AN IMPROVED RESPONSE γ-RAY SPECTROMETER

AN IMPROVED RESPONSE, COMPUTER BASED γ-RAY SPECTROMETER SYSTEM

By

GORDON CHARLES CORMICK, B.ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

December 1976

1977

MASTER OF SCIENCE (1976) (Physics)

McMASTER UNIVERSITY Hamilton, Ontario

TITLE: An Improved Response, Computer Based γ-ray Spectrometer System

AUTHOR: Gordon Charles Cormick, B.Eng. (McMaster University)

SUPERVISOR: Dr. T. J. Kennett

NUMBER OF PAGES: vi, 147

SCOPE AND CONTENTS:

This thesis is intended to document a gamma ray spectrometer that has been under development for several years. Some recent improvements in the background continuum are described here and include rise time discrimination of the Ge(Li) detector pulses and rejection of bremsstrahlung degraded events. The experimental arrangement also allowed a study to be made of the individual background components and the results are presented here. In addition a pulse height analyzer was designed for use with the spectrometer. The design was based around a NOVA mini-computer and its hardware and software features are completely described.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor, Dr. T. J. Kennett, under whose direction and assistance this work was completed. His attitude and insight helped me break down barriers of caution and delve deeply into the many mysteries of electronics, computers and other tools of experimental physics.

I would also like to thank the many faculty members, technicians and fellow students whose help and comradeship made my work easier and my stay more rewarding.

For the typing of this thesis my appreciation goes to Mrs. Dorothy Matthews for a job well done.

McMaster University, The National Research Council and the Government of Ontario deserve a special thanks for providing the financial assistance that made possible this important part of my education.

<u>TABLE OF CONTENTS</u>

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

A Ge(Li) - NaI(Tℓ) triple coincidence pair spectrometer has been in use for several years in the McMaster Nuclear Reactor for the measurement of gamma-ray spectra(1). Some applications of the spectrometer have included thermal and resonance neutron capture gamma-ray studies(2-4), inelastic neutron scattering studies(5) and the investigation of small angle Delbrük scattering. These experiments have relied on the high resolution that is inherent in the germanium detector as well as the relatively simple response function of the pair spectrometer. The response to a monoenergetic photon flux is not the ideal delta function, though, and the distortion introduces a background continuum that is added to the spectra.

This distortion or imperfect response function arises as a result of secondary photon and charged particle interactions within the detector and also because of the finite size of the detector. The spectral response of the Ge(Li) detector has been studied both on the basis of theoretical models(6,7) and from actual experimental data(8,9). For monoenergetic gamma rays the response function is not a delta function but is the sum of several components. By examining the sources of the additional components it may be possible to derive methods of eliminating them.

Ge(Li) pair spectrometers have been in use since 1964[10-12]. They reduce the Compton component of the background by selecting only those events that interact with the detector through pair production. This work reports two further refinements that have been included to reduce the background. Firstly a technique was developed to detect the presence of bremsstrahlung radiation that has escaped from the detector, thereby degrading the pulse height. Such events can be rejected rather than added to the background. A modification of this technique allowed the bremsstrahlung component to be studied by itself. Secondly a method of pulse shape distrimination was added to allow only pulses of short rise time to be collected. Gamma spectrometer systems with this feature have been reported as early as 1967 and have resulted in a background reduction of 67% for a small (2 mm intrinsic depth) single crystal Ge(Li) spectrometer[13] and a background reduction of 30% in larger pair spectrometers[14]. The method used in the present work was easier to implement and resulted in a background reduction of 60% at 8 MeV and 80% at 2 MeV.

The resolution of the spectrometer was good enough that for (n,γ) work it became desirable to acquire the spectra with an analog to digital convertor (ADC) that had an 8192 channel ramp. Since experimental configurations often change, this pulse height analyzer (PHA) needed to be flexible and expandable. These constraints pointed to the need for a computer as the heart of the system so the PHA was implemented using a commercially available 13 bit ADC interfaced to a NOVA 2/4 16 bit mini-computer with 16K words of memory. A keyboard and alpha-numeric display and a graphic display were then added to provide a man-machine interface.

CHAPTER 2

PHOTON INTERACTIONS IN GERMANIUM

The Ge(Li) detector can be viewed as a germanium diode, with a large intrinsic region sandwiched between the p-type and n-type regions, that is back biased to a voltage of about 1.0 to 1.5 KV (figure 2.1). Reverse leakage currents are made negligible by cooling the detector to liquid nitrogen temperatures. When ionizing radiation is formed within the detector, electron-hole pairs are produced and are swept to the junction by the electric field. This current is collected in the preamplifier as a charge whose magnitude depends on the number of charge carriers (electron-hole pairs) that were created. This number is directly proportional to the energy that was deposited in the active region of the detector. Thus the charge collected is a direct measure of the energy of the incident radiation, provided none of the energy escapes from the detector in the total interaction process.

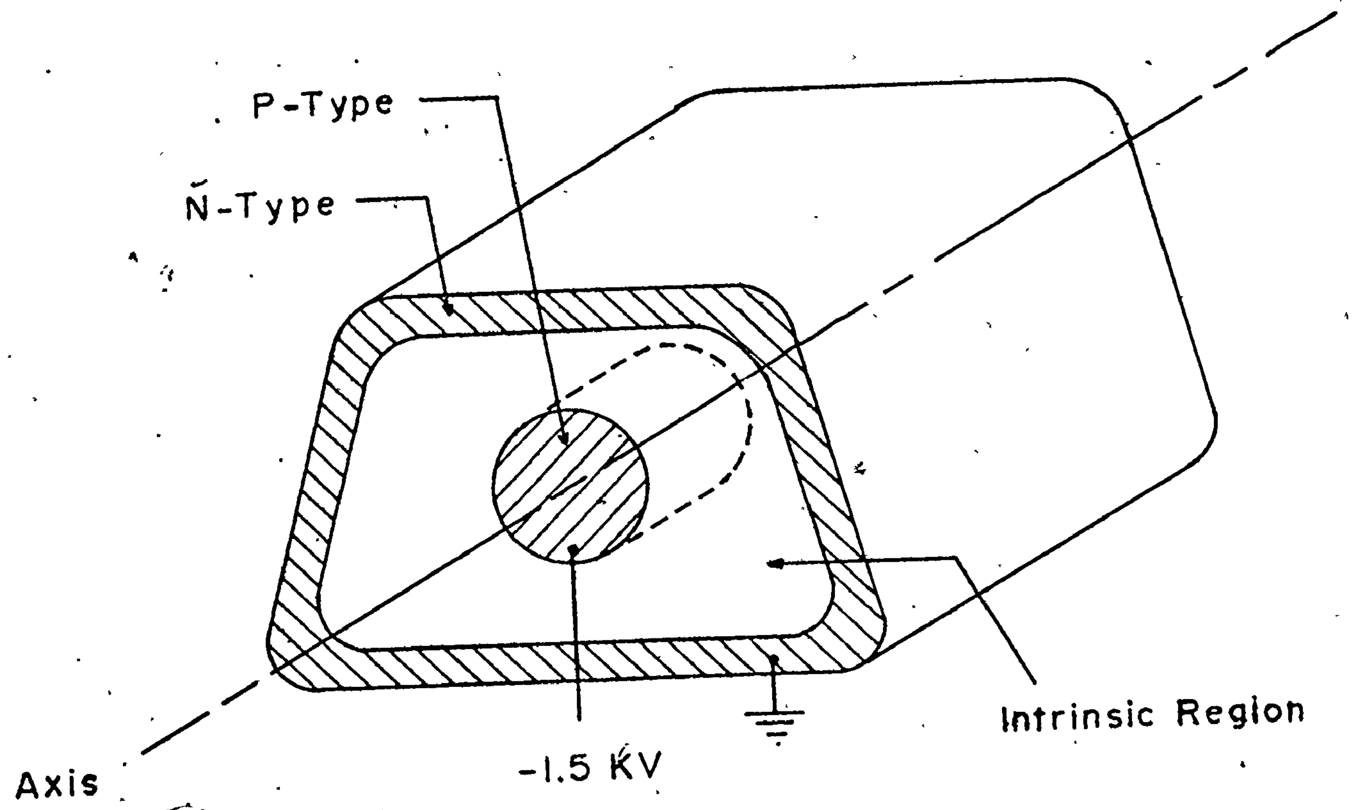


FIG. 2.1[6]
The Ge(Li) Detector Crystal

Photons interact with matter through three major processes. At low energies the photoelectric effect predominates. This occurs when a gamma ray collides inelastically with an atomic electron. The electron becomes disassociated from the atom and acquires an energy equal to the original gamma energy minus the electrons binding energy. The electron contributes to the charge carriers in the germanium and the photon is completely absorbed. The atom then de-excites by emitting one or more X-rays which can escape from the detector but are more likely to be reabsorbed by a second photoelectric interaction. This contributes further electrons to the germanium as charge carriers. The cross section for the photoelectric effect is shown in figure 2.2.

At energies of above 0.1 MeV, the Compton effect becomes important. In this process the photon loses energy by scattering from an unbound electron, transferring part of its energy to that electron. This Compton electron will contribute directly to the charge carriers. The scattered photon can then either undergo further Compton scattering, escape from the detector or be absorbed in a photoelectric process. The probability of escape is fairly

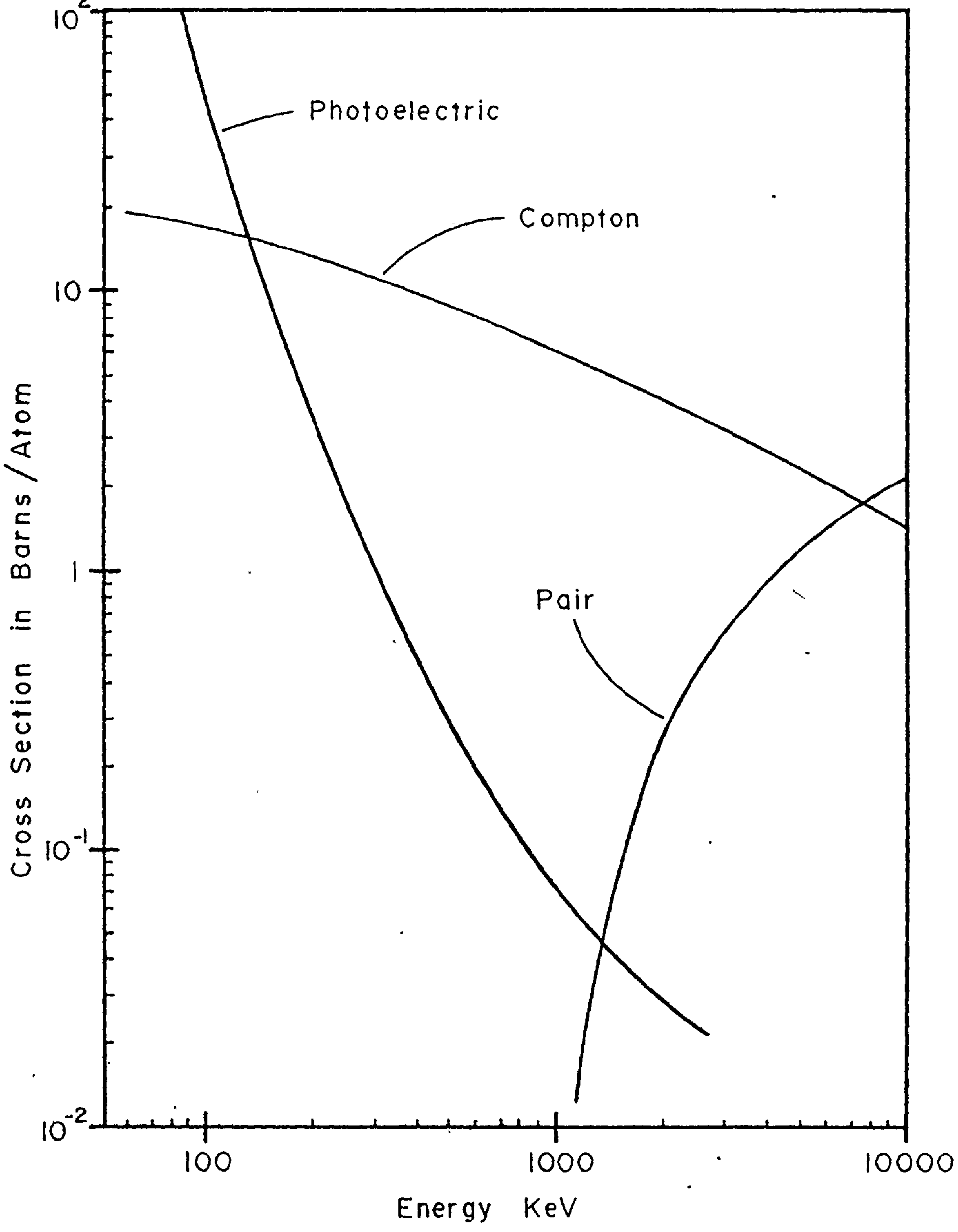


Fig. 2.2
Photon-Germanium Interaction
Cross Sections

high in a small detector, giving rise to the large Compton step below each full gamma energy peak that is typical of gamma-ray spectra.

The third process of interaction is that of pair production. In the presence of the large coulomb field near a nucleus a high energy photon can be completely absorbed and produce an electron-positron pair. The thres-hold energy for pair production is 1.022 MeV, the rest energy of the electron-positron pair. Since the incident gamma completely disappears a kinetic energy of E_γ - 1.022 MeV is left and it is shared by the electron and positron. Energy is transferred to the detector through ionizing colli-sions of the electron and positron. The positron will eventually be annihilated, producing two oppositely directed .511 MeV photons. If both of these photons are reabsorbed the detector output pulse represents the full energy peak. If one or both photons escape from the detector the spectrum will show first and second escape peaks that are displaced .511 and 1.022 MeV below the full energy peak. Complications arise if the electron and positron do not transfer their full energy to the detector but instead slow down by emitting bremsstrahlung radiation which subsequently escapes from the detector. This results in a continuum of degraded events that are added to the spectrum below each peak. Also the .511 MeV photons could lose part of their energy to the detector, producing a toe or high energy step on the first and second escape peaks.

The electronic processing system must reduce the background continuum by selecting only those events that deposit the full photon energy in the active region of the detector. This means distinguishing between the three modes of interaction. The process of pair production is the easiest to positively identify because of the emission of the oppositely directed .511 MeV photons. This process is also likely to deposit most of the gamma energy within the detector due to the short range of beta particles in germanium. This is the principal behind the triple-coincidence pair spectrometer. The only events that are recorded are those which occur with the simultaneous detection of a Ge(Li) pulse and two oppositely directed .511 MeV photons. This selection process automatically rejects gamma rays whose initial energy is less than the threshold level of 1.022 MeV.

For many studies, including neutron capture gamma analysis, this does not exclude the main region of interest so is not critical.

