibi, J., I. Aryanpur, M. Paydar, A. Yazdanyar, and B. Siassi. 1976. Simplified Vectorcardiographic Method for Assessment of Pulmonary Arterial Pressure in Children with Chronic Rheumatic Mitral Value Disease. Jpn. Heart J. 17(6):727-30.

37. Stein, P. and A. Simon. 1976. Vectorcardiographic Diagnosis of Diaphragmatic Myocardial Infarction. Am. J. Cardiol. 38(5): 568-75.

38. Strong, P. 1973. Biophysical Measurements. Tektronix, Inc., Beaverton, Oregon, 499 opp.

39. Talbot, S., D. Ilpatrick, A. Jonathan, and M. Raphael. 1977. QRS Voltage of the electrocardiogram and Frank vectorcardiogram in relation to ventricular volume. Br. Heart J. 39(10): 1109-13.

40. Talbot, S., D. Kilpatrick, and B. Weaks. 1978. Vectorcardiographic features of ventricular extrasystoles correlated with conventional scalar electrocardiographic interpretation. Br. Heart J. 40(8):883-890.

41. Toshima, H., Y. Koga, and N. Kimura. 1977. Correlations between electrocardiographic, vectorcardiographic, and echocardiographic findings in patients with left ventricular overload. Am. Heart J. 94(5):547-556.

42. Van Bemmel, J., J. Duisterhout, G. van Herpen, and L Bierwolf. 1971. Pushbutton VCG/ECG processing system. In Chr. Zywietz and B. Schneider, eds. Computer Application on ECG and VCG Analysis. North-Holland Publishing, Amsterdam.

43. Wartak, J. 1970. Simplified Vectorcardiography. J. B. Lippincott Company, Philadelphia, Pennsylvania. 182 pp.

44. Wickline, S., and J. McNamara. 1978. Vectorcardiographic Quantification of Infarct Size in Baboons. Circulation 57(5):910-920.

45. Yokota, M., Y. Ishibe, K. Yamachi, H. Tanimura, and Y. Watanabe. 1977. Sex and Age Differences in Normal P-Loops in the Frank Lead Vectorcardiogram. Jpn. Heart J. 18(1):1-16.

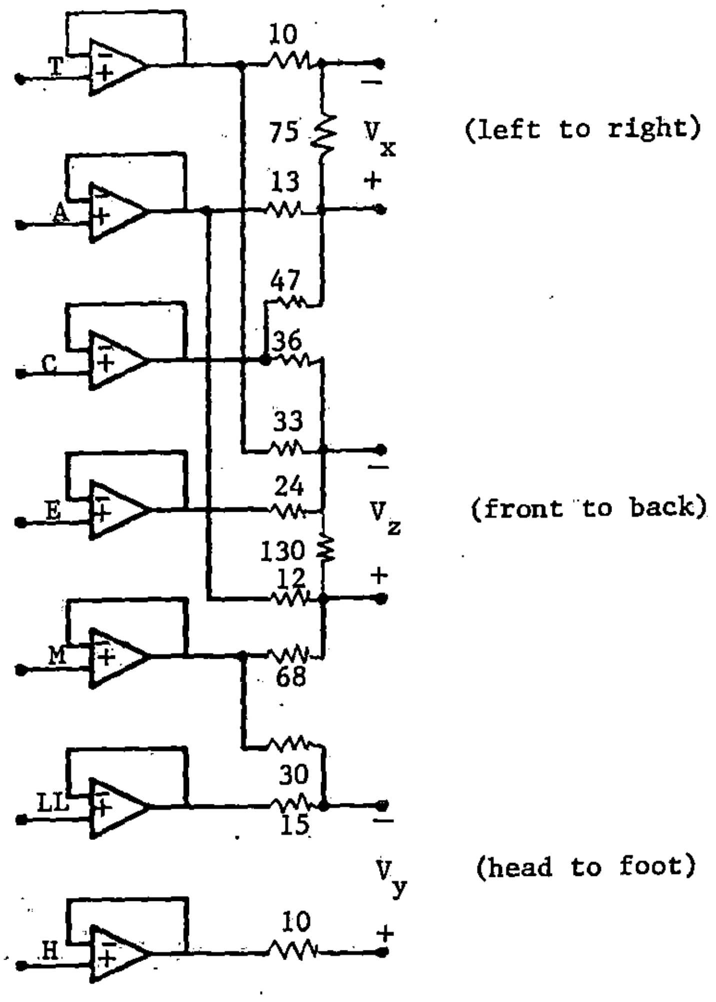
46. Zoneraich, O., S. Zoneraich, J. Rhee, and D. Jordon. 1978. Artial flutter. Electrocardiographic, vectorcardiographic and echocardiographic correlation. Am. Heart J. 96(3):286-294.

47. Zywietz, Chr., and B. Schneider, eds. 1971. Computer Application on ECG and VCG Analysis. North-Holland Publishing Company, Amsterdam, London. 583 pp.

ACKNOWLEDGEMENTS

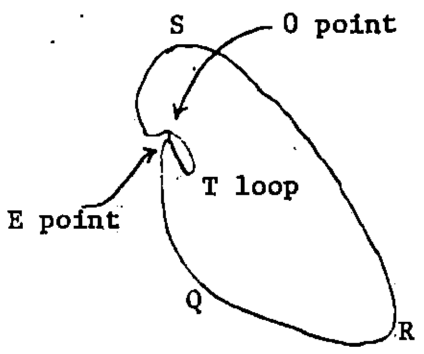
I would like to thank Dr. Curran Swift, Dr. William Brockman, Dr. David Carlson, Dr. Terry Smay, and Dr. Richard Seagrave for their encouragement and support throughout the duration of this project. I would also like to express my gratitude to Alan Luse, Mark Magrane, and Mitchell Chase for their continued interest and moral support.

FIGURES AND ILLUSTRATIONS



All resistor values are in Kohms.

Figure 1. Frank lead network with unity gain buffers on inputs



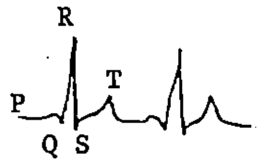
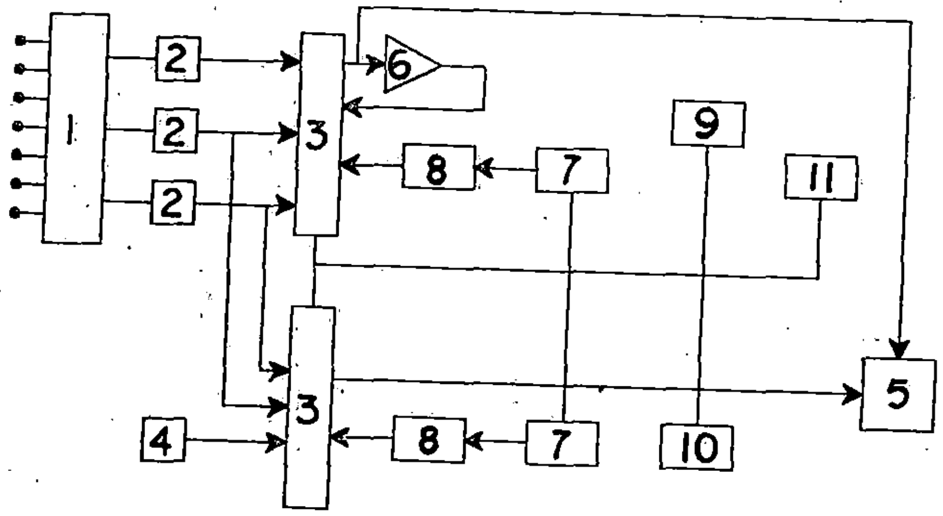
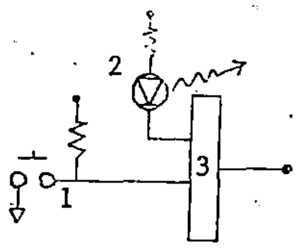


Figure 2. Vectorcardiograph and electrocardiograph waveforms



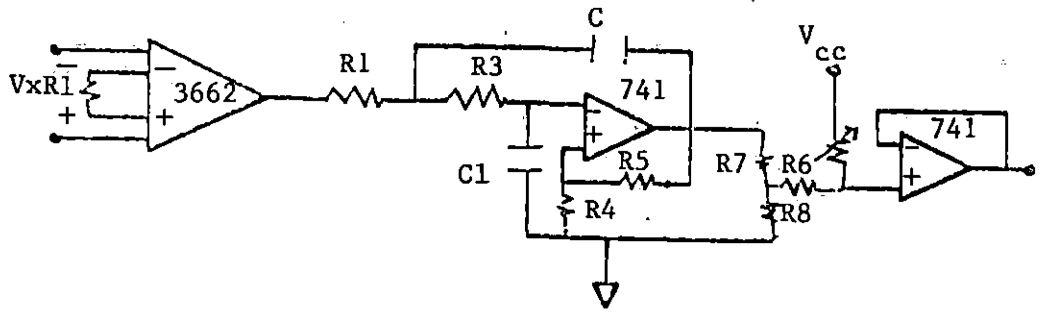
1.	Frank lead network	7.	D/A converter
2.	Instrumentation amplifier and filter	8.	Output filter
3.	16 channel multiplexer and A/D converter	9.	Microprocessor
4.	Sawtooth generator	10.	Memory
5.	Cathode ray tube display	11.	Control panel
6.	Sample/hold amplifier		