The dominant method of energy loss, hence signal loss, from the detector is through bremsstrahlung radiation that is given off by the electron and positron. Thus any event for which one or more bremsstrahlung photons are detected outside the detector, would obviously be degraded and should be rejected.

Another method for detecting defective pulses is to examine the rise time of the integrated detector signal. This rise time is affected by the region in the Ge(Li) crystal in which the photon interacts. Extensive studies[15-18] have described Ge(Li) detector pulse shapes and their distributions and have shown that photons that interact at or near the dead layer of the detector give rise to pulses of long rise time. These events, since they occur in the dead layer just outside the active region, are more likely to result in part of the original photon energy escaping from the detector, thereby producing a degraded pulse. This can be referred to as the range effect and it can be reduced in the spectrum background by rejecting any events with long rise times. The process of pair production and its accompanying reactions is shown schematically in figure 2.3.

The foregoing sets out the logic requirements of the electronic processing system. The Compton background is reduced by selecting only pair production events. The bremsstrahlung background is reduced by rejecting events if any external photons are detected. Finally defective pulses are rejected on the basis of pulse shape. Events are only collected, then, if the following logic condition is true:

$$\text{(EVENT) AND } \left|\text{(PAIR) AND } (\overline{\text{BREMSSTRAHLUNG}}) \text{ AND (GOOD TIMING)}\right|$$

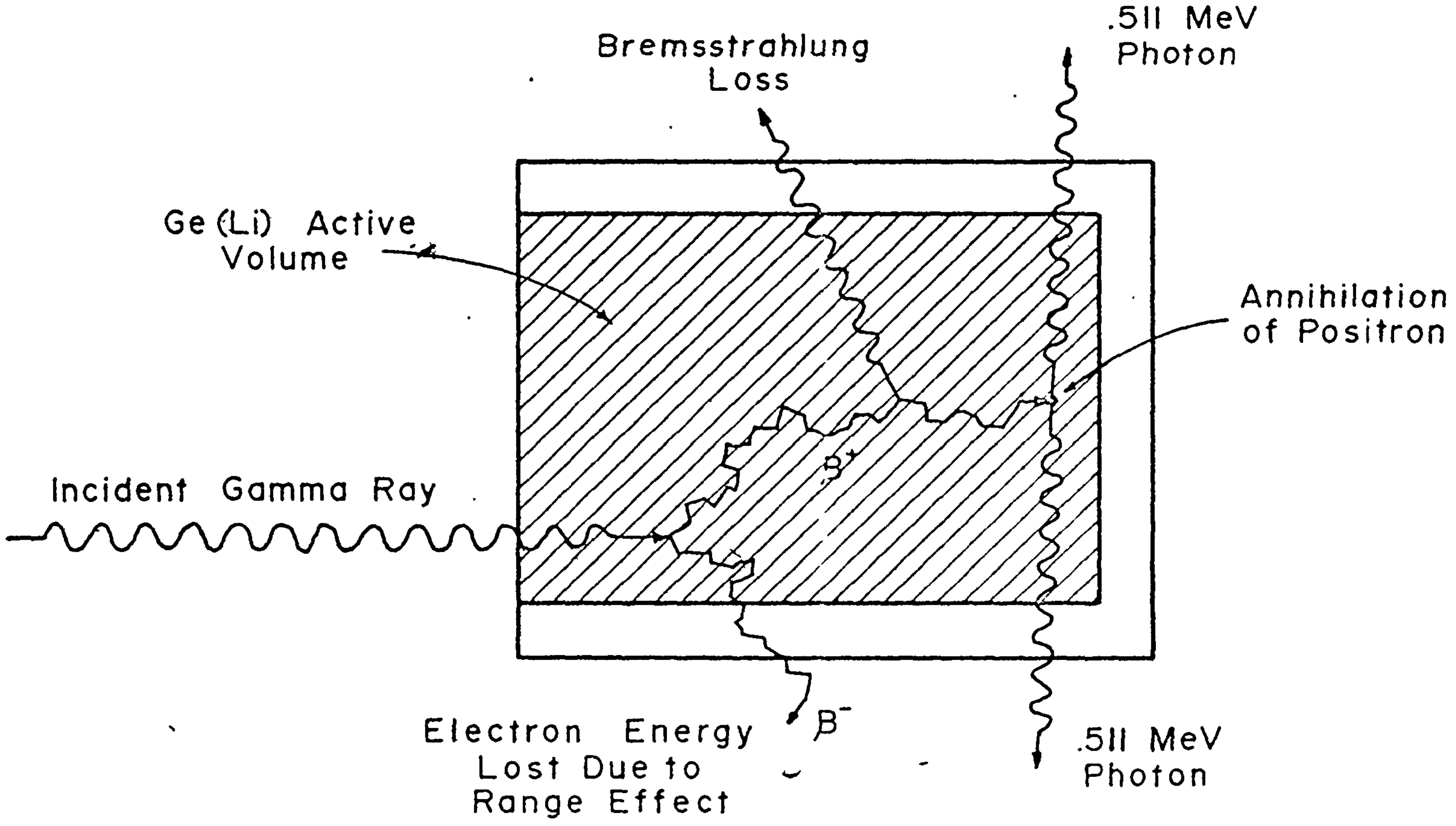


FIG. 2.3

Energy Losses in the Detector

CHAPTER 3

EXPERIMENTAL SETUP

3.1 The McMaster Reactor and the Tangential Through Tube Facility(1).

The McMaster reactor is a light water pool type experimental reactor that uses enriched ^{235}U. The thermal power output is 5 megawatts maximum but was held at 1.5 megawatts throughout most of this research. The thermal neutron flux within the core is approximately 10^{13} neutrons/cm^2-sec, although the flux drops to about 10^{12}n/cm^2-sec at the sample position that was situated in the tangential through tube. The reactor and through tube geometries that are used for (n,γ) work are shown in figure 3.1. In addition to the tangential portion of the through tube, there was a tube leading to the surface of the pool. This allowed on-line sample changing without affecting reactor operation. Samples were placed into graphite carriers (figure 3.2) and lowered into the irradiation position in the tangential tube.

Under neutron irradiation the sample will give off an isotopic field of gamma radiation whose intensity depends on the thermal neutron flux and on the sample size and thermal neutron absorption cross section. The intensity observed at

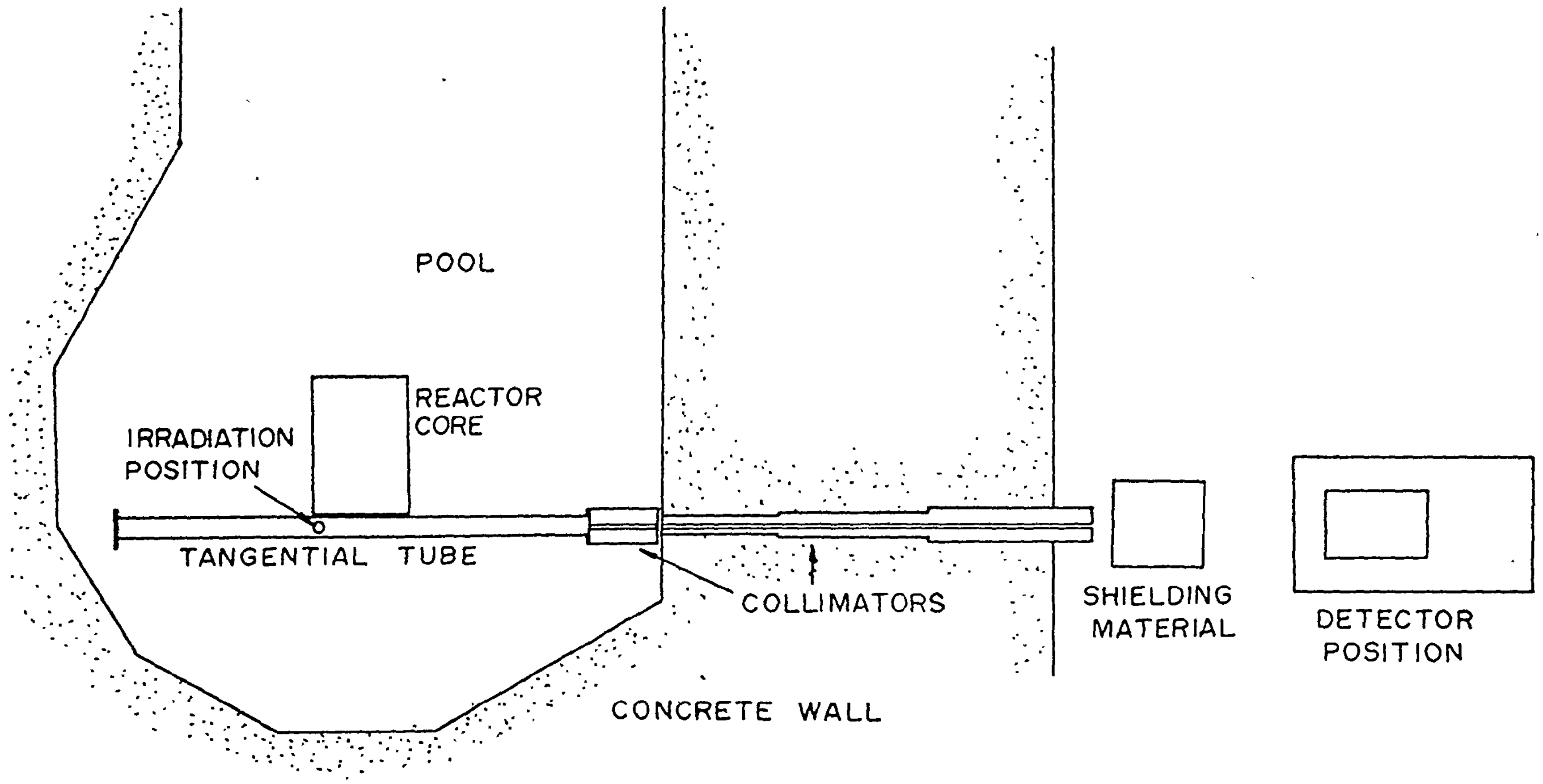


FIG. 3.1
The Reactor and Through Tube Geometries

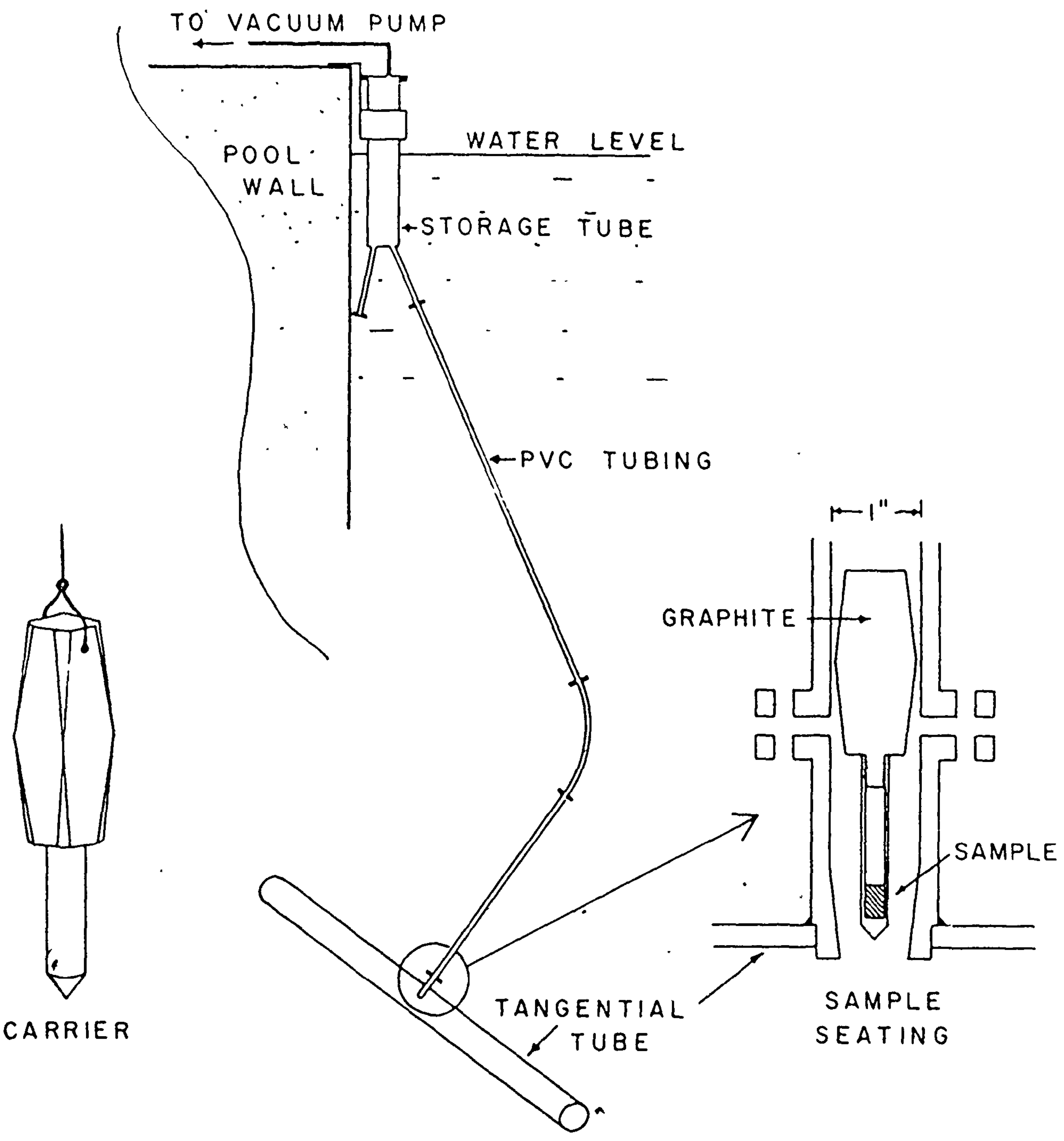


FIG. 3.2

Sample Positioning in the Through Tube

the detector will depend on the solid angle subtended by the collimators. The configuration presently in use gives a well collimated 1 cm diameter beam of gamma rays that impinges on the Ge(Li) detector. As an example, a sample of nitrogen will absorb thermal neutrons in the reaction $^{14}N(n,\gamma)^{15}N$ where the emitted gamma rays will have characteristic energies of 1.89, 5.27, 5.30, 5.53 and 6.32 MeV plus a number of other lower probability energies in the range 244 keV to 10.83 MeV.

3.2 The Ge(Li) - NaI(Tℓ) Triple-Coincidence Pair Spectrometer

The heart of the spectrometer is the germanium detector manufactured by Princeton Gamma-Tech. It is a coaxial lithium drifted germanium detector with a diameter of 3.0 cm and a length of 3.65 cm. The drift depth is 1.1 cm, giving an active volume of 15 cm^3. This crystal is lined up with the 1 cm diameter photon beam approximately 2 meters from the collimators (figure 3.3). Paraffin wax is placed in part of the intervening distance to remove any fast neutrons that could damage the detector. Although the wax does protect the detector it tends to harden the gamma spectra.