Figure 3. System block diagram



1. Switch with pullup resistor
2. Light emitting diode
3. Peripheral interface adapter

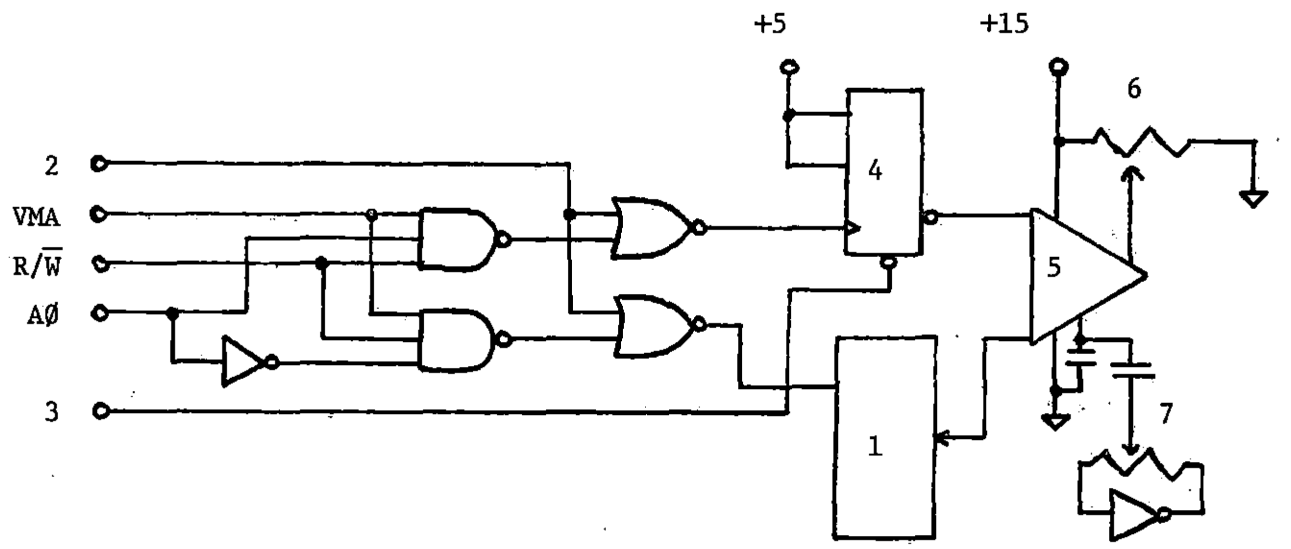
Figure 4. Soft front panel switch



R1	2.2 Kohms
R2	4.7 Kohms
R3	1.2 Kohms
R4	1.2 Kohms

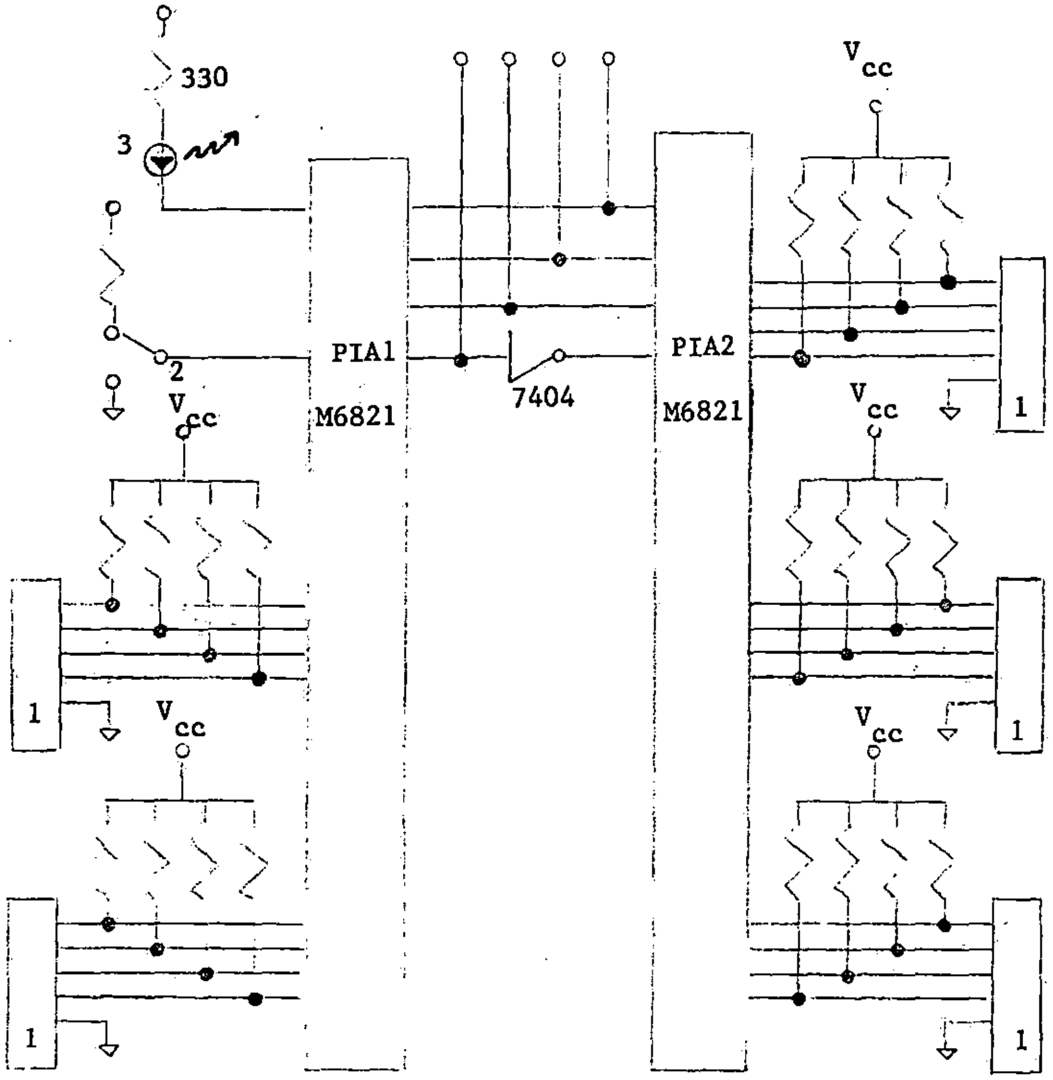
R5	
R6	
R7, R8	1.5 Kohms
C,C1	0.1 uF

Figure 5. Instrumentation amplifier, desampling filter and DC offset adjust



1. A/D IC
2. Start conversion, signal hold mode
3. Return to sampling state
4. Sample-hold mode flip-flop
5. Sample/hold amplifier
6. Output offset bias control
7. Hold step zeroing

Figure 6. Analog-to-Digital conversion



1. BCD switches (6 digits)   2. Toggle switches (5)

3. Error light

Figure 7. Front panel

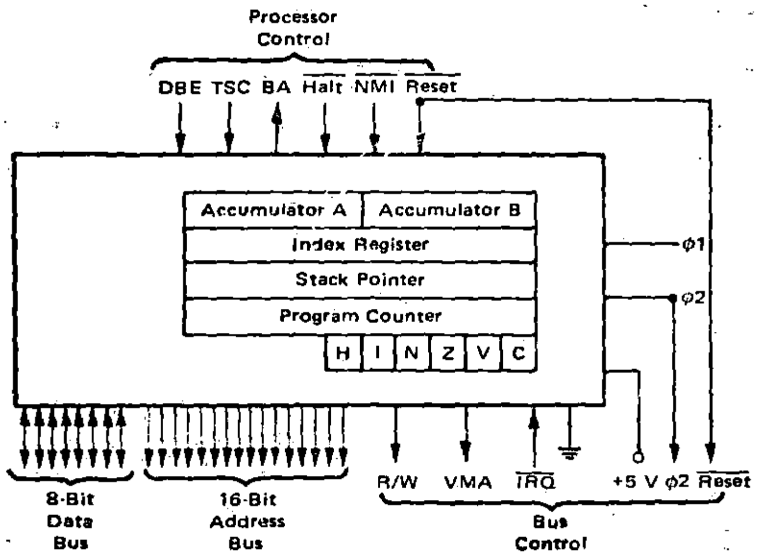


Figure 8. Microprocessor package

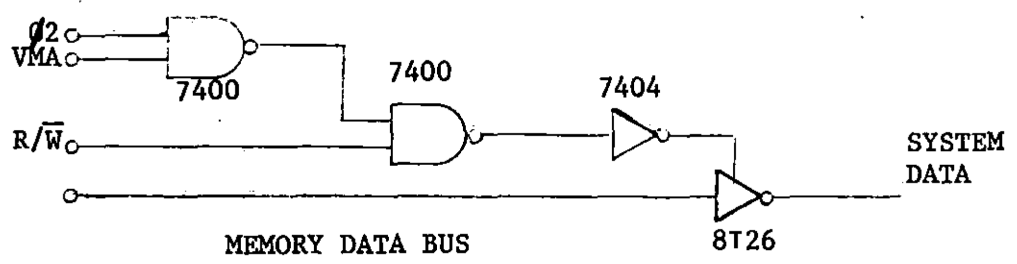


Figure 9. Memory board data bus enable

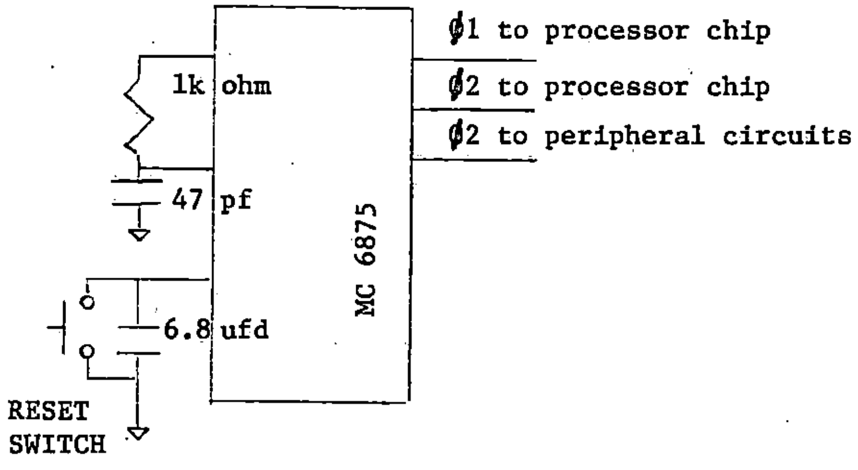


Figure 10. Clock circuit

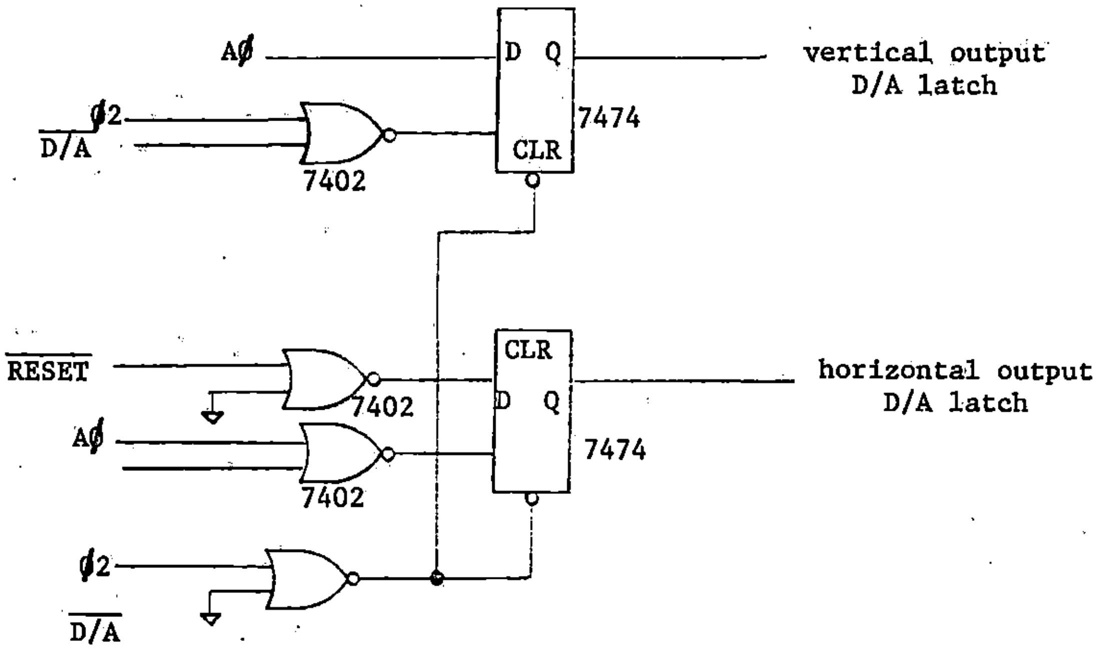


Figure 11. D/A data latch enable

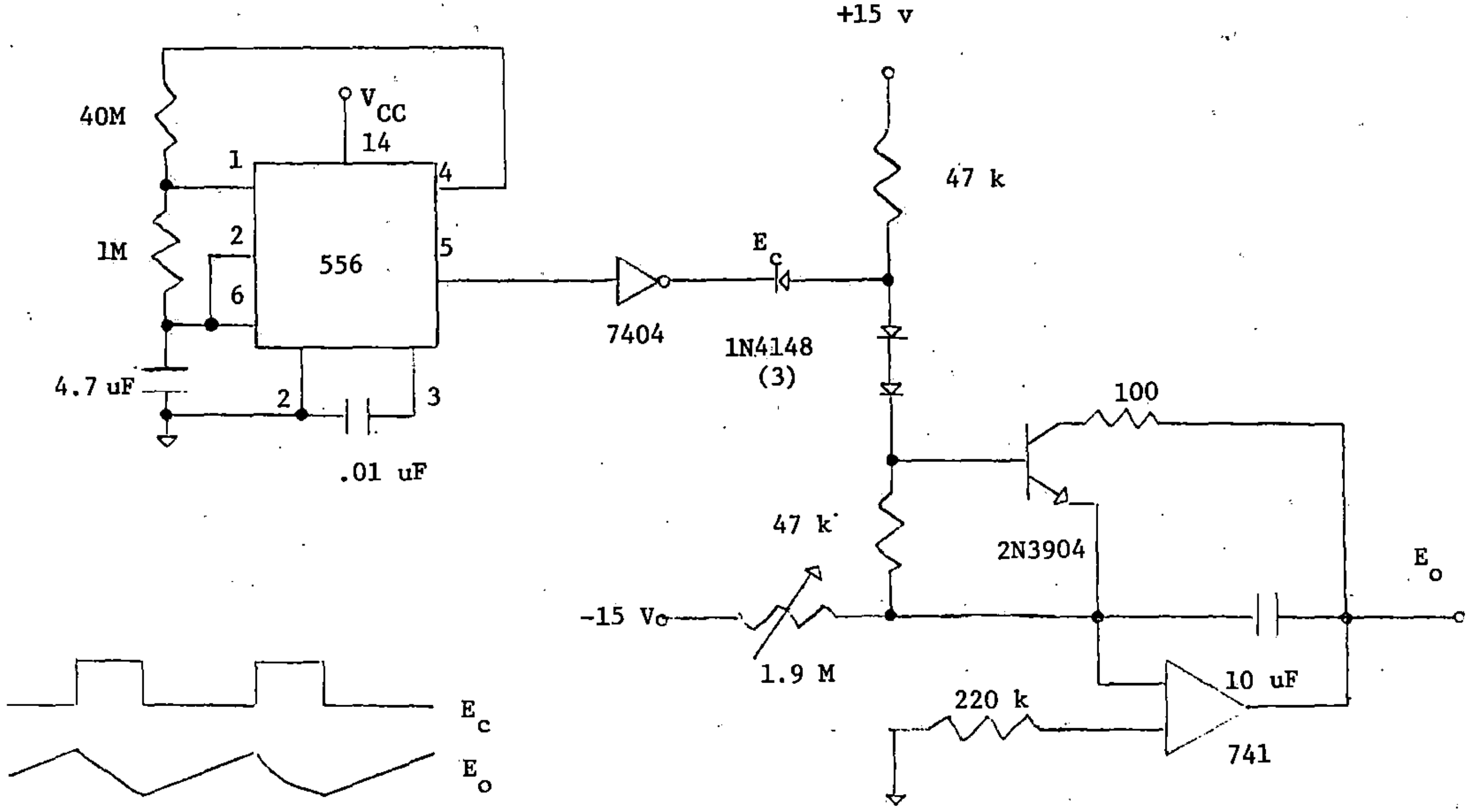
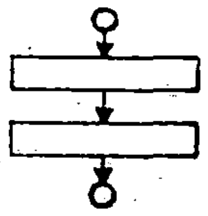
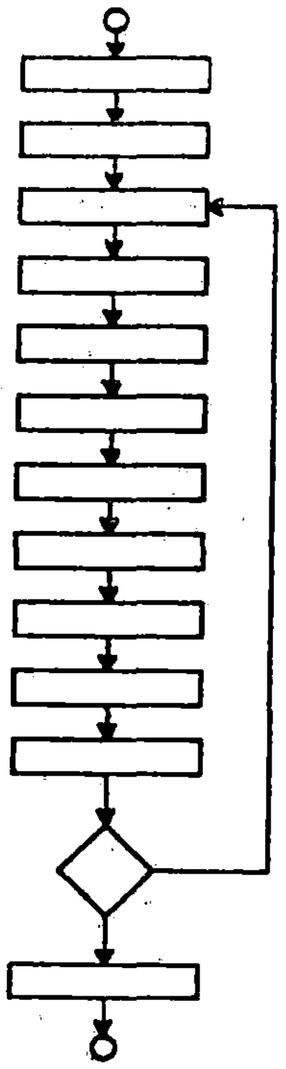


Figure 12. Sawtooth generator

1.