Surrounding the germanium detector is an annulus of thallium doped sodium iodide |NaI(Tℓ)|. This annulus is

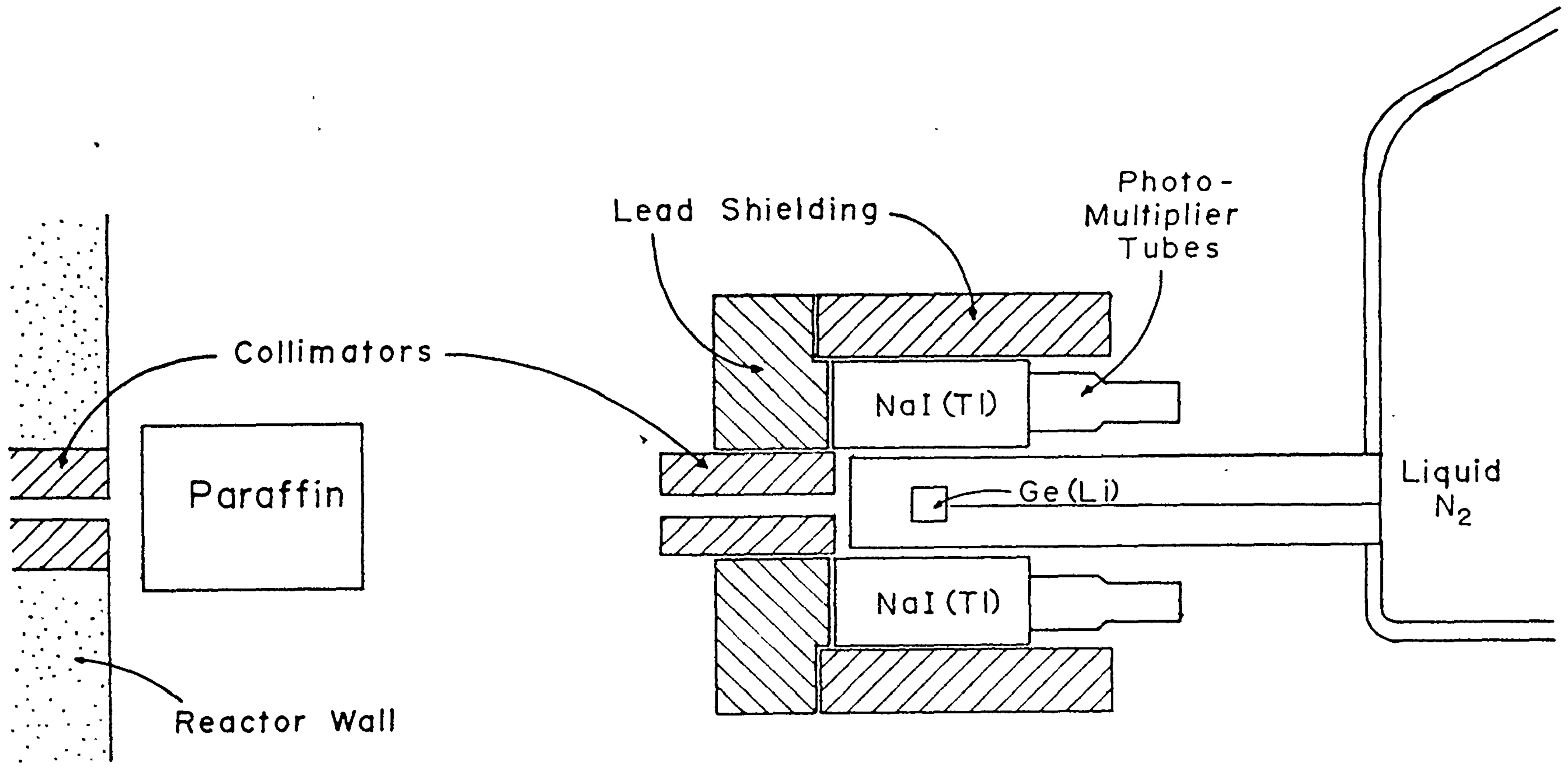


Fig. 3.3

Ge(Li) - NaI (Tl) Geometries

coaxial with the beam and is centered about the Ge(Li) crystal. It is 15.2 cm long and has an inner diameter of 8.6 cm and an outer diameter of 23.0 cm. The annulus is optically divided into quadrants, each with its own photomultiplier tube and associated electronics.

3.3 Spectrometer Logic and Real Time Processing

The complete spectrometer logic is shown in figure 3.4. The logic consists of two sections. One part performs the pair selection and bremsstrahlung rejection based on the NaI(Tℓ) detector signals and the other part does the Ge(Li) detector pulse shape discrimination.

The signals from each of the four quadrants of the NaI(Tℓ) annulus are preamplified and amplified independently before being fed into single channel analyzers (SCA's). Since the objective is to detect oppositely directed .511 MeV photons, the SCA windows are set to allow only that energy to give an output. The window width is primarily determined by the energy resolution of the sodium iodide which is about 40 keV at .511 MeV. The window can be set a little wider than 40 keV and the selected width will be a tradeoff between the background rejection efficiency and the count rate. A pair event, then, is indicated by the simultaneous (within

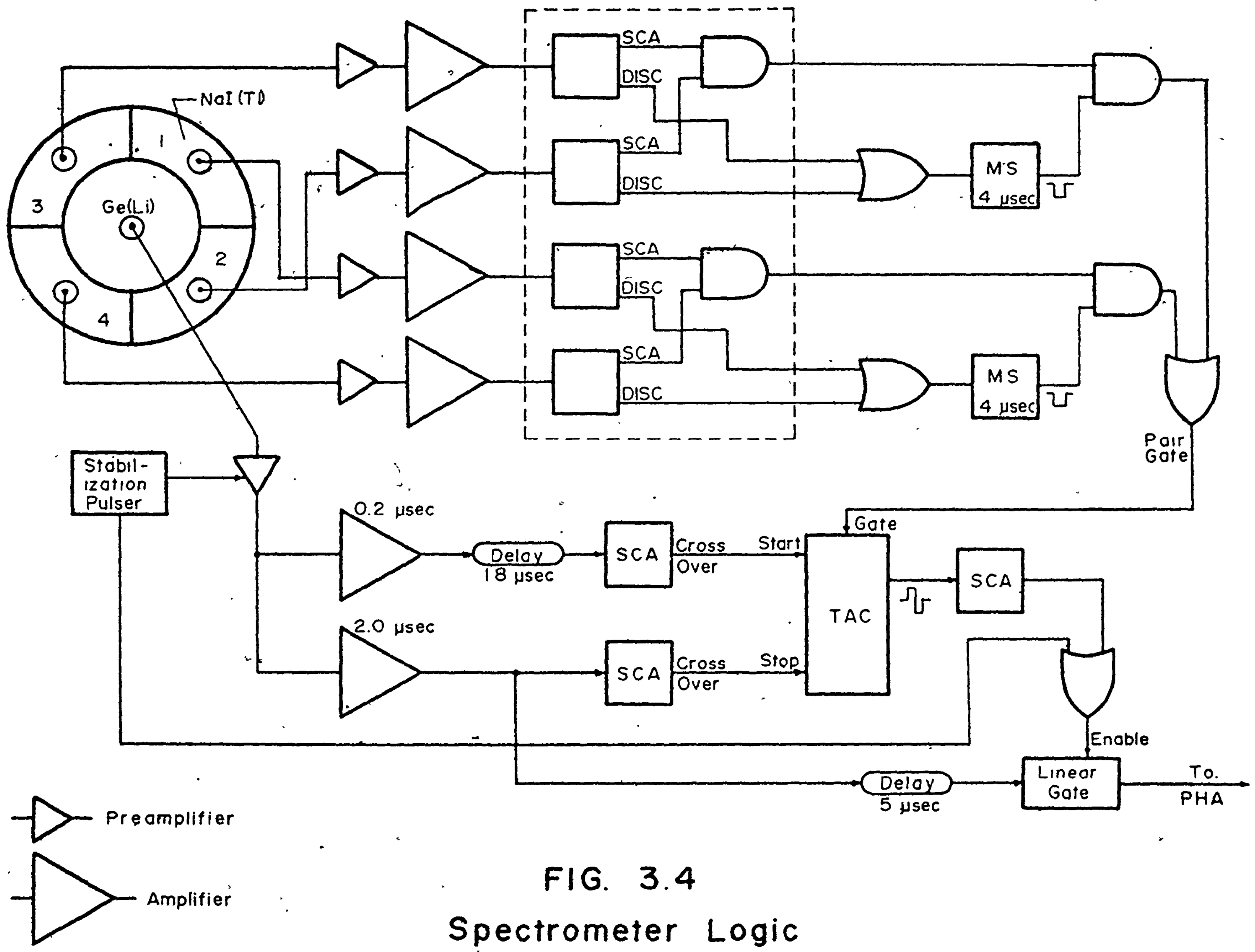


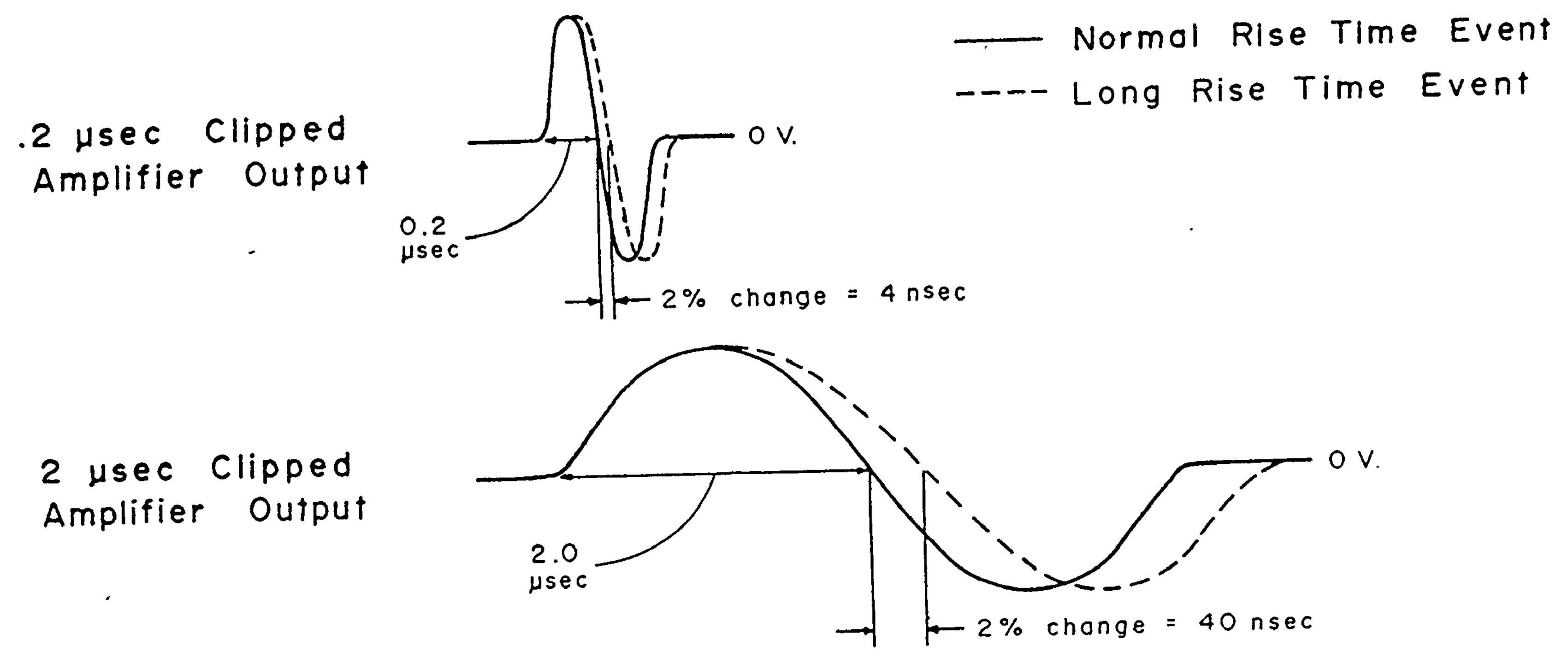
FIG. 3.4
Spectrometer Logic

30 nsec) detection of an SCA signal from quadrants 2 and 3.

The bremsstrahlung photons are also detected with the NaI(Tℓ) annulus. For this, the discriminator outputs of the SCA's are set to indicate the presence of any photons of energy greater than 50 keV. If the .511 MeV pair is detected in quadrants 1 and 4 for example, a discriminator signal from quadrants 2 or 3 will inhibit the pair event signal. Note that if the bremsstrahlung photon had gone into either quadrant 1 or quadrant 4 the event could still have been rejected if the total energy of .511 MeV plus the bremsstrahlung energy was outside the SCA window.

The pulse shape discriminator takes advantage of the fact that the zero crossover of the bipolar output of a spectroscopy amplifier, while stable in time for a fixed pulse shape, does shift according to the rise time of the input. For signals of equal rise time, the crossover is independent of signal height so this system is steady over a wide range of incident photon energies. Two amplifiers are used to treat the Ge(Li) preamplifier signal. One has a short clipping time of 0.2 μsec and the other has a clipping time of 2.0 μsec. If an event comes in and produces a pulse with a long rise time the crossover time of the amplifiers will be extended, with the 2.0 μsec clipped amplifier showing a greater absolute change. An example with a 2% change in crossover time is illustrated in figure 3.5. The zero crossover time is detected in both cases with SCA's and the .2 μsec clipped signal is also delayed in order to eliminate the constant 1.8 μsec difference. The actual difference is measured with a time to amplitude converter (TAC). The pulse height spectrum of the TAC output is equivalent to the rise time probability distribution so a window can be put on the TAC output using an SCA to select only short rise time events.

During this research it was desired to examine the spectral components that were being rejected in order to better understand the detector-photon interactions. This was made possible through the routing facility of the Nuclear Data 3300, 16K pulse height analysis system. The spectra were collected in 4096 channel groups, allowing four spectra to be collected concurrently. The spectrometer logic was then modified to allow the various rejected components to be saved in different groups instead of being simply rejected. This modified logic is shown in figure 3.6 and the results of the experimental runs are given in the next chapter.