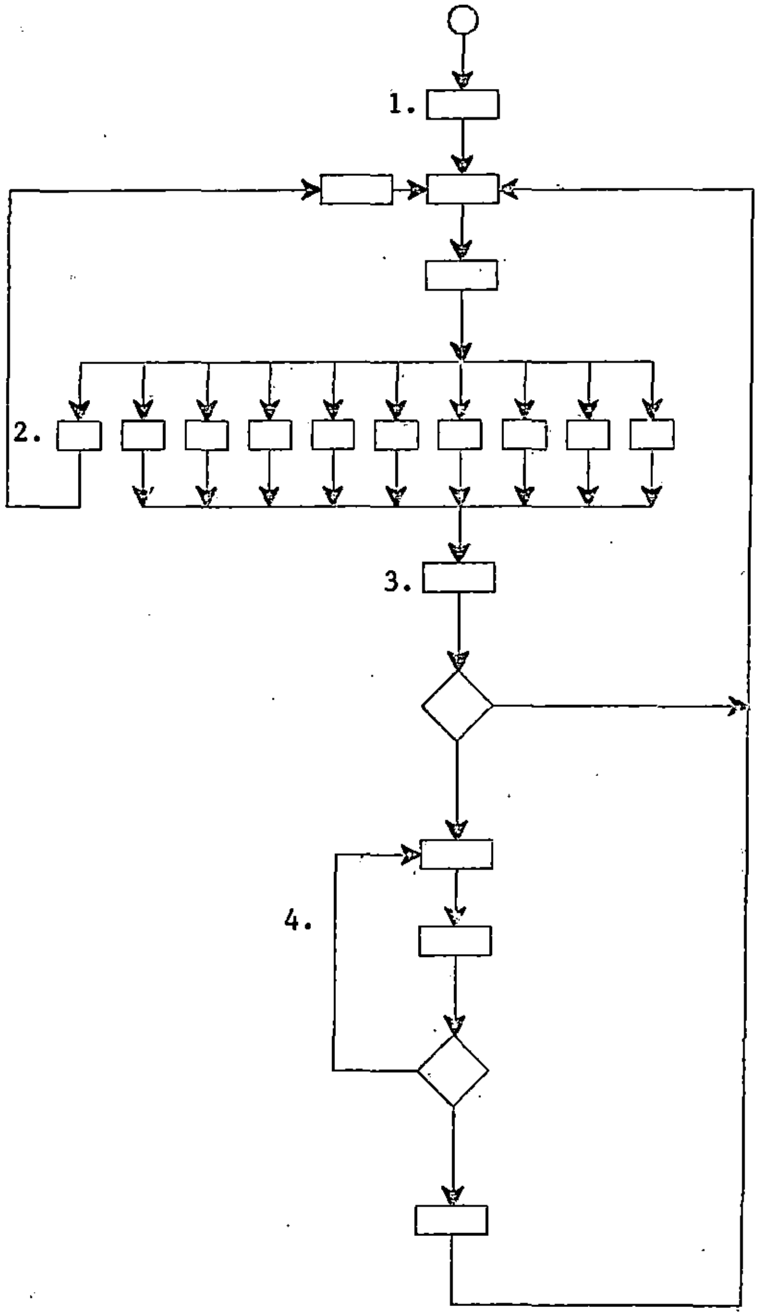
2.



1. Flow chart for instructions in Figure 22.

2. Flow chart for instructions in Figure 21.

Figure 13. Software block diagram I.



1. Instructions of Figure 15b.
2. Instructions of Figure 17.

2. Instructions of Figure 16
4. Instructions of Figures 18, 19 and 20.

Figure 14. Software block diagram II

```
ADDRA    EQU    $4400    PIA A data direction
                         register for A port
BCD34    EQU    $4000    PIA A peripheral port A
ACRA     EQU    $4002    PIA A control register for
                         A side
ADDRB    EQU    $4002    PIA A peripheral port B
BCD56    EQU    $4002    PIA A peripheral port B
ACRB     EQU    $4003    PIA A control register for
                         B side
BDDRA    EQU    $4005    PIA B control register for
                         A side
FPSWI    EQU    $4004    PIA B peripheral port A -
                         front panel
                         switches and error light
BCRA     EQU    $4007    PIA B control register for
                         B side
BDDRB    EQU    $4006    PIA B data direction register
                         for B side
BCD12    EQU    $4006    PIA B peripheral port B
BCRB     EQU    $4007    PIA B control register for
                         B side
```

Figure 15a. PIA register addresses

```
LDA     A       #$00     To address data direction
                         register A
STA     A       ACRA     Store 0 in control register A
LDA     A       #$00     Set data direction register to
                         inputs
STA     A       ADDRA    For peripheral port A
LDA     A       #$04     To address peripheral port A
STA     A       ACRA     Store 4 in control register A
LDA     A       #$00     To address data direction
                         register B
STA     A       ACRB     Store 0 in control register B
LDA     A       #$00     Set data direction to input
STA     A       ADDRB    For peripheral port B
LDA     A       #$04     To address peripheral port B
STA     A       ACRB     Store 4 in control register B
LDA     A       #$0      To address data direction reg
                         A
STA     A       BCRA     Store 0 in control register A
LDA     A       #$01     Set data direction to input
                         except bit 0
STA     A       BDDRA    For peripheral port A
LDA     A       #$07     To address peripheral port,
                         and enable CAI on rising edge
STA     A       BCRA     Store 7 in control register A
LDA     A       #$00     To address data direction
                         register B
STA     A       BCRB     Store 0 in control register B
LDA     A       #$00     To set peripheral port to input
STA     A       BDDRB    Store 0 in data direction
                         register B
LDA     A       #$04     To address peripheral port B
STA     A       BCRB     Store 4 in control register B
```

Figure 15b. Initialization instructions for PIA registers

```
STACK     EQU        $1000
          LDX        #STACK
          TXS                        Set Stack Pointer

ERROR     LDA A FPSWI
          AND A #%1111 1110
          STA A FPSWI
SCAN      LDA A #$70                 Initialize high byte of
                                     subroutine pointer to
          STA A SWITCH               High byte of location of
                                     subroutines
          JSR LITEOF                 Jump to LITEOF

LITEOF    LDA A FPSWI                Load accumulator with
                                     front panel switches
          ORA A #%0000 0001          Set bit 0 high, other bits
                                     unchanged
          STA A FPSWI                Write into peripheral register,
                                     turning LED off
          AND A #%0011 1110          Force bits 7, 6 and 0 too,
                                     keeping toggle status
          ASL A                      Multiply by 2
          STA A SWITCH + 1           Use toggle setting as low byte
                                     of subroutine
          LDX SWITCH                 Place subroutine address in
                                     x register
          RTS

          JSR STRXY                  Handle stored x lead vs y lead
          RTS
```

Figure 16a. Specific response to front panel settings

```
STRXY     LDA A #DAHLO         Place low byte of address of
                               horizontal
                               D/A in accumulator
          STA A HMUX + 1       Store it in low byte of
                               horizontal
                               multiplexer address
          LDA A #DAVLO         Place low byte of address of
                               vertical
                               D/A in accumulator
          STA A VMUX + 1       Store in low byte of vertical
                               multiplexer address
          LDA A #$00           X lead data are the first
                               stored in lead data
          STA A OFST2          Array so its offset = 0
          LDA A #$01           Y lead data are stored 1 byte
                               below X data
          STA A OFST3          So its offset is 1
          RTS
```

Figure 16b. Specific response to front panel settings, continued

```
          JSR  SETMUX         Latch addresses into
                              multiplexers
          LDA  A FPSWI        Bring front-panel switch
                              settings
                              into multiplexer
          ASR  A              Shift real-time/stored bit
          ASR  A              into carry bit
          BCS  SCAN           If real-time is selected,
                              return to scan routine
SETMUX    LDA  A #VHI         Place high order byte of
                              address of vertical multi-
                              plexer in accumulator
          STA  A VMUX         Store in high order byte
                              of channel address
          LDA  A #HHI         Same for horizontal drive
          STA  A HMUX         multiplexer
          LDA  A VMUX         Place channel addresses on
          LDA  A HMUX         address bus, thereby
                              selecting channels
```

Figure 17. General response to front panel settings

```
          JSR GBCDS              Get values of BCD switches
          LDA A START + 2        Place value from switch 3
                                 into accumulator
          PSH A                  Then onto stack
          LDA A START + 1        Same for switch 2
          PSH A
          LDA A START            Same for switch 1
          PSH A
          JSR BCDTHX             Convert the three digits
                                 to 16 bits
          PUL A                  Of binary
          STA A BSTRT + 1        Pull the binary value
          PUL A                  off the stack
          STA A BSTRT            And store

GBCDS     LDA A BCD12            Read the peripheral port
                                 connected to
          COM A                  the first two BCD switches,
          CLR B                  complementing the data
          ASL A                  Clear the B accumulator
          ROL B                  shift the highest order bit
                                 into the
          ASL A                  Carry and then to low end
          ROL B                  Of B accumulator, 4 times
          ASL A                  Which puts switch 1 into the B
          ROL B                  Accumulator
          ASL A
          ROL B
          STA B START            Store the value of switch 1
          LSR A                  Bring the value of switch 2
          LSR A                  Back to low-order half of
          LSR A                  Accumulator A
          LSR A
          STA A START 41         Store value of switch 2

          .
          .                      Do the same for remaining 4
          .                      switches
          RTS
```

Figure 18. Stored data range determination

```
LDA  A  #TABLO          Place low-order byte of address on lead
PSH  A                  data storage table onto stack
LDA  A  #TABHI          Same for high-order byte
PSH  A
LDA  A  BSTRT+1         Place low-order byte of address of
PSH  A                  Begin point within table onto stack
LDA  A  BSTRT           Same for high-order byte
PSH  A
LDA  A  #3              Place number of different interleaved
PSH  A                  Values in table onto stack
LDA  A  OFST2           Place 0-based index of interleaved value
PSH  A                  Within table onto stack
JSR  INTAB              Retrieve location of addressed datum
PUL  A                  Pull high-order byte of address off stack
STA  A  OUTV            Store in high-order byte of address
PUL  A                  Same for low-order byte
STA  A  OUTV+1          Having located data for vertical display
ADD  A  OFST3           Add offset to get location of data for
STA  A  OUTH+1          Horizontal display
LDA  A  OUTV            Propagating carry
ADC  A  #0              if any
STA  A  OUTV

Followed by same for endpoint of range
```

Figure 19a. Determination of absolute address of stored data

```
INTAB     PUL A              Pull high-order byte of return
          STA A RETHI        address from stack and store
          PUL A              Same for low-order byte
          STA A RETLO
          PUL A              Pull index into interleaved
          STA A COL          layers from stack and store
          PUL A              Pull number of layers inter-
          STA A NCOLS        leaved off stack and store
          PUL A              Pull index within layer off
          STA A ROW          stack and store high-order byte
          PUL A              Same for low-order byte
          STA A ROW+1        Pull
          PUL A              starting address of table off
          STA A TEMPI        stack and store high-order byte
          PUL A              same for low-order byte
          STA A TEMPI+1      Load index
          LDX ROW            register with index within layer
          BEQ ADOFST         If it is zero, just add offset
                             to table address

ADNCOL    LDA A TEMPI+1      Take starting address,
          ADD A NCOLS        add number of layers to it
                             as many times as the
          STA A TEMPI+1      index into the individual layer
          LDA A TEMPI
          ADC A #0
          STA A TEMPI
          DEX
          BNE ADNCOL
ADOFST    LDA A COL          Then add the offset
          ADD A TEMPI+1
          PSH A              Push resulting address on stack
          LDA A TEMPI
          ADC A #0           Propagate carry into high-order
          PSH A              byte and push it on stack
          LDA A RETLO        Bring back the return
          PSH A              address onto stack
          LDA A RETHI        and then return
          PSH A
          RTS
```

Figure 19b. Determination of absolute address stored data

```
OUTPUT    LDX OUTV          Load X with the location
                            of data to be converted
          LDA A ZERO,X      Load accumulator with data
                            to be converted
          STA A VERT        Write these data to vertical
                            D/A converter
          LDX OUTH          Same for horizontal output data
          LDA A ZERO,X
          STA A HORIZ
          LDA A OUTHI+1     Load accumulator with low-order
                            byte of
          ADD A #$3         Address of horizontal datum,
                            add 3 to address
          STA A OUTH+1      to get address of next horizon-
          LDA A OUTH        tal datum to be converted
          ADC A #0          Propagate carry
          STA A OUTH        and store
          LDA A OUTV+1      Same for vertical
          ADD A #$3
          STA A OUTV+1
          LDA A OUTV
          ADC A #0
          STA A OUTV
          LDA A LOUTV       Load A, B and X with addresses
          LDA B LOUTV+1     of end of range and current
          LDX OUTV          value within range
          JSR CMP16         Call 16 bit compare
          BGE OUTPUT        If endpoint is equal or higher
          RTS               than current pointer, keep
                            converting
                            else return
```

Figure 20. Output routine

```
NMI      CLI                     Clear interrupt mask
                                 for EOC to interrupt
         LDX #TABLE2             Load X with pointer to ECG data
         STX TABPTR              Store beginning of table
                                 in table pointer
GETX     LDA A XLEADV            Read to A/D makes S/H hold
         STA A XLEADV            Write to A/D initiates
                                 conversion
         LDA A #1                Turn waiting for conversion
         STA A DATAWT            flag on and store
WAIT1    LDA A DATAWT            When waiting flag reset to zero
         BNE WAIT1               Data have been loaded
         LDX TABPTR              Increment pointer to location
         INX                     For next datum
         STX TABPTR


         Same for Y,Z


                   After datum from Z lead stored
         LDX #TABPTR             Load X with current pointer
                                 into data table
         LDA A #TABFHI           Load A, B, with address of end
         LDA B #TABFLO           of table
         JSR CMP16               Compare
         BGE GETX                Until current pointer exceeds
         SEI
         RTI                     Endpoint. Continue to convert
                                 Else return
```

Figure 21. Gathering of ECG lead data

```
*GATHERING VIA NMI
NMI     CLI   [illegible]       LCLEAR INTERRUPT MSK SO EOCS
        LDX       #TABLE2       TABLE POINTER IN X. INIT TO
        STX       TABPTR
GETX    LDA  A    XLEADV       READ TO A/D MAKES S/H HOLD
        STA  A    XLEADV       WRITE TO A/D INITIATES CONVER
*INTERRUPT HANDLER LOADS DATA. RESETS WAIT FLAG
        LDA  A    #1           SET
        STA  A    DATAWT       WAITING FOR DAT AFLAG
WAIT1   LDA  A    DATAWT
        BNE       WAIT1
*DATA MUST HAVE BEEN LOADED TO REACH HERE
        LDX       TABPTR       SO INCREMENT POINTER INTO
        INX                    TABLE
        STX       TABPTR
        LDA  A    YLEADV      SET A/C ADDRESS TO Y LEAD TEL
        STA  A    YLEADV       WRITE TO A.D. TO INITIATE CONV
*INTERRUPT HANDLER LOADS DATA RESETS WAIT FLAG
        LDA  A    #1
        STA  A    DATAWT
```

Figure 22. Storage of converted data

APPENDIX: COMPUTER PROGRAM

(MOTOROLA M68SAM CROSS-ASSEMBLER)