Normal Event Time Difference = 2.0 − .2 = 1.8 μsec

Long Rise Time Event Difference = 2.04 − .204 = $\underline{1.836\ \mu\text{sec}}$

.036 usec Change

FIG. 3.5

Pulse Shape Discrimination Timing

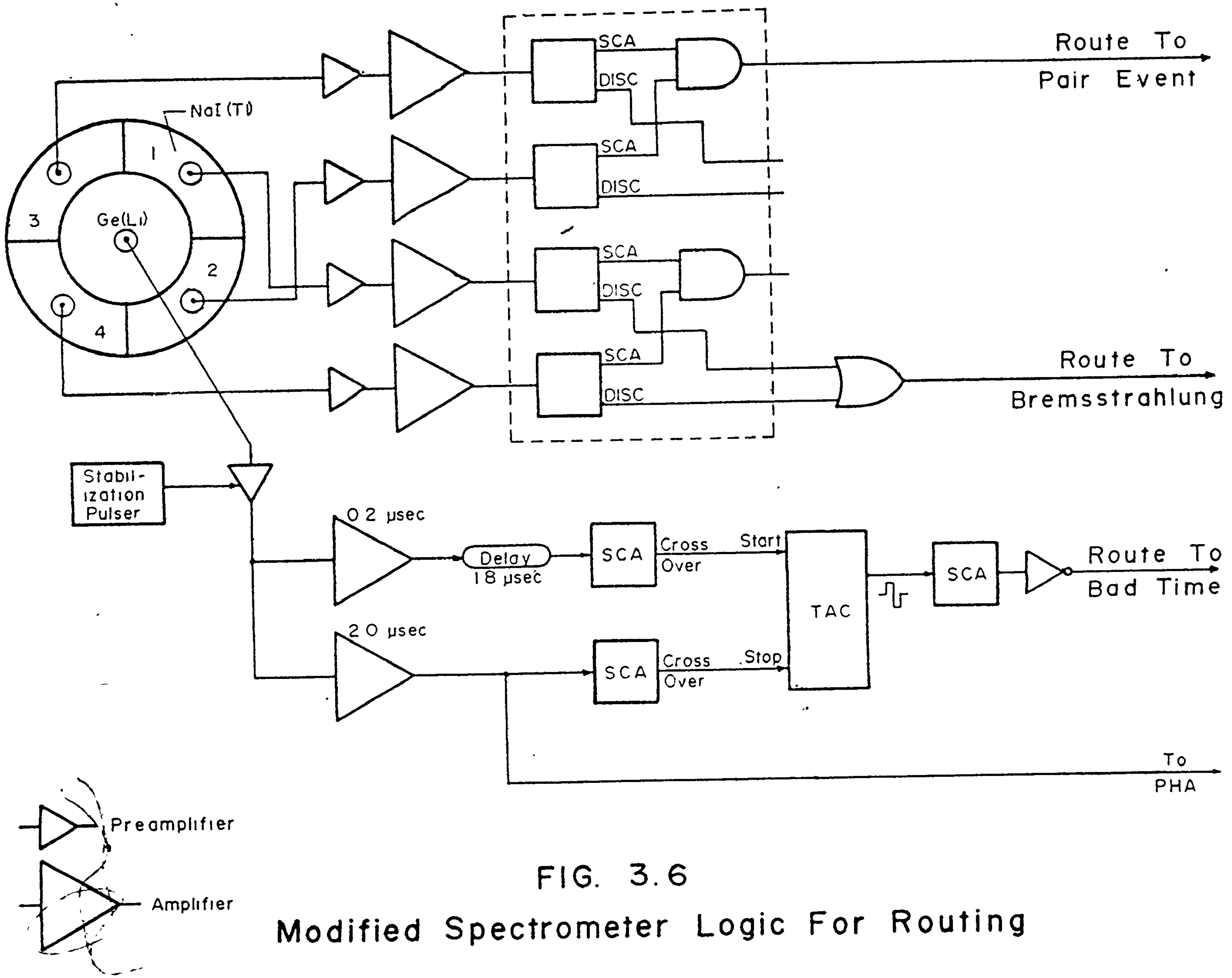


FIG. 3.6
Modified Spectrometer Logic For Routing

CHAPTER 4

γ-RAY SPECTRA AND THEIR COMPONENTS

The response function of the detector is the product of the different interaction processes that were described in chapter 2. For a single crystal (singles) detector the response departs from the ideal delta-function through the addition of components as shown in figure 4.1. The background low energy tails from each peak are additive, producing a relatively flat continuum in the low energy region of the spectrum. This background can swamp any low intensity peaks by reducing the signal to noise ratio (i.e. peak area to background ratio) especially in the low energy region where the density of peaks is likely to be the highest.

For a detector operated as a pair spectrometer the Compton background is eliminated and the full energy and first escape peaks removed, making it easier to see the remaining components of the background. Figure 4.2 shows the shapes of these components as they are added to a single peak spectrum.

Thermal neutron capture gamma-ray spectra were collected for various elements using the ND-3300 analyzer on a 4096 channel ramp.

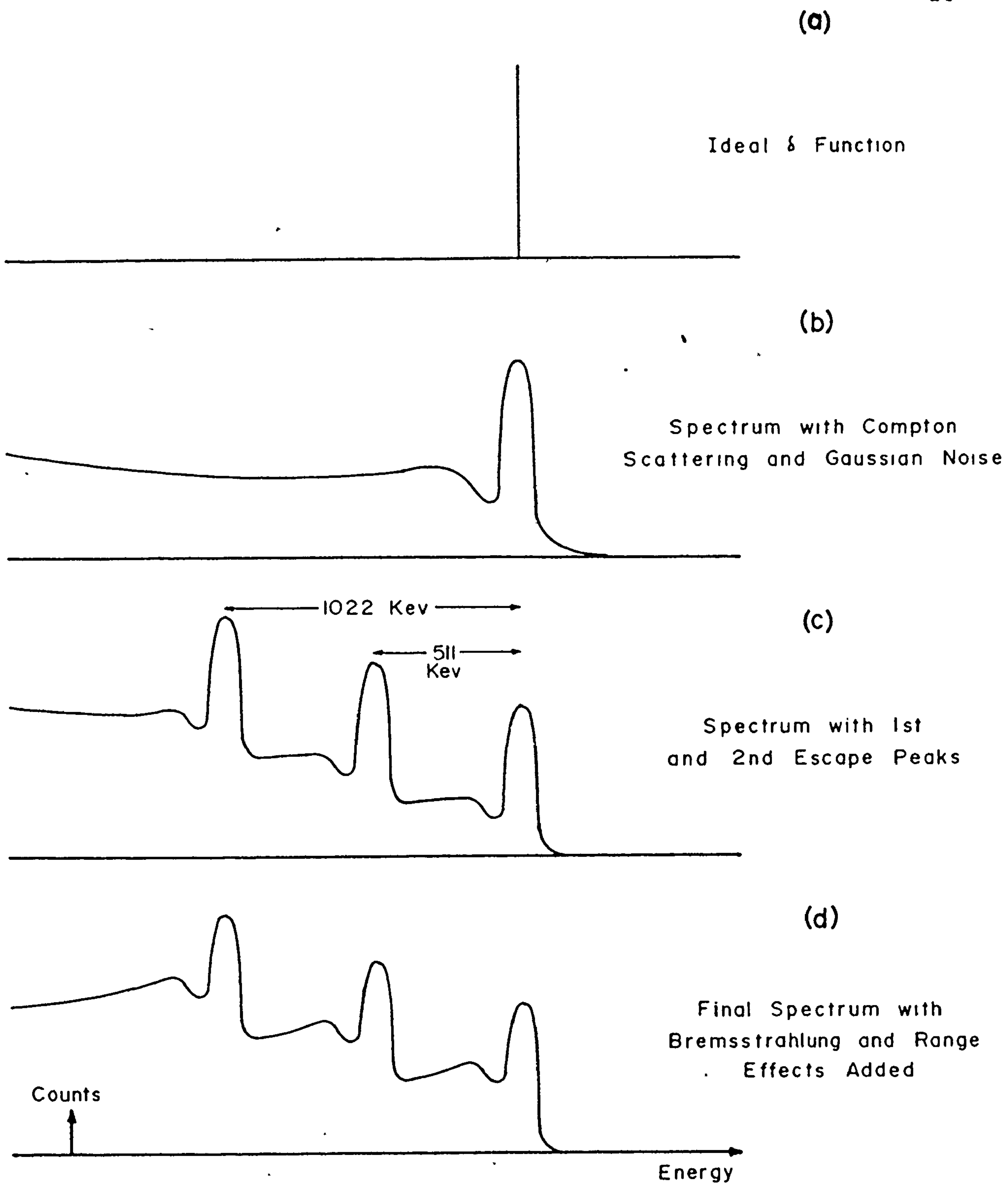

(a)
Ideal δ Function
(b)
Spectrum with Compton Scattering and Gaussian Noise
1022 Kev
511 Kev
(c)
Spectrum with 1st and 2nd Escape Peaks
(d)
Final Spectrum with Bremsstrahlung and Range Effects Added
Counts
Energy

FIG. 4.1

Theoretical Response Function for a Monoenergetic Gamma in a Single Crystal Detector

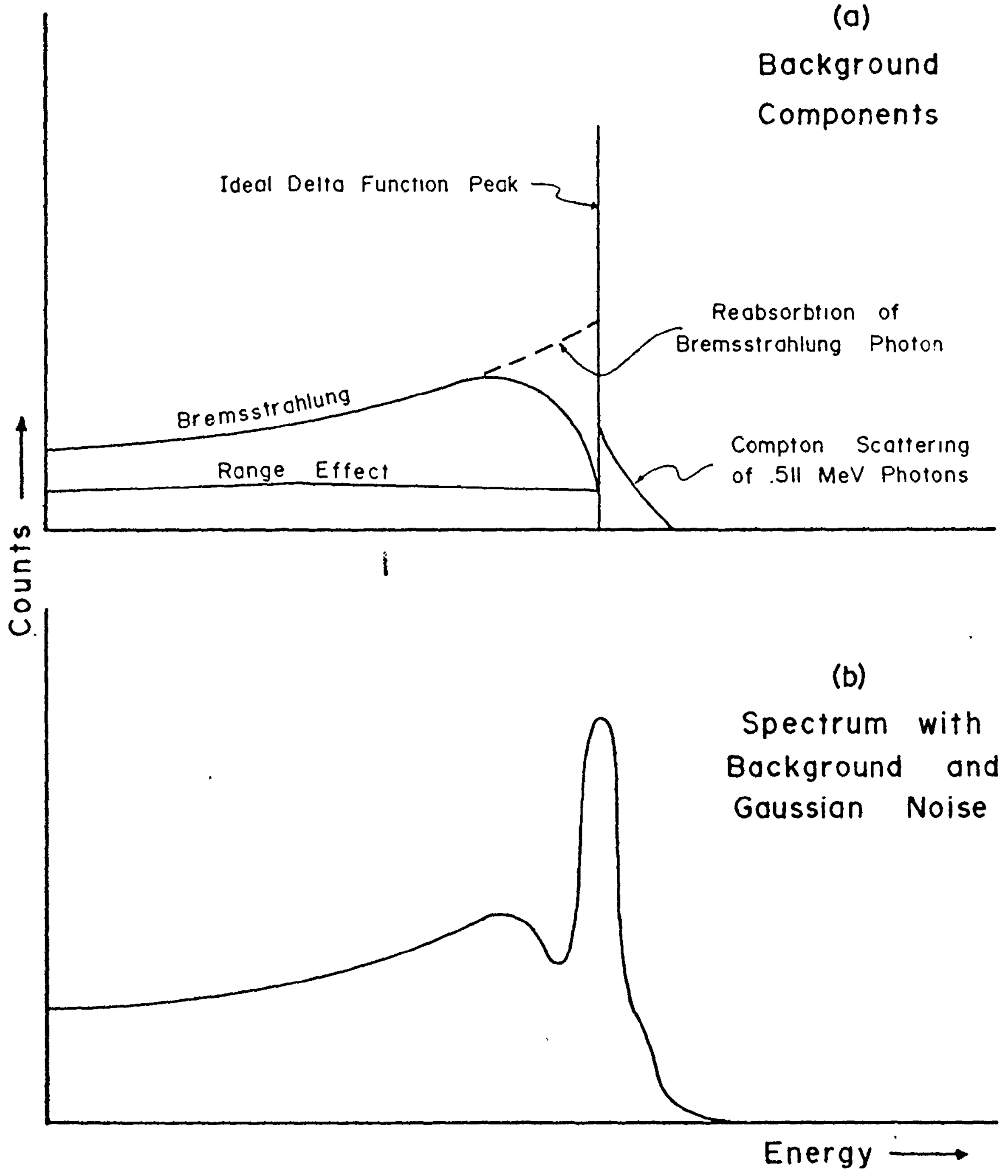


FIG. 4.2

Theoretical Response Function for a Monoenergetic Gamma In a Pair Spectrometer

An example of a singles spectrum is shown in figure 4.3(a). The target used here was iron and the principal reaction was $^{56}Fe(n,\gamma)^{57}Fe$. At the high energy end the doublet can easily be seen as the full energy peaks and as the first and second escape peaks. Few peaks can be detected above the background ~~at~~ energies below 5 MeV. With the detector operated as a pair spectrometer we get the spectrum of figure 4.3(b). Only the second escape peaks remain and the reduction in low energy background allows one to see a number of previously undetectable peaks. When compared to the singles spectrum the pair spectrum has a signal to noise ratio that is better by a factor of 4.1 at 3 MeV and decreases linearily to a factor of 1.8 at 8 MeV. These improvement factors are spectral dependent, however, and may be different for targets other than iron.

Time discrimination, when applied to both singles and pair spectra, was seen to provide a significant improvement in the signal to noise ratio. The experimental setup lent itself very well to the study of the rise time distributions. The output of the TAC (see figures 3.4 and 3.6) is a distribution of pulse heights corresponding to Ge(Li) detector rise times.