(RELEASE 1.1)

```
                          NAM   VECT
7000                      ORG   $7000
7000  BD  7200            JSR   ERROR
7003  39                  RTS
7004  BD  7200            JSR   ERROR
7007  39                  RTS
7008  BD  7200            JSR   ERROR
700B  39                  RTS
700C  BD  7200            JSR   ERROR
700F  39                  RTS
7010  BD  71EC            JSR   STRXY
7013  39                  RTS
7014  BD  717F            JSR   RELXY
7017  39                  RTS
7018  BD  71BB            JSR   STRDX
701B  39                  RTS
701C  BD  7164            JSR   REALX
701F  39                  RTS
7020  BD  71DB            JSR   STRYZ
7023  39                  RTS
7024  BD  7176            JSR   RELYZ
7027  39                  RTS
7028  BD  71AA            JSR   STRDY
702B  39                  RTS
702C  BD  715B            JSR   REALY
702F  39                  RTS
7030  BD  7200            JSR   ERROR
7033  39                  RTS
7034  BD  7200            JSR   ERROR
7037  39                  RTS
7038  BD  7200            JSR   ERROR
703B  39                  RTS
703C  BD  7200            JSR   ERROR
703F  39                  RTS
7040  BD  71CA            JSR   STRXZ
7043  39                  RTS
7044  BD  716D            JSR   RELXZ
7047  39                  RTS
7048  BD  7199            JSR   STRDZ
704B  39                  RTS
704C  BD  7152            JSR   REALZ
704F  39                  RTS
7050  BD  7200            JSR   ERROR
7053  39                  RTS
7054  BD  7200            JSR   ERROR
7057  39                  RTS
7058  BD  7200            JSR   ERROR
```

```
705B 39                RTS
705C BD  7200          JSR     ERROR
705F 39                RTS
7060 BD  7200          JSR     ERROR
7063 39                RTS
7064 BD  7200          JSR     ERROR
7067 39                RTS
7068 BD  7200          JSR     ERROR
706B 39                RTS
706C BD  7200          JSR     ERROR
706F 39                RTS
7070 BD  7200          JSR     ERROR
7073 39                RTS
7074 BD  7200          JSR     ERROR
7077 39                RTS
7078 BD  7200          JSR     ERROR
707B 39                RTS
707C BD  7200          JSR     ERROR
707F 39                RTS
     4001     ACRA     EQU     $4001
     4000     ADDRA    EQU     $4000
     4002     ADDRB    EQU     $4002
     4003     ACRB     EQU     $4003
     4005     BCRA     EQU     $4005
     4007     BCRB     EQU     $4007
     4004     BDDRA    EQU     $4004
     4006     BDDRB    EQU     $4006
     5003     XLEADV   EQU     $5003
     5006     YLEADV   EQU     $5006
     5009     ZLEADV   EQU     $5009
     6009     RAMP     EQU     $6009
     6003     XLEADH   EQU     $6003
     6006     ZLEADH   EQU     $6006
     600A     YLEADH   EQU     $600A
     0EFF     STACK    EQU     $0EFF
     4004     FPSWI    EQU     $4004
     4006     BCD12    EQU     $4006
     4000     BCD34    EQU     $4000
     4002     BCD56    EQU     $4002
     7000     TABLEI   EQU     $7000
     0000     ZERO     EQU     $0000
     0004     HMUX     EQU     $0004        *HOLDS LO BYTE OF ADDRESS TO
     0006     VMUX     EQU     $0006        *HOLDS LO BYTE OF ADDRESS TO
     3002     DTAH     EQU     $3002        *D/A CONVERTER FOR HIORIZ
     3003     DTAV     EQU     $3003        *D/A CONVERTER FOR VERT
     0008     OFST2    EQU     $0008        *HOLDS OFFSET INTO LEAD DATA
     000C     DAHLO    EQU     $0C          *D/A
     0009     RMPLO    EQU     $09          *LO BYTE OF ADDRESS OF RAMP
     0003     VXLO     EQU     $03          *LO BYTE OF ADDRESS OF X (VER
     0006     VYLO     EQU     $06          SAME Y
     0009     VZLO     EQU     $09          *SAME Z
     0003     HXLO     EQU     $03          *SAME X HORIZ
```

```
           0006      HYLO    EQU    $06
           0008      HZLO    EQU    $08
           000C      DAVLO   EQU    $0C              *D/A
           0050      VHI     EQU    $50
           0060      HHI     EQU    $60
           0009      OFST3   EQU    $0009            *HOLDS OFFSET INTO LEAD DATA
           000A      SWITCH  EQU    $000A
           000C      BSTRT   EQU    $000C
           000F      BSTOP   EQU    $000F
           0010      START   EQU    $0010
           0013      STOP    EQU    $0013
           0100      TABLE2  EQU    $0100
           0000      TABLO   EQU    $00              LO BYTE OF ADDRESS OF LEAD DA
           0001      TABHI   EQU    $01              HI SAME
           0016      RETLO   EQU    $0016            TABLE INTERFACE ROUTINE
           0017      COL     EQU    $0017
           0018      NCOLS   EQU    $0018            HOLDS # COLS
           0019      ROW     EQU    $0019            HOLDS ROW # 2 BYTES
           001B      TABST   EQU    $001B
           001D      TEMP1   EQU    $001D
           001F      LOUTV   EQU    $001F
           0021      TABPTR  EQU    $0021
           0023      OUTV    EQU    $0023
           0025      OUTH    EQU    $0025
           0027      RETH    EQU    $0027
           0028      RETL    EQU    $0028
           0029      BCD3    EQU    $0029
           002A      BCD2    EQU    $002A
           002B      BCD1    EQU    $002B
           002C      BIN1    EQU    $002C
           002D      BIN2    EQU    $002D
           000C      TABFHI  EQU    $0C
           00B7      TABFLO  EQU    $B7
           002E      DATAWT  EQU    $002F
           0030      RETHI   EQU    $0030
                     *INIT
                     *RESET  VECTOR POINTS HERE
7080  86  00         RESET   LDA A  #%00000000   PATTERN FOR SET DATA DIRE
7082  B7  4001               STA A  ACRA      INTO CONTROL REGISTER
7085  86  00                 LDA A  #%00000000   PATTERN FOR ALL INPUTS
7087  B7  4000               STA A  ADDRA     INTO DTAA DIRECTION REGISTER
708A  86  04                 LDA A  #%00000100   PATTERN FOR LOOK AT DATA
708C  B7  4001               STA A  ACRA      INTO CONTROL RESGISTER
708F  86  00                 LDA A  #%00000000
7091  B7  4003               STA A  ACRB
7094  86  00                 LDA A  #%00000000
7096  B7  4002               STA A  ADDRB
7099  86  04                 LDA A  #%00000100
709B  B7  4003               STA A  ACRB
709E  86  00                 LDA A  #%00000000
70A0  B7  4005               STA A  BCRA
70A3  86  01                 LDA A  #%00000001
```

```
70A5 B7 4004          STA A   BDDRA
70A8 86 07            LDA A   #%00000111
70AA B7 4005          STA A   BCRA
70AD 86 00            LDA A   #%00000000
70AF B7 4007          STA A   BCRB
70B2 86 00            LDA A   #%00000000
70B4 B7 4006          STA A   BDDRB
70B7 86 04            LDA A   #%00000100
70B9 B7 4007          STA A   BCRB
70BC CE 0EFF          LDX     #STACK
70BF 35               TXS               SET STACK POINTER
70C0 7F 0000          CLR     ZERO
               *FPSWITCHES  READS FRONT PANEL AND LOOKS IN TABLE FO
               *  DISPLAYING
70C3 86 70     SCAN   LDA A   #$70
70C5 97 0A            STA A   SWITCH    *TABLE HOLDING JMP ADDRESSES
70C7 BD 71F0          JSR     LITEOF    READ TOGGLES. LEAVING SUB ADD
70CA 0A               CLV
70CB AD 00            JSR     ZERO.X
70CD 29 F4            BVS     SCAN
               *CALL APPROPRIATE HANDLER WHICH SETS LO ADDRESS BYTE
70CF BD 7188          JSR     SETMUX    SETS HI BYTE OF MUX ADDRESS P
               *INTO MUXES
70D2 B6 4004          LDA A   FPSWI     GET TOGGLES
70D5 47               ASR A             ROTATE REAL/STORED INTO CARRY
70D6 47               ASR A
70D7 25 EA            BCS     SCAN      IF REAL TIME. GO GET FRONT PA
70D9 BD 7217          JSR     GBCDS     ELSE GO GET START. STOP FROM
70DC 96 12            LDA A   START+2    PUSH BCD SWITCH BYTES ONTO
70DE 36               PSH A             STACK
70DF 96 11            LDA A   START+1    CONVERT START
70E1 36               PSH A
70E2 96 10            LDA A   START
70E4 36               PSH A
70E5 BD 7257          JSR     BCDTHX    BRANCH TO CONVERT ROUTINE
70E8 32               PUL A             PULL BINARY VALUES OFF STACK
70E9 97 0D            STA A   BSTRT+1
70EB 32               PUL A
70EC 97 0C            STA A   BSTRT
70EE 96 15            LDA A   STOP+2    PUSH STOP BCD VALUES ONTO STA
70F0 36               PSH A
70F1 96 14            LDA A   STOP+1
70F3 36               PSH A
70F4 96 13            LDA A   STOP
70F6 36               PSH A
70F7 BD 7257          JSR     BCDTHX    CONVERT STOP VALUE
70FA 32               PUL A
70FB 97 0F            STA A   BSTOP+1
70FD 32               PUL A
70FE 97 0E            STA A   BSTOP
               *16 BIT CONPARE OF BSTOP WITH SBSTR TO CHECK THAT BS
7100 CE 000C          LDX     #BSTRT
```

```
7103 96 0E             LDA A  BSTOP
7105 D6 0F             LDA B  BSTOP+1
7107 BD 72AB           JSR    CMPI6      IF STOP . START THEN
****ERROR 208  ERROR
710A 2D 01             BLT    ERROR      TURN ERROR LIGHT ON AND SCAN
710C 86 00             LDA A  #TABLO     ELSE CONMPUTE ABSOLUTE ADDRES
710E 36                PSH A             FIRST STORED DATUM. CORRESPON
710F 86 01             LDA A  #TABHI     TO START VALUE FROM FRONT PAN
7111 36                PSH A
7112 96 0D             LDA A  BSTRT+1
7114 36                PSH A
7115 96 0C             LDA A  BSTRT
7117 36                PSH A
7118 86 03             LDA A  #3
711A 36                PSH A
711B 96 08             LDA A  OFST2      FOR VERTICAL OUTPUT
711D 36                PSH A
711E BD 72BC           JSR    INTAB
7121 32                PUL A
7122 97 23             STA A  OUTV
7124 32                PUL A
7125 97 24             STA A  OUTV+1
7127 9B 09             ADD A  OFST3      COMPUTE ABSOLUTE ADDRESS
7129 97 26             STA A  OUTH+1     FOR HORIZONTAL OUTPUT
712B 96 23             LDA A  OUTV
712D 89 00             ADC A  #0
712F 97 25             STA A  OUTH
7131 86 00             LDA A  #TABLO     COMPUTE ABSOLUTE ADDRESS OF
7133 36                PSH A             LAST SOTED DATUM
7134 86 01             LDA A  #TABHI     CORRESPONDING TO BCD STOP VAL
7136 36                PSH A             FORM FORNT PANEL
7137 96 0F             LDA A  BSTOP+1
7139 36                PSH A
713A 96 0E             LDA A  BSTOP
713C 36                PSH A
713D 86 03             LDA A  #3
713F 36                PSH A
7140 96 08             LDA A  OFST2
7142 36                PSH A
7143 BD 72BC           JSR    INTAB
7146 32                PUL A
7147 97 1F             STA A  LOUTV
7149 32                PUL A
714A 97 20             STA A  LOUTV+1
714C BD 72E8           JSR    OUTPUT
714F 7F 70C3           JMP    SCAN
                 *SET UP MUXES FOR REAL TIME DATA NLOW BYTEE
7152 86 09       REALZ LDA A  #RMPLO
7154 97 05             STA A  HMUX+1     INTO LO BYTE FOR MUX NHORIZE
7156 86 09             LDA A  #VZLO
7158 97 07             STA A  VMUX+1     FOR VERTICAL MUX
715A 39                RTS
```

```
715B 86 09   REALY  LDA A  #RMPLO    LO BYTE FOR RAMP ADDRESS
715D 97 05          STA A  HMUX+1    INTO LO BYTE FOR HORIZ MUX
715F 86 06          LDA A  #VYLO     LO BYTE OF Y LEAD
7161 97 07          STA A  VMUX+1    FOR VERTICAL MUX
7163 39             RTS
7164 86 09   REALX  LDA A  #RMPLO    LO BYTF OF RAMP ADDRESS
7166 97 05          STA A  HMUX+1    FOR HORIZ MUX
7168 86 03          LDA A  #VXLO     LO BYTE OF X LEAD
716A 97 07          STA A  VMUX+1    FOR VERT MUX
716C 39             RTS
716D 86 08   RELXZ  LDA A  #HZLO     LO BYTE OF Z LEAD
716F 97 05          STA A  HMUX+1    FOR HORIZ MUX
7171 86 03          LDA A  #HXLO     LO BYTE OF X LEAD
7173 97 07          STA A  VMUX+1    FOR VERT MUX
7175 39             RTS
7176 86 08   RELYZ  LDA A  #HZLO     LO BYTE OF Z LEAD
7178 97 05          STA A  HMUX+1    FOR HORIZ MUX
717A 86 06          LDA A  #VYLO     LO BYTE OF Y LEAD
717C 97 07          STA A  VMUX+1    FOR VERT MUX
717E 39             RTS
717F 86 06   RELXY  LDA A  #HYLO     LO BYTE OF Y LEAD
7181 97 05          STA A  HMUX+1    FOR HORIZ MUX
7183 86 03          LDA A  #VXLO     LO BYTE OF X LEAD
7185 97 07          STA A  VMUX+1    FOR VERT MUX
7187 39             RTS
             *SET UP HI BYTE OF MUX ADDRESSES
             *
7188 86 50   SETMUX LDA A  #VHI      *HIGHT BYTE OF ADDRESS OF VER
718A 97 06          STA A  VMUX      STORED IN ADDRESS
718C 86 60          LDA A  #HHI      HI BYTE OF ADDRESS OF HORIZ M
718E 97 04          STA A  HMUX      STORED IN ADDRESS
             *PROCEDURE TO LATCH ADDRESS INTO MUX
7190 DE 06          LDX    VMUX
7192 A7 00          STA A  ZERO.X
7194 DE 04          LDX    HMUX
7196 A7 00          STA A  ZERO.X
7198 39             RTS
             *SET UP MUXES FOR DISPLAY OF STORED DATA NLO BYTEE
7199 86 09   STRDZ  LDA A  #RMPLO    GET LO BYTE OF MUX INPUT TIES
719B 97 05          STA A  HMUX+1
719D 86 0C          LDA A  #DAVLO    GET LO BYTE OF MUX INPUT TIED
719F 97 07          STA A  VMUX+1
71A1 86 02          LDA A  #$2
71A3 97 08          STA A  OFST2     SET OFFSET 2 INTO STORED LEAD
71A5 86 98          LDA A  #$98      MINUS 2
71A7 97 09          STA A  OFST3     SET OFFSET3 INTO SOTRED LEAD
71A9 39             RTS
71AA 86 09   STRDY  LDA A  #RMPLO    GET LO BYTE OF MUX INPUT TIED
71AC 97 05          STA A  HMUX+1
71AE 86 0C          LDA A  #DAVLO    GET LOW BYTE OF MUX INPUT TIE
71B0 97 07          STA A  VMUX+1
71B2 86 01          LDA A  #$1       SET OFFSET2 INTO SOTRED LEAD
```

```
71B4 97 08            STA A  DFST2
71B6 86 99            LDA A  #$99       MU  MINUS I
71B8 97 09            STA A  DFST3      SET OFFSET 3 INTO STORED LEAD
71BA 39               RTS
71BB 86 09     STRDX  LDA A  #RMPLO
71BD 97 05            STA A  HMUX+1
71BF 86 0C            LDA A  #DAVLO
71C1 97 07            STA A  VMUX+1
71C3 86 00            LDA A  #$0
71C5 97 08            STA A  DFST2      SET OFFSET2 INTO STORED DATA
71C7 97 09            STA A  DFST3      SET OFFSET 3 INTO STORED DATA
71C9 39               RTS
71CA 86 0C     STRXZ  LDA A  #DAHLO
71CC 97 05            STA A  HMUX+1
71CE 86 0C            LDA A  #DAVLO
71D0 97 07            STA A  VMUX+1
71D2 86 00            LDA A  #$0
71D4 97 08            STA A  DFST2      SET OFFSET 2 INOT STORED DATA
71D6 86 02            LDA A  #$2
71D8 97 09            STA A  DFST3      SET OFFSET 3 INTO STOERED DAT
71DA 39               RTS
71DB 86 0C     STRYZ  LDA A  #DAHLO
71DD 97 05            STA A  HMUX+1
71DF 86 0C            LDA A  #DAVLO
71E1 97 07            STA A  VMUX+1
71E3 86 01            LDA A  #$1
71E5 97 08            STA A  DFST2      SET OFFSET 2 INTO STORED DATA
71E7 86 01            LDA A  #$1
71E9 97 09            STA A  DFST3      SET OFFSET3 INTO STORED DATA
71EB 39               RTS
71EC 86 0C     STRXY  LDA A  #DAHLO
71EE 97 05            STA A  HMUX+1
71F0 86 0C            LDA A  #DAVLO
71F2 97 07            STA A  VMUX+1
71F4 86 00            LDA A  #$0        SET OFFSET 2 INTO STRED DATA
71F6 97 08            STA A  DFST2
71F8 86 01            LDA A  #$1
71FA 97 09            STA A  DFST3      SET OFFSET 3 INTO STORED DATA
71FC 39               RTS
71FD B6 4004   LITEOF LDA A  FPSWI
7200 8A 01            ORA A  #%00000001
7202 B7 4004          STA A  FPSWI
7205 84 3E            AND A  #%00111110   ONLY TREAT 32 CASES
7207 48               ASL A
7208 97 0B            STA A  SWITCH+1
               *LOCATION OF JMP ADDRESS TABLE IN IT
720A DE 0A            LDX    SWITCH    * X CONTAINS ADDRESS OF SUBRO
720C 39               RTS
720D B6 4004   ERROR  LDA A  FPSWI
7210 84 FE            AND A  #%11111110
7212 B7 4004          STA A  FPSWI
7215 0B               SEV
```

```
7216 39                 RTS
               *GET BCD SWITCHES
7217 B6 4006 GBCDS      LDA A   BCD12      GET 2 DIGITS OF START ADDRESS
721A 43                 COM A
721B 5F                 CLR B              CLEAR B ACCUMULATOR
721C 48                 ASL A              ROTATE DIGIT 2 TO HIGH  NIBBL
721D 59                 ROL B              WHILE DIGIT 1 MOVES INTO LO
721E 48                 ASL A              NIBBLE OF B
721F 59                 ROL B
7220 48                 ASL A
7221 59                 ROL B
7222 48                 ASL A
7223 59                 ROL B
7224 D7 10              STA B   START      STORE HIGHEST DIGIT OF START
7226 44                 LSR A
7227 44                 LSR A              MOVE DIGIT 2 TO LO NIBBLE OF
7228 44                 LSR A
7229 44                 LSR A
722A 97 11              STA A   START+1     STORE 2ND DIGIT OF START
722C B6 4000            LDA A   BCD34      GET 2 DIGITS LAST OF START AN
722F 43                 COM A
7230 5F                 CLR B              HIGHEST OF STOP
7231 48                 ASL A              ROTATE LAST START DIGIT
7232 59                 ROL B              INTO LO NIBBLER OF B
7233 48                 ASL A
7234 59                 ROL B
7235 48                 ASL A
7236 59                 ROL B
7237 48                 ASL A
7238 59                 ROL B
7239 D7 12              STA B   START+2     STORE LAST START DIGIT
723B 44                 LSR A              MOVE FIRST STOP DIGIT TO
723C 44                 LSR A              LO NIBBLE OF A
723D 44                 LSR A
723E 44                 LSR A
723F 97 13              STA A   STOP       STORE FIRST STOP DIGIT
7241 B6 4002            LDA A   BCD56      GET 2 LOWER DIGITS OF STOP
7244 43                 COM A
7245 5F                 CLR B
7246 48                 ASL A              ROTATE MIDDLE DIGIT INTO
7247 59                 ROL B
7248 48                 ASL A              LO NIBBLE OF B
7249 59                 ROL B
724A 48                 ASL A
724B 59                 ROL B
724C 48                 ASL A
724D 59                 ROL B
724E D7 14              STA B   STOP+1     STORE MIDDLE DIGIT
7250 44                 LSR A              ROTATE LO DIGIT OF STOP
7251 44                 LSR A              INTO LO NIBBLE OF A
7252 44                 LSR A
7253 44                 LSR A
```

```
7254 97 15          STA A   STOP+2    STORE LAST DIGIT OF STOP
7256 39             RTS
            *BCD TO HEX     DEPENDS UPON DIGITS STORED IN INDIVIDU
7257 32     BCDTHX  PUL A
7258 97  27         STA A   RETH