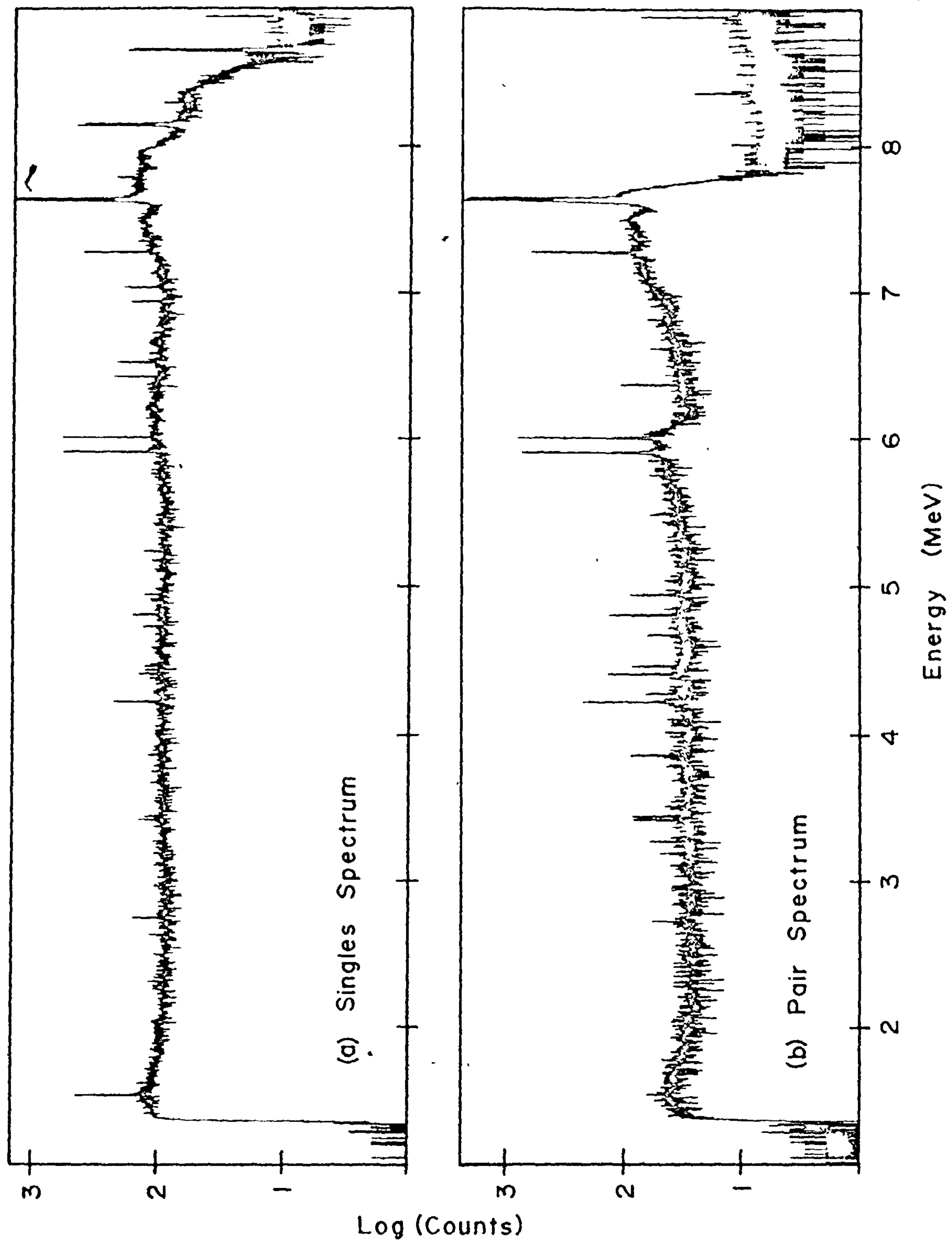


FIG. 4.3
Spectra for (n,γ) on Iron

This output was fed into the ND-3300 analyzer and the results are shown in figure 4.4 for neutron capture in iron. Part (a) shows clearly the large narrow peak which corresponds to events of good rise time. The full width-half maximum of this peak is approximately 1.1 nsec, indicating the need for highly stable electronics in the time discrimination circuitry. The long tail corresponding to long rise time events, represents rejected events and comprises approximately 25% of the events in the distribution. The distribution in part (b) was collected using the same setup except that events were collected at an average rate of 6.8 Khz instead of 1.2 Khz as in part (a). Random adding of Ge(Li) detector pulses is more predominant here and its effect is to spread the distribution into longer rise times. This indicates the potential of time discrimination to reduce the effects of random adding as well as reducing the range effect background.

Singles spectra were collected for (n,γ) on nickel using the routing logic of figure 3.6. This allowed the concurrent collection of both the good events and the events that were rejected on the basis of long rise time. The results are shown in figure 4.5(b) for the good events and 4.5(c) for the rejected components. Part (a) is the sum of parts (b) and (c)

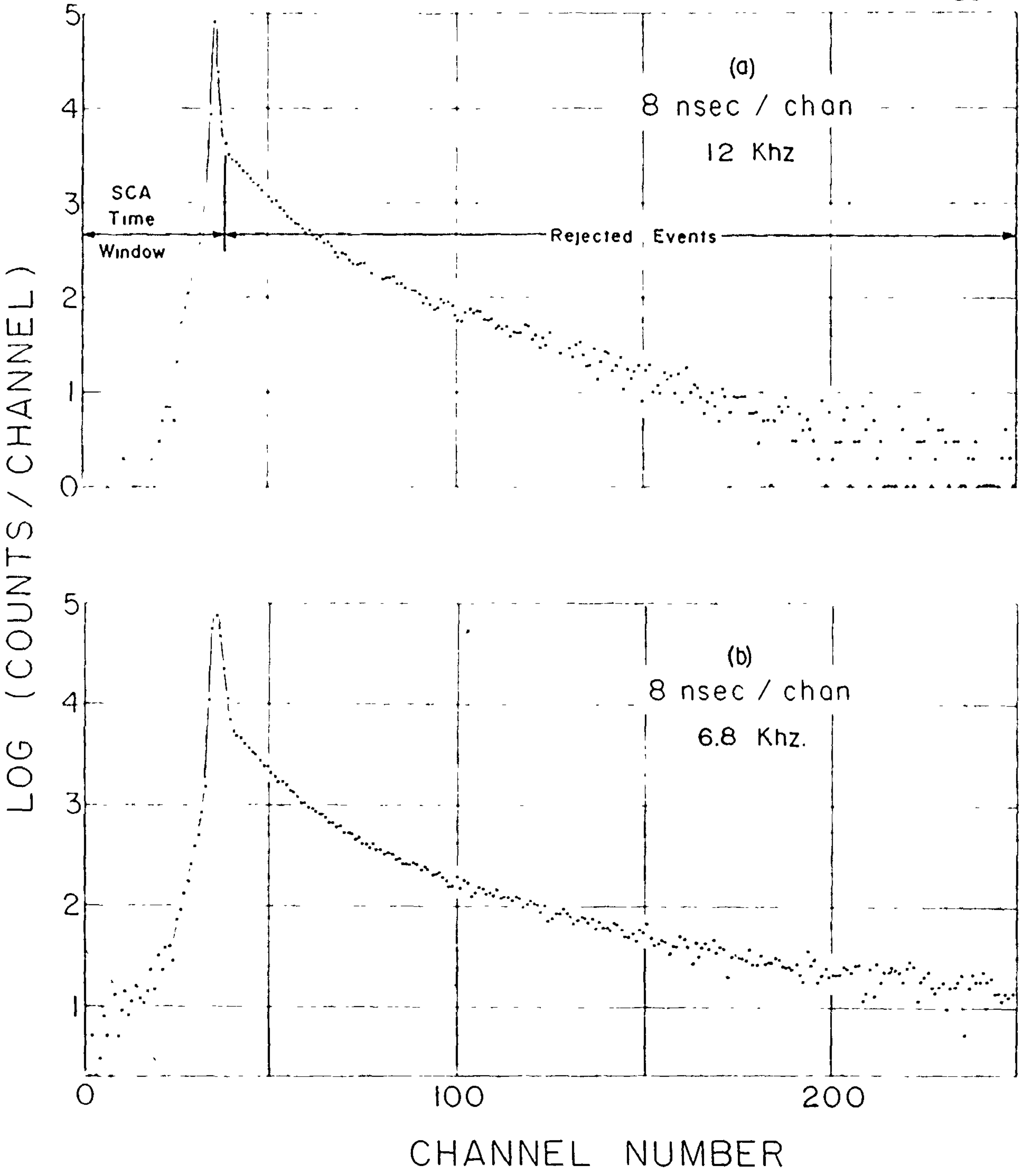


FIG. 4.4

Rise Time Distributions

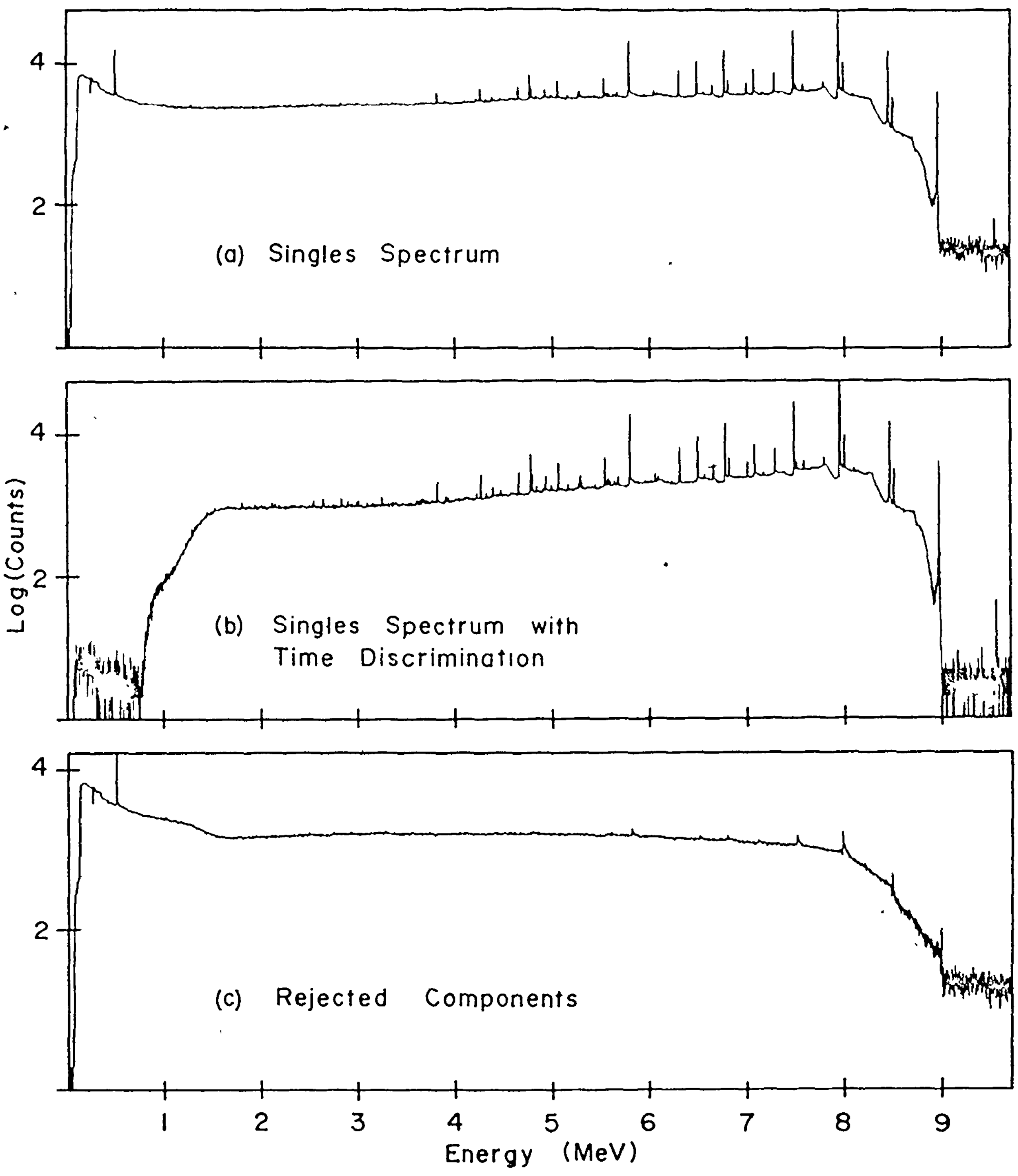


FIG. 4.5

Rise Time Discrimination in Singles Spectra for (n,γ) on Nickel and is the spectrum as it would be without time discrimination. The improvement in signal to noise ratio when rise time discrimination is added amounts to a factor of 2.3 at 4 MeV and 1.2 at 8 MeV. This improved response at low energies for singles spectra is important for applications requiring analysis of gamma energies below 2 MeV where the threshold limitation of pair spectra would be prohibitive.

Several pair spectra were collected with time discrimination. The spectra for (n,γ) on nickel are shown in figure 4.6. Parts (b) and (c) show the good events and the long rise time events respectively and part (a) shows the sum of parts (b) and (c). The peaks labelled P represent a stabilization pulser signal that is injected into the preamplifier and is used by the ADC to stabilize the gain and zero level of the system over the length of the run. The signal to noise ratio improvement here is much greater than for singles spectra due to the fact that in pair spectra the range effect provides a greater fraction of the background. This improvement amounts to a factor of 6.5 at 2 MeV and 2.5 at 8 MeV.

From the spectrum of time rejected events (figure 4.6(c)) we can see that the background behind each peak is characterized by a fairly linear gradually declining continuum of events.

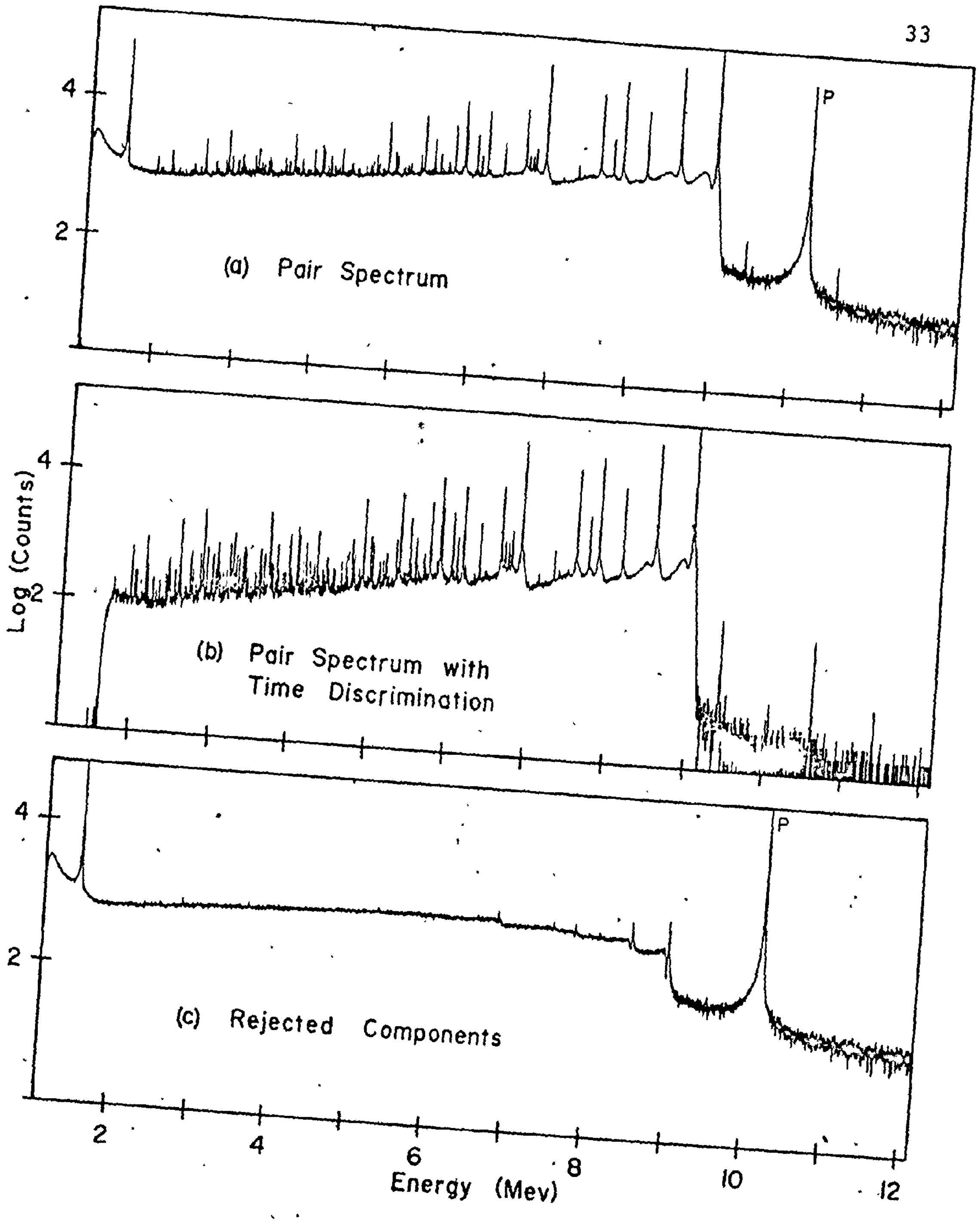


FIG. 4.6
Rise Time Discrimination in Pair Spectra for (n,γ) on Nickel

This is seen more easily in a spectrum with fewer peaks such as the spectrum for (n,γ) on nitrogen that is shown in figure 4.7(a). The area of the rejected background, i.e. the total rejected counts, due to each individual peak is a function of that peak area as well as the peak energy. Figure 4.7(b) shows the normalized background step height (STEP/PEAK AREA) as a function of energy for the nitrogen run. The solid curve is derived from a simple theoretical model for beta particle absorbtion in the detector. This model assumes that the Ge(Li) active volume is a hemisphere and the beta range in this volume is forward scattered and directly proportional to the initial beta energy. By assuming a uniform spatial distribution of pair creation events over the active volume, the probability of a beta particle ranging outside the active volume, hence contributing to the background, can be calculated. One further assumption, that the step height is proportional to total background area, allowed the curve to be plotted. The agreement between the simple model and the experimental data indicates the likelihood that the range effect does explain the background due to events of long rise time.

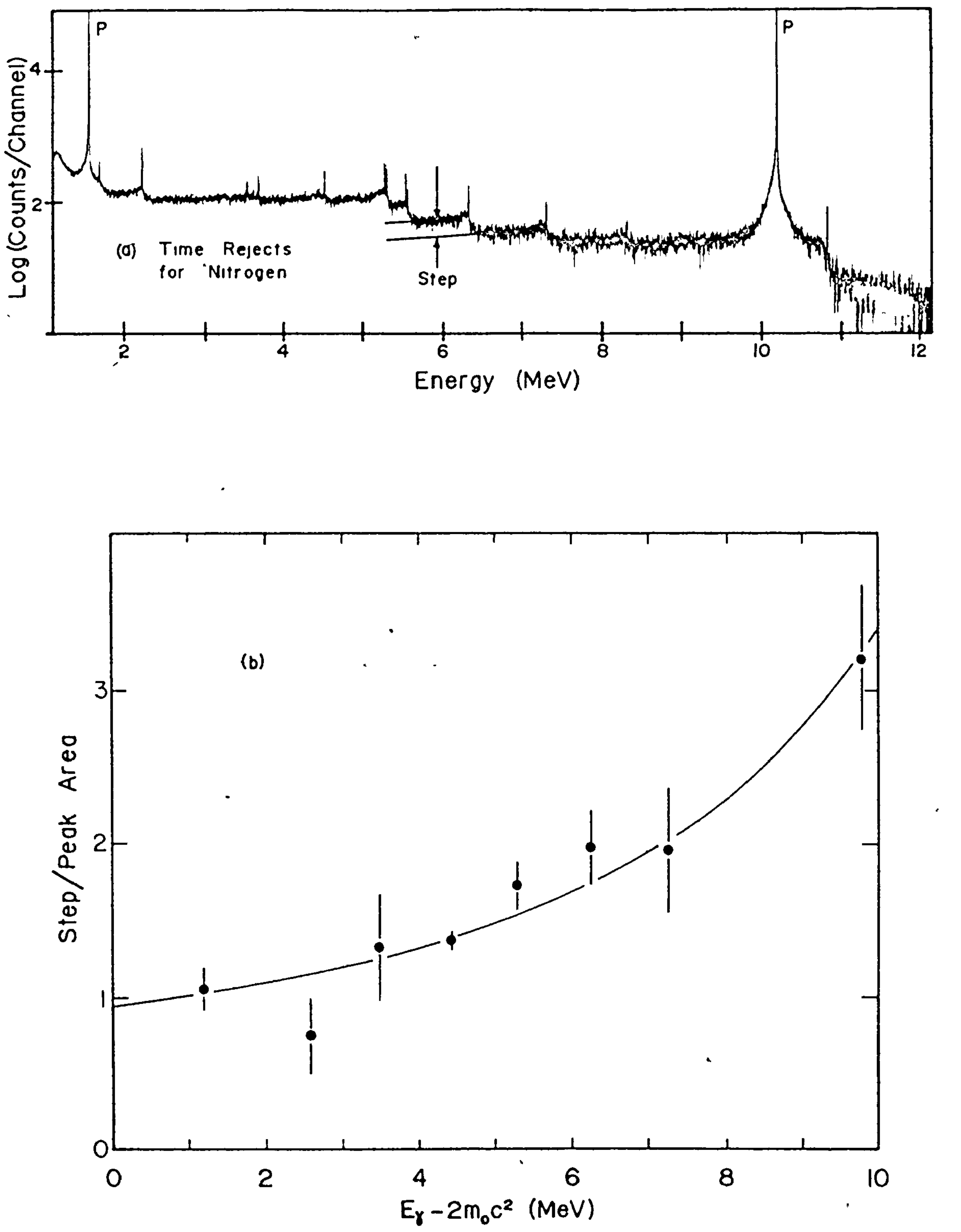


FIG. 4.7

Time Discriminator Rejected Components

Bremsstrahlung rejection provided still a further improvement in response by removing a significant portion of the background. Spectra were collected for (n,γ) on nitrogen using melamine as the target. The spectra are all pair spectra with time discrimination so that the bremsstrahlung component could be identified more clearly. Figure 4.8(a) and 4.8(b) show the original spectrum and the spectrum with rejection. Figure 4.8(c) shows the rejected bremsstrahlung events that have been routed out. The signal to noise ratio improvement with rejection is approximately 1.15 at 3 MeV and 1.5 at 9 MeV. The actual improvement for each peak depends partly on the peak energy and partly on the presence of other large peaks with energies just slightly greater. This can be seen by considering figure 4.8(c). The number of bremsstrahlung events rejected is greatest just behind the large peaks so the signal to noise ratio improvement will be greater in those areas.

From the final spectrum (figure 4.8(b)) some background events can be seen remaining. The general shape is similar to the shape of the rejected bremsstrahlung indicating that not all of the bremsstrahlung events have been detected. This efficiency factor is due to the geometry of the detector and is affected by any tendency towards forward scattering of the

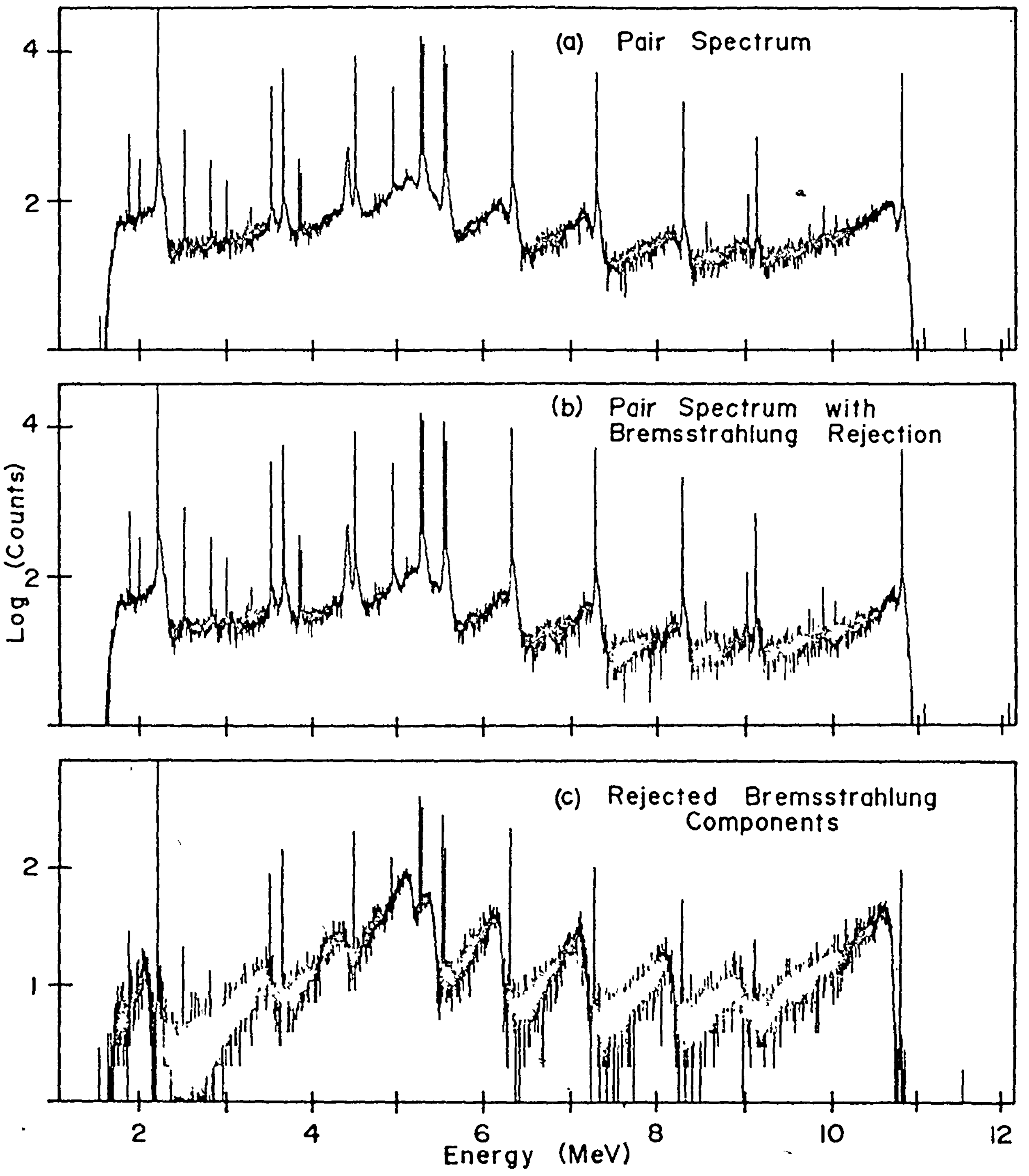
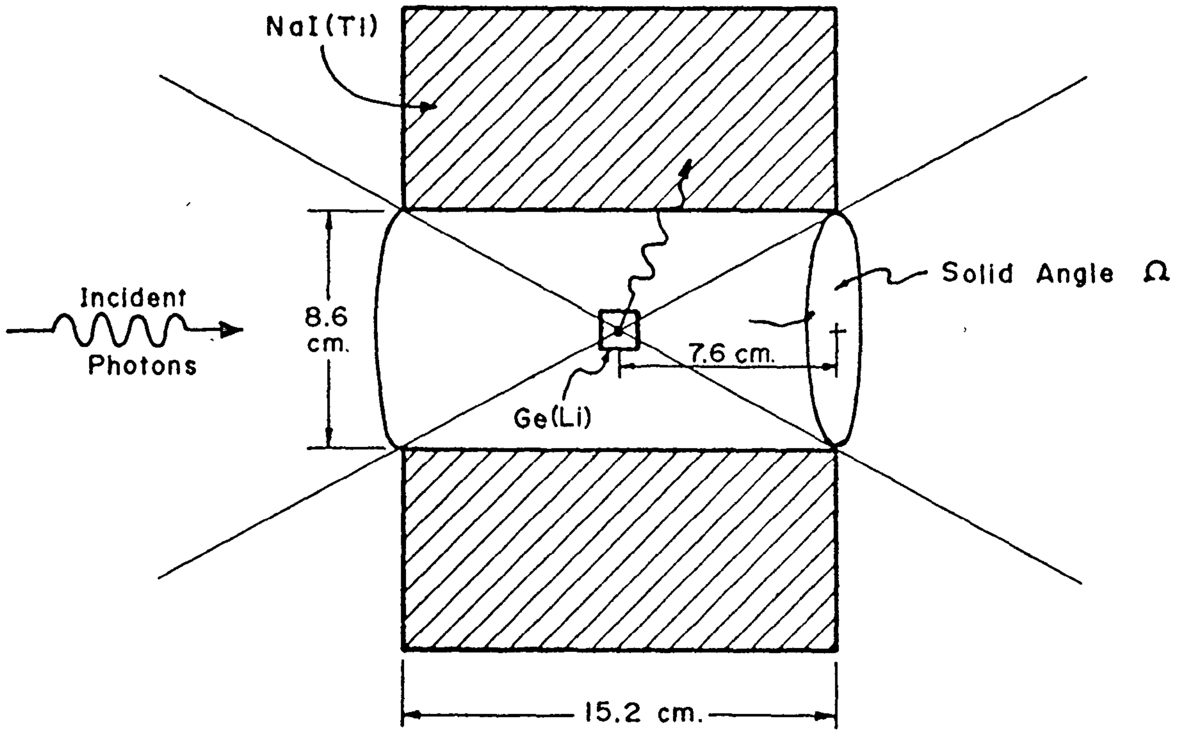


FIG. 4.8
Bremsstrahlung Rejection
for (n,γ) on Nitrogen

electron positron pair and subsequent bremsstrahlung photons. The geometry is described in chapter 3 and is further illustrated in figure 4.9. The extent of forward scattering was checked by measuring the rate of detection of bremsstrahlung (in counts per megawatt hour of reactor power) as a function of Ge(Li) detector position along the axis of the annulus. The peak rate occurred for the detector displaced only 2 mm from the center position (towards the reactor), indicating a nearly isotropic distribution of bremsstrahlung. In this case the detection efficiency is approximately 87% minus any losses due to photon absorption in the Ge(Li) detector and NaI(Tℓ) annulus supporting structure.