725A 32             PUL A
725B 97  28         STA A   RETL
725D 32             PUL A             PULL RETURN ADDRESS OFF STACK
725E 97  29         STA A   BCD3      PULL 3 BYTES OFF STACK
7260 32             PUL A             STORE IN BCD 1 DIGIT PER BYTE
7261 97  2A         STA A   BCD2
7263 32             PUL A
7264 97  2B         STA A   BCD1
7266 7F  002C       CLR     BIN1      STORE O IN
7269 7F  002D       CLR     BIN2      RESULT AREA
726C 96  29  BETA   LDA A   BCD3
726E 27  11         BEQ     ALPHA     IF ZERO GO TO 10%S
7270 4A             DEC A             DECREMENT # OF 100%S
7271 97  29         STA A   BCD3
7273 96  2D         LDA A   BIN2      GET BINARY # SO FAR LO BYTE
7275 8B  64         ADD A   #100      ADD IN 100
7277 97  2D         STA A   BIN2      UPDATE BIN 2
7279 96  2C         LDA A   BIN1      GET BINARY HIGH BYTE
727B 89  00         ADC A   #0        PROPAGATE CZRRY
727D 97  2C         STA A   BIN1      UPDATE BIN HIGH BYTE
727F 20  EB         BRA     BETA      REPEAT ADDING 100%S
7281 96  2A  ALPHA  LDA A   BCD2      GET # OF 10%S LEFT TO ADD IN
7283 27  11         BEQ     GAMMA     IF NOE LEFT PROCEED TO 1%S
7285 4A             DEC A             REDUCE # OF 10%S LEFT TO GO
7286 97  2A         STA A   BCD2      UPDATE # OF 10%S LEFT TO GO
7288 96  2D         LDA A   BIN2      GET LO BYTE OF BINARY #
728A 8B  0A         ADD A   #10       INCREASE IT BY 10
728C 97  2D         STA A   BIN2      UPDATE LO BYTE OF BINARY
728E 96  2C         LDA A   BIN1      GET HIGH BYTE OF BINARY
7290 89  00         ADC A   #0        PROPAGATE CARRY
7292 97  2C         STA A   BIN1      UPDATE HIGH BYTE
7294 20  EB         BRA     ALPHA     REPEATE ADDING 100%S
7296 96  2B  GAMMA  LDA A   BCD1
7298 9B  2D         ADD A   BIN2      ADD TO LO BYTE OF BINARY
729A 97  2D         STA A   BIN2      UPDATE LO BYTE OF BINARY
729C 96  2C         LDA A   BIN1      GET HIGH BYTE OF BINARY
729E 89  00         ADC A   #0        PROPAGATE CARRY
72A0 36             PSH A             PUSH HIGH BYTE ONTO STACK
72A1 96  2D         LDA A   BIN2      GET LO BYTE OF BINARY
72A3 36             PSH A             PUSH LO BYTE ONTO STACK
72A4 96  28         LDA A   RETL      RETURN LO BYTE OF RETURN ADDR
72A6 36             PSH A             TO STACK
72A7 96  27         LDA A   RETH      RETURN HI BYTE OF RETUYRN ADD
72A9 36             PSH A             TO STACK
72AA 39             RTS               RETURN
72AB 36      CMP16  PSH A             COMPARES A||B WITH X
72AC E1  01         CMP B   #1,X
```

```
72AE A2 00           SBC A   #0,X
72B0 26 08           BNE     OUT
72B2 07              TPA
72B3 E1 01           CMP B   #1,X
72B5 27 02           BEQ     EQ
72B7 84 FB           AND A   #%11111011
72B9 06        EQ    TAP
72BA 32        OUT   PUL A
72BB 39              RTS
               *COMPUTES ABSOLUTE ADDRESS OF BYTE IN TABLE
               *GIVEN OFFSET
               *NUMBER OF COLUMNS O FOR VECTORCARDIOGRAPH .= 255
               *ROW NUMBER IE INDEX INTO TABLE 2 BYTES
               *AND ABS ADDRESS OF TABLE BASE HI. LO
               *PULLED OFF STACK IN ORDER LISTED
72BC 32        INTAB PUL A
72BD 97 30           STA A   RETHI      GET RETURN VALUE FROM STACK
72BF 32              PUL A              HIGH THEN LO BYTES
72C0 97 16           STA A   RETLO
72C2 32              PUL A
72C3 97 17           STA A   COL        GET OFFSET IE COL #
72C5 32              PUL A
72C6 97 18           STA A   NCOLS
72C8 32              PUL A
72C9 97 19           STA A   ROW
72CB 32              PUL A
72CC 97 1A           STA A   ROW+1      GET INDEX INTO TABLE NROWE
72CE 32              PUL A
72CF 97 1D           STA A   TEMP1      GET TABLE STARTING LOCATION
72D1 32              PUL A
72D2 97 1E           STA A   TEMP1+1     HI THEN LO BYTES
               *INITIALIZE ABS ADDRESS OF BYTE TO TABLE ADDRESS
72D4 DE 19           LDX     ROW        INITIALIZE COUNTER FOR ROWS
72D6 27 0F           BEQ     ADDFST     IR ROW EQUALS ZERO JUST ADD O
72D8 96 1E     ADNCOL LDA A  TEMP1+1     GET LO BYTE OF ABS ADDRESS I
72DA 9B 18           ADD A   NCOLS      ADD # OF COLUMNS CONE FOR EAC
72DC 97 1E           STA A   TEMP1+1     UPDATE LO BYTE OF ABS ADDRES
72DE 96 1D           LDA A   TEMP1
72E0 89 00           ADC A   #0         PROPAGATE CARRY
72E2 97 1D           STA A   TEMP1      UPDATE HI BYTE OF ABS ADDRESS
72E4 09              DEX                REDUCE ROWS TO GO COUNTER
72E5 26 F1           BNE     ADNCOL     IF MORE ROWS TO GO. LOOP BACK
72E7 96 17     ADDFST LDA A  COL        ADD OFFSET
72E9 9B 1E           ADD A   TEMP1+1     TO LO BYTE OF ABS ADDRESS
72EB 36              PSH A              PUSH ONTO STACK
72EC 96 1D           LDA A   TEMP1      PROGATE CARRY
72EE 89 00           ADC A   #0         PUSH ONTO STACK
72F0 36              PSH A
72F1 96 16           LDA A   RETLO
72F3 36              PSH A              GET BACK RETURN ADDRESS ONTO
72F4 96 30           LDA A   RETHI
72F6 36              PSH A
```

```
72F7 39                RTS
                *SOTART OUTPUT DATA
                *INCREMENT POINTERS INTO TABLE
                *COMPARE POINTER FOR VERTICAL TO OUTV
                *LAST POINTER FOR VERTICAL ENDPOINTE LOUTV
                *RETURN WHEN LAST POINT HAS BEEN REACHED
                *AND DISPLAYED
72F8 DE 23      OUTPUT LDX      OUTV
72FA A6 00             LDA A    ZERO,X     ADDRESS INDEXED THE DATA TO B
72FC B7 3003           STA A    DTAV
72FF DF 25             LDX      OUTH
7301 A6 00             LDA A    ZERO,X     ADDRESS INDEXED THE H DATA TO
7303 B7 3002           STA A    DTAH
                *HORIZ OUTPUT WILL NOT SHOW IF DISPLAYING EKG BECAUS
7306 96 26             LDA A    OUTH+1     INCREMENT HORIZ OUTPUT
7308 8B 03             ADD A    #$3        POINTER BY 3
730A 97 26             STA A    OUTH+1
730C 96 25             LDA A    OUTH
730E 89 00             ADC A    #0
7310 97 25             STA A    OUTH
7312 96 24             LDA A    OUTV+1     INCREMENTR VERT OUTPUT
7314 8B 03             ADD A    #$3        POINTER BY 3
7316 97 24             STA A    OUTV+1
7318 96 23             LDA A    OUTV
731A 89 00             ADC A    #0
731C 97 23             STA A    OUTV
731E 96 1F             LDA A    LOUTV      COMPARE VERT OUTPUT
7320 D6 20             LDA B    LOUTV+1     POINTER WITH ADDRESS OF
7322 CE 0023           LDX      #OUTV
7325 BD 72AB           JSR      CMP16
7328 2C CE             BGE      OUTPUT     IF LAST VALUE >= CURRENT VALU
732A 39                RTS                 KEEP GOING ELSE RETURN
                *GATHERING VIA NMI
732B 0E         NMI -- CLI   LDA A [illegible]   LCLEAR INTERRUPT MSK SO EOCS
732C CE 0100           LDX      #TABLE2     TABLE POINTER IN X. INIT TO
732F DF 21             STX      TABPTR
7331 B6 5003    GETX   LDA A    XLEADV     READ TO A/D MAKES S/H HOLD
7334 B7 5003           STA A    XLEADV     WRITE TO A/D INITIATES CONVER
                *INTERRUPT HANDLER LOADS DATA. RESETS WAIT FLAG
7337 86 01             LDA A    #1         SET
7339 97 2E             STA A    DATAWT     WAITING FOR DAT AFLAG
733B 96 2E      WAIT1  LDA A    DATAWT
733D 26 FC             BNE      WAIT1
                *DATA MUST HAVE BEEN LOADED TO REACH HERE
733F DE 21             LDX      TABPTR     SO INCREMENT POINTER INTO
7341 08                INX                 TABLE
7342 DF 21             STX      TABPTR
7344 B6 5006           LDA A    YLEADV     SET A/C ADDRESS TO Y LEAD TEL
7347 B7 5006           STA A    YLEADV     WRITE TO A.D TO INITIATE CONV
                *INTERRUPT HANDLER LOADS DATA RESETS WAIT FLAG
734A 86 01             LDA A    #1
734C 97 2E             STA A    DATAWT
```

```
734E 96 2E     WAIT2  LDA A  DATAWT
7350 26 FC            BNE    WAIT2
               *DATA MUST HAVE BEEN LOADED T REACH HERE
7352 DE 21            LDX    TABPTR
7354 08               INX
7355 DF 21            STX    TABPTR
7357 B6 5009          LDA A  ZLEADV
735A B7 5009          STA A  ZLEADV
735D 86 01            LDA A  #1
735F 97 2E            STA A  DATAWT
7361 96 2E     WAIT3  LDA A  DATAWT
7363 26 FC            BNE    WAIT3
               *DATA MUST HAVE BEEN LOADED
               *LDX TABPTR  DONT NEED THIS BECAUSE ALREADY IN X
7365 08               INX
7366 DF 21            STX    TABPTR
7368 CE 0021          LDX    #TABPTR
736B 86 0C            LDA A  #TABFHI
736D C6 B7            LDA B  #TABFLO
736F BD 724B          JSR    CMP16
7372 2C BD            BGE    GETX
7374 0F               SEI
7375 3B               RTI
               *TRIGGERED ON POSITIVE GIONG EDGE OF EOC
               *READ A/D VIA IRQ
7376 0F        IRQ    SEI
7377 DE 21            LDX    TABPTR
7379 B6 5003          LDA A  XLEADV    ANY ADDRESS NOT ANSWERED BY 0
737C A7 00            STA A  ZERO,X
737E 7F 002F          CLR    DATAWT
7381 0E               CLI
7382 3B               RTI
73F8                  ORG    $73F8
73F8 7376             FDB    IRQ
73FA 7080             FDB    RESET
73FC 732B             FDB    NMI
73FE 7080             FDB    RESET
                      END
```