The spectrum of bremsstrahlung rejects in figure 4.8(c) is plotted as energy vs. log (counts). The shape of the background behind each peak can be seen then as an exponentially declining tail with a maximum height just below the peak. This is what would be expected of the process of bremsstrahlung radiation production where the probability is highest for the production of low energy photons (hence small losses) and lowest for high energy photons. This produces the exponentially shaped tail that is characteristic of bremsstrahlung. Low energy bremsstrahlung photons are also likely to be reabsorbed



For Isotropic Bremsstrahlung Emission

Fraction Collected = F

$$F = 1 - 2\Omega/4\pi$$

$$= 1 - .815/2\pi$$

$$= 87\%$$

FIG. 4.9

Bremsstrahlung Detection Efficiency

by the detector with an energy dependence as shown in figure 4.10. This produces the dip in the background that appears just behind each peak.

The analysis that was made of the rejected bremsstrahlung events was similar to the analysis for the range effect events. Both the step height and the total background area due to each peak were determined and normalized to the actual peak area. The results, given in figure 4.11, show that the bremsstrahlung background step varies directly with the peak energy, that is the initial energy available in the electron-positron pair. Considering the bremsstrahlung spectrum for monoenergetic electrons, such as the production curve in figure 4.10(a) we can see that the end point energy E_o is determined by the initial electron energy. Since the spectral curves for different electron energies are functionally similar, and the end point energy varies directly with initial electron energy, to a first approximation the spectral intensity at $E_b=0$ (i.e. the step height) will vary directly with energy as shown in figure 4.11. The fraction of the electron (or positron) energy that is converted to bremsstrahlung photons is known(19) to vary directly with that energy. The integrated bremsstrahlung energy from a peak

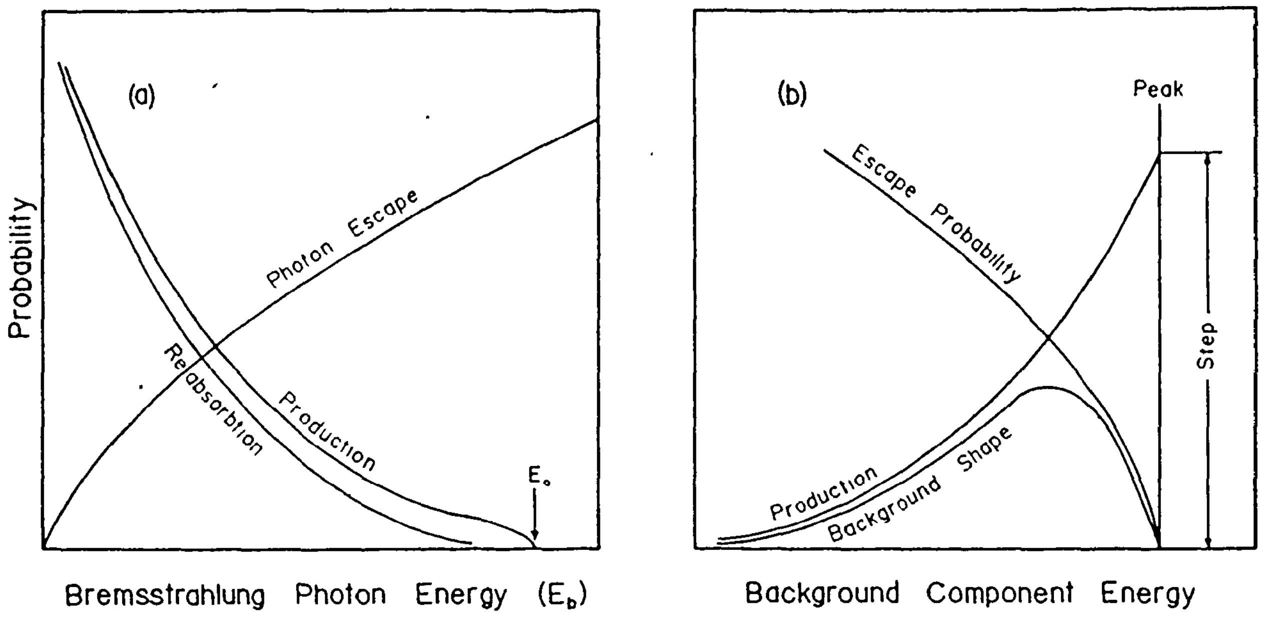


FIG. 4.10

Bremsstrahlung Production and Reabsorbtion

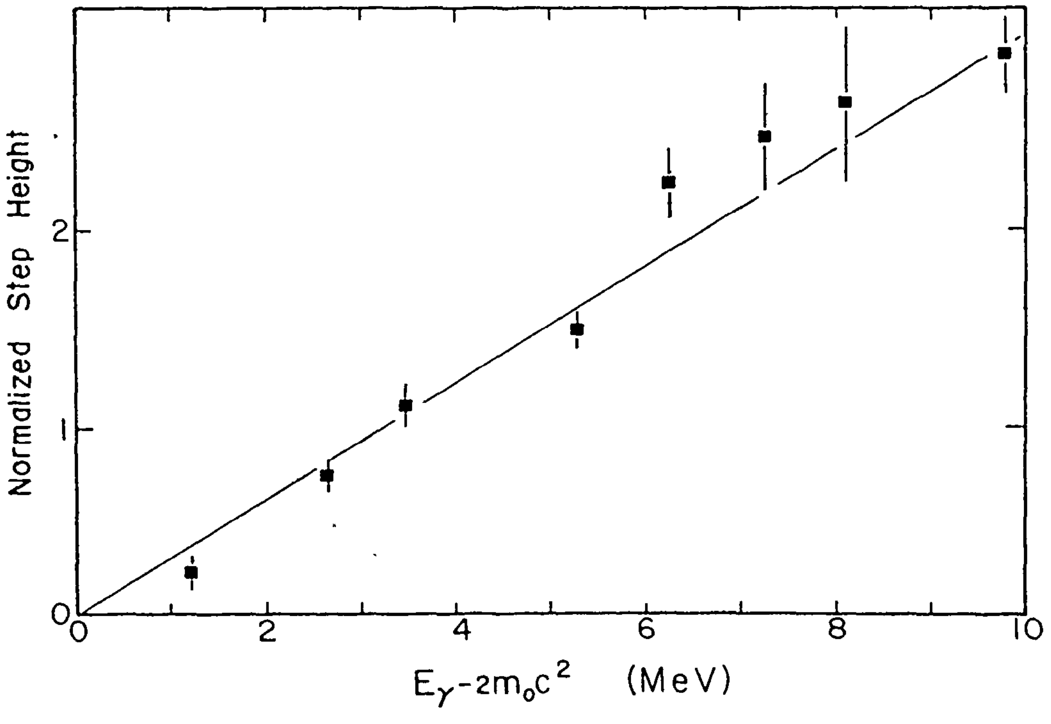


FIG. 4.11

Bremsstrahlung Rejection Components

will then vary as the square of the peak energy. A similar relation was observed to exist between the normalized bremsstrahlung background area and the peak energy. This is shown in figure 4.12. The fact that the experimental data follows in form what is known about bremsstrahlung shows that the rejected components are most likely produced from the electron and positron pair through a radiative stopping process.

A summary was made of the rejected components from a pair spectrum for (n,γ) on nitrogen. (See figures 4.7 and 4.8 for the original spectra.) The total rejected background area due to each peak was normalized to the peak area and plotted as a function of peak energy. The results, in figure 4.12, show that the major portion of the background is due to higher energy peaks. Time discrimination was seen to provide the greatest improvement. The residual background remaining in the final spectrum is given and it shows the potential for still further improvement in the response function of Ge(Li) pair spectrometers.

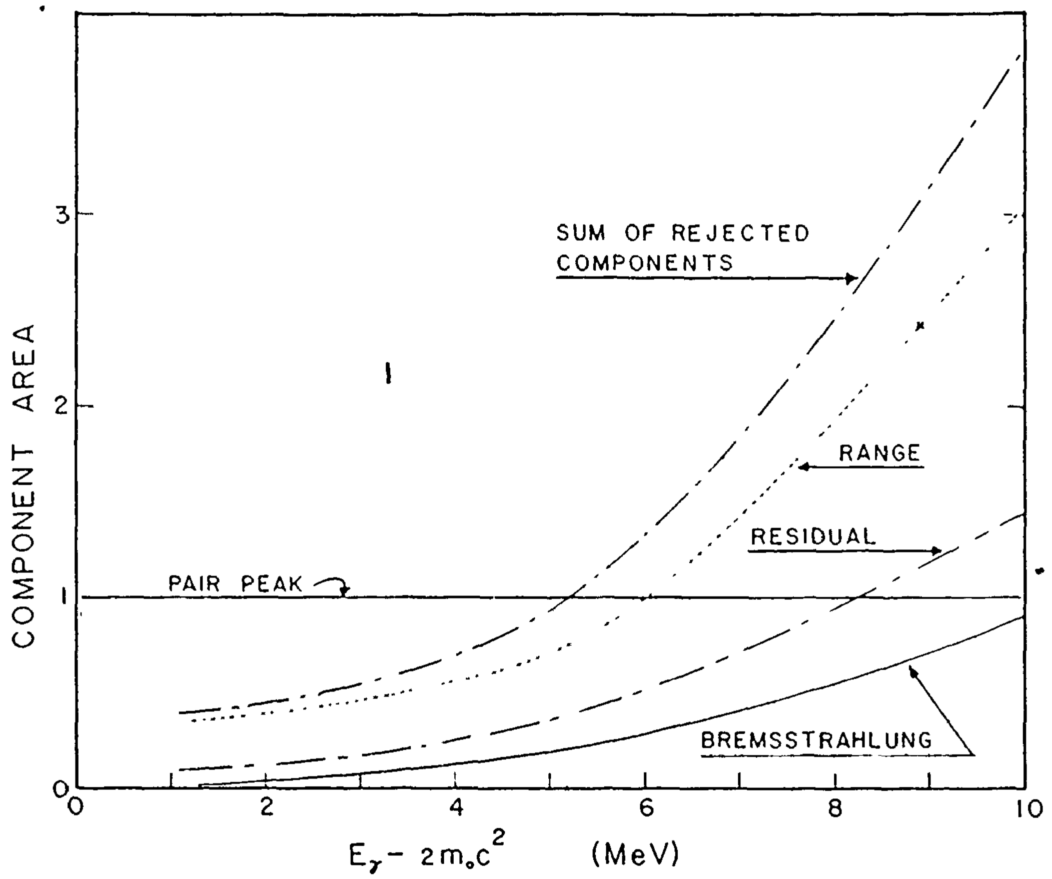


FIG. 4.12

Pair Peak

Rejected Components

CHAPTER 5

THE PULSE HEIGHT ANALYZER

5.1 The System

The output from the Ge(Li) detector amplifier consists of a 2μsec wide pulse, the amplitude of which is proportional to the energy of the incident gamma-ray. In order to obtain an energy spectrum of the incident gamma radiation one must record the probabilities that the pulse amplitude, V, lies between V and V+ΔV for a range of values of V from 0-10 volts. A pulse height or multi-channel analyzer is by far the most common device that is used to perform this function. Analyzers of this type record the spectrum by encoding the pulse height into a number and then incrementing a memory location that corresponds to that number.

Early PHA's, such as the 100 channel device described in 1951(22), were limited both in the number of channels and especially in their speed of $\approx$ 1600 events per second. The introduction of new electronic and memory technology from the fledgling computer industry in the mid-1950's produced much faster 100 channel (1954)(23) and 256 channel (1956)(24) analyzers. These devices were hardwired to perform a fixed set of functions and could analyze up to $8x10^4$ events per second. Although they used computer techniques, these early devices had none of the flexibility of a general purpose computer. It wasn't until the early 1960's with the introduction of mini-computers such as Digital Equipment Corporation's PDP series machines that analyzers were designed with many of the functions performed by software(25,26). This flexibility, however, was offset by high cost and limited software support so hardwired systems continued to dominate the scene with 1024 channel and 4096 channel analyzers being most common. The mini and micro-computer revolution of the 1970's lowered the cost of a programmable computer, and resulted in software advances so that computer based PHA's are now used extensively.

A pulse height analysis system was designed for use with the gamma spectrometer that has been described in the preceding chapters. The requirements for gamma-ray analysis indicated the need for the following system parameters:

- 8192 (8K) channel capacity maximum

- variable size spectra i.e. size selectable from 256 to 8192 channels.
- multiple spectrum storage so that spectra can be compared, added etc.
- graphical and numeric output of data.
- timer for run time and dead time measurements.
- simple on-line analysis functions eg. integration over regions in the spectra.
- data output for off-line analysis eg. mag-tape → CDC 6400.
- flexibility for future expansion or reconfiguration.

These requirements were most easily and inexpensively realized in the minicomputer based system that is shown in figure 5.1.

The analog pulses are fed into the ADC where the pulse height is encoded as a 13 bit binary number. This channel number is used by the ADC interface to compute the address in the computer's memory that stores the number of counts for that channel. The event is added to the spectrum by incrementing that memory location. Software in the computer is used to monitor and control the entire system under commands from the keyboard. The terminal and display provide data output to the user.

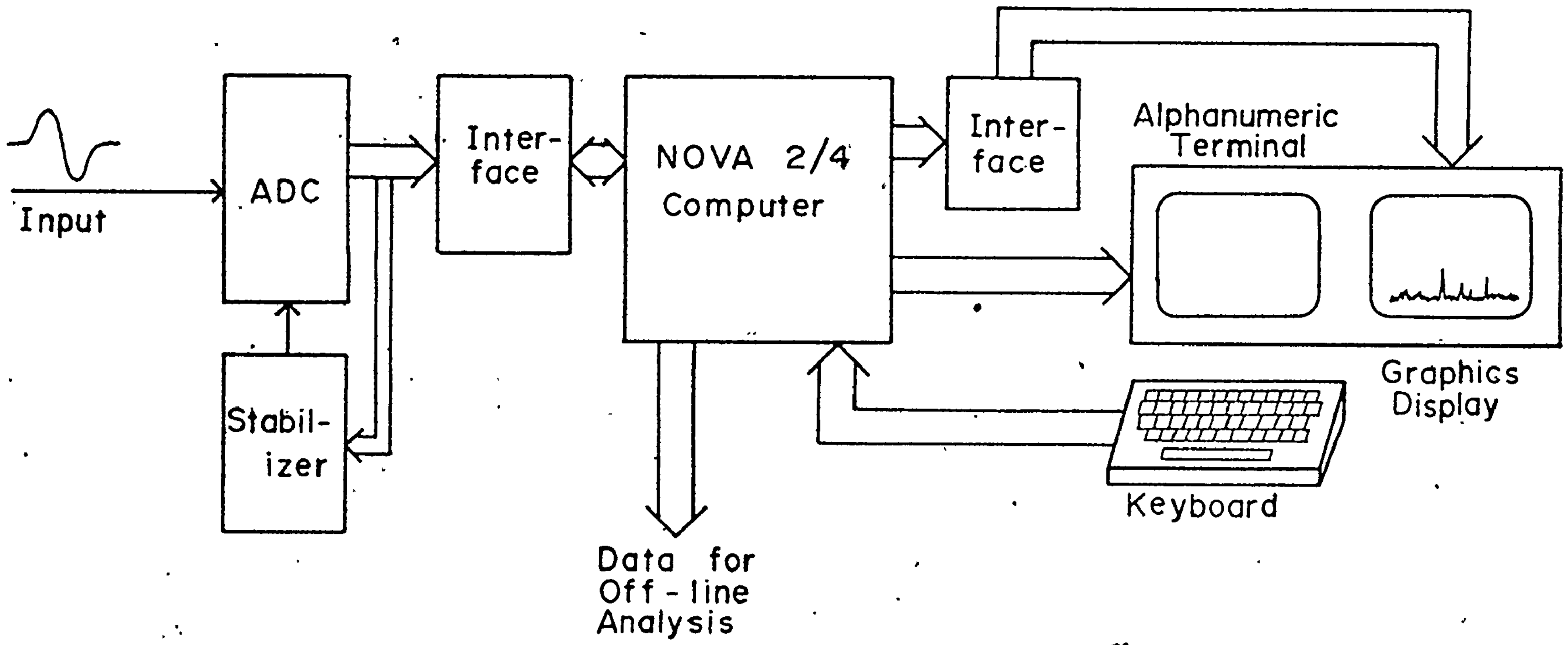


FIG. 5.1

PHA System Block Diagram

5.2 Hardware

The ADC that was selected is a Tracor Northern NS-621 50 Mhz Wilkinson type with a 13 bit capacity, giving an 8192 channel base. This unit accepts 0-10V bipolar or unipolar pulses and can provide active or passive baseline restoration. The conversion gain or ramp is selectable from 256 channels to 8192 channels and the group size and zero level controls can be used to select any portion of the ramp for analysis.

Coupled to the ADC is a Tracor Northern NS-409 digital stabilizer. This unit is used to correct for any drift in gain of the analog circuitry that may occur over long experimental runs. This is done by putting a digital window around a peak in the spectrum. This peak is most often a precision pulser signal that is injected into the detector preamplifier. Any event, that results in the ADC output falling within that window, is assumed to come from a monoenergetic source and the ADC gain is corrected so as to put that event at the center of the window.

The heart of the PHA is the Data General NOVA 2/4 minicomputer(27). It is a general purpose computer with a

16 bit word length, 16,384 words of core memory, real time clock and teletype interface. Additional features of the NOVA include 4 accumulators, a 16 level priority interrupt system, a high speed data channel and 16 front panel switches that are sensed by the software to control the graphical output. The CPU works on a 1μsec cycle time for an average execution rate of 500,000 instructions per second.

Connected to the computer is a general purpose interface bin that was designed to allow custom interfaces to be built up easily in order to service the special peripherals. The bin consists of a PC card cage, power supply, buffering electronics and device decoders, and a back panel for distributing the interface signals to the card slots. An interface can be designed and wired onto a standard sized PC board which is then plugged into one of the 8 card slots. The buffering electronics provide the link between the interfaces and the computer by both buffering and terminating all of the I/O signals. The buffering provides protection for the NOVA against wiring errors or other faults in the interface, and the termination reduces cable ringing while maintaining high switching speeds in the open collector I/O lines. The electronics are illustrated in figure 5.2. For a complete description of the functions of the signals see reference 27 pages A3-A6. Signals that go from the NOVA to the devices are low pass filtered to remove the effects of ringing and then are squared up using an AND gate to give a fanout of 10 to the bin. Signals from the interfaces to the NOVA must allow wired OR'ing since each device may selectively activate each signal. The signal lines are pulled up to +5V through a 2.2 K resistor and the devices drive the lines low using open collector gates such as the 7403 or 7438 NAND's. The 16 bidirectional data lines were buffered in both directions with tri-state transceivers (DM 8833). The tri-state concept was used here instead of the standard open-collector philosophy because of the increased noise immunity of the tri-state buffers and also because no open collector transceivers are currently available that match the packaging density of the DM 8833. This design added one extra signal requirement to the standard NOVA I/O lines. When a device puts data on the 16 bit data bus it must also pull the $\overline{\text{STROBE}}$ line low to disable the receiver and enable the transmitter that sends the data to the NOVA. The bin

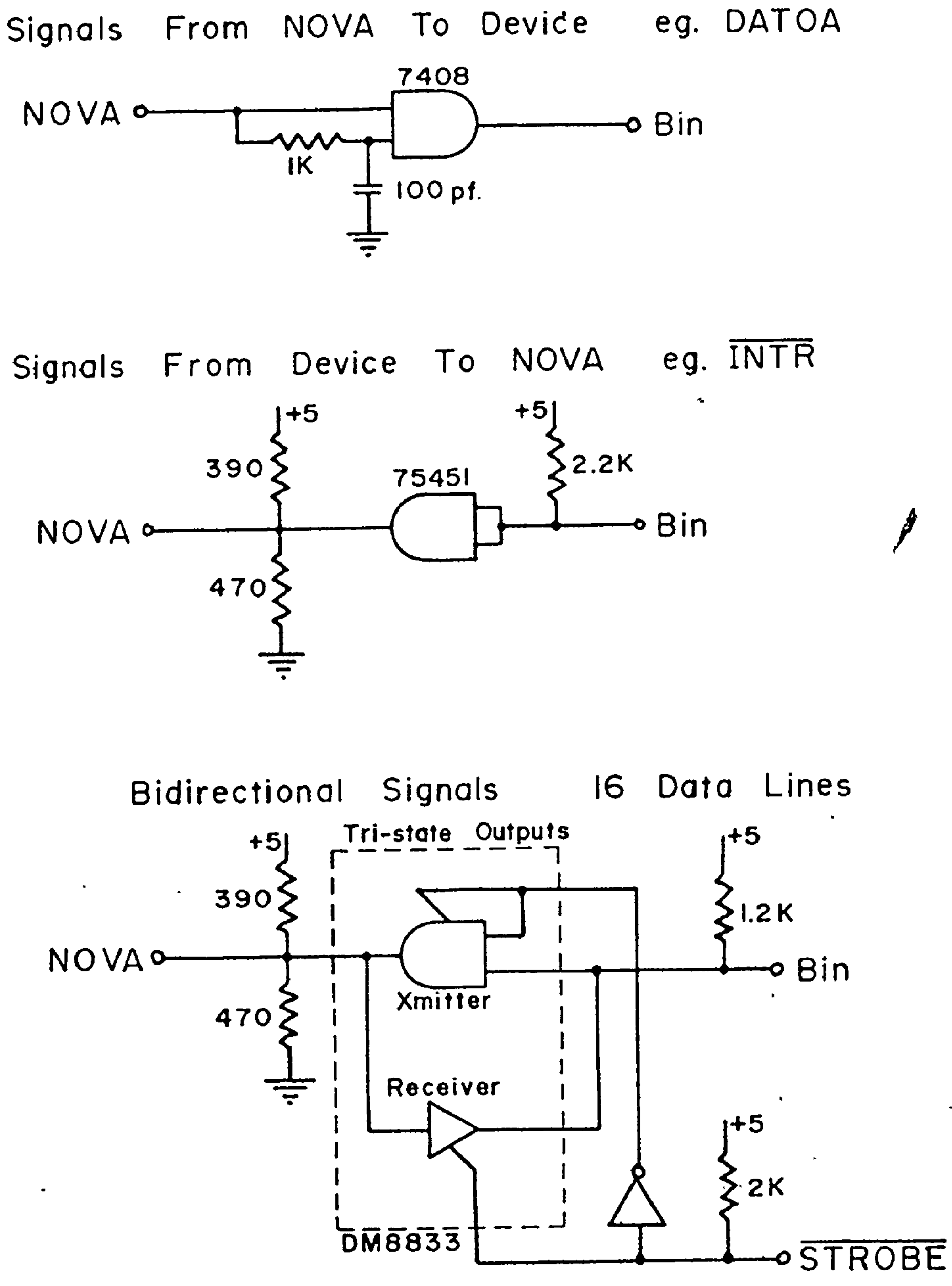

Signals From NOVA To Device eg. DATOA
7408
NOVA
Bin
1K
100 pf.
Signals From Device To NOVA eg. $\overline{\text{INTR}}$
+5
390
470
75451
2.2K
NOVA
Bin
Bidirectional Signals 16 Data Lines
Tri-state Outputs
+5
390
470
NOVA
Xmitter
Receiver
DM8833
1.2K
Bin
2K
$\overline{\text{STROBE}}$

FIG. 5.2

NOVA Bin Buffering Electronics

performs the device decoding for all devices in order to save parts in the interface. Each slot can be assigned one device code, that is selected with jumpers on the power supply device decoder board.

The back panel distributes all of the I/O signals, plus the +5V and -12V power supplies, to the edge connectors that hold the interface boards. Figures C3 and C6 give a list of the I/O signals and their pin assignments for the back panel connectors.

Several bins were assembled for NOVA's that are used in research as well as in undergraduate instruction. Interfacing was found to be greatly simplified when packaging and power supply problems were eliminated, leaving the designer to concentrate on the logic. The interchangeability of interfaces is also expected to increase the usefulness of any custom peripherals.

The first interface that was designed was for the ADC. This interface is required to take the 13 bits of data from the ADC and control all of the timing and data transfers to effect the incrementing of the correct word in memory. Each spectrum is stored in the NOVA's memory in a block or buffer of consecutive memory locations. For a given buffer, the starting address specifies the location of channel 0 of the spectrum. When an event is detected, the ADC produces a 13 bit channel number that must be added to the buffer starting address to get the absolute memory address of the word that is incremented. This addition can be done by one of two possible design routes, either entirely by hardware using the NOVA's data channel (direct memory access) or by a combination of hardware and software using the program interrupt facility. The two methods each have their own advantages and drawbacks. Data channel accesses, while requiring more complex hardware, are inherently faster than standard interrupt driven transfers to the accumulators. The standard transfers, however, allow much more extensive programmed pre-processing of the data before storing. This could be scaling or discriminating or setting up a two-dimensional array etc. In order to maintain flexibility and maximize speed, provision was made in the interface design for both types of transfer although the system software was written to utilize the data channel since no pre-processing is required for the present experiments.

An outline of the data flow and control is given in figure 5.3. When the ADC is finished a conversion it presents the 13 bit channel number and raises the STORE signal to a logical 1. The interface "immediately" responds by latching the data and clearing the ADC, allowing it to be available for another conversion with a minimum of delay or dead time. This datum is added to the buffer starting address which has previously been loaded into the base address register by the program. In an effort to save hardware the base address register was made only 12 bits long with the least significant 3 address bits taken as 0's. The only restriction that this places on the software is that buffer starting addresses must be a multiple of 8. The adder output is the absolute memory address of the channel of interest.

The STORE signal also goes to the control logic where either a data channel request or an interrupt request is generated, depending on which mode of transfer the program has selected. A simplified block diagram of this logic is shown in figure 5.4 and the detailed schematics of both the data flow and control logic circuits are given in appendix C.

If the data channel mode is being used the program will have set the DCH ADC flip-flop with an NIOP instruction

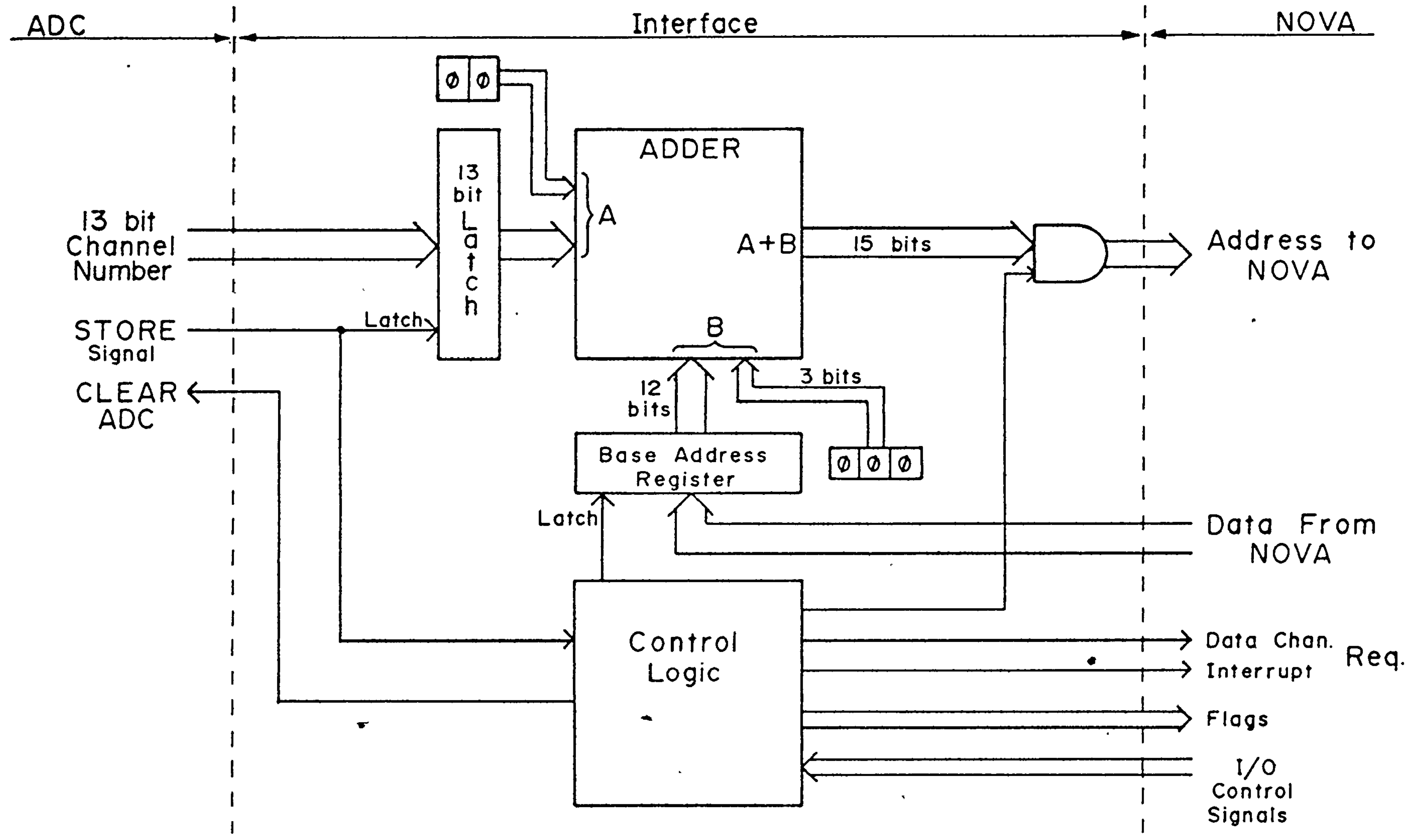


FIG. 5.3

ADC Interface Data Flow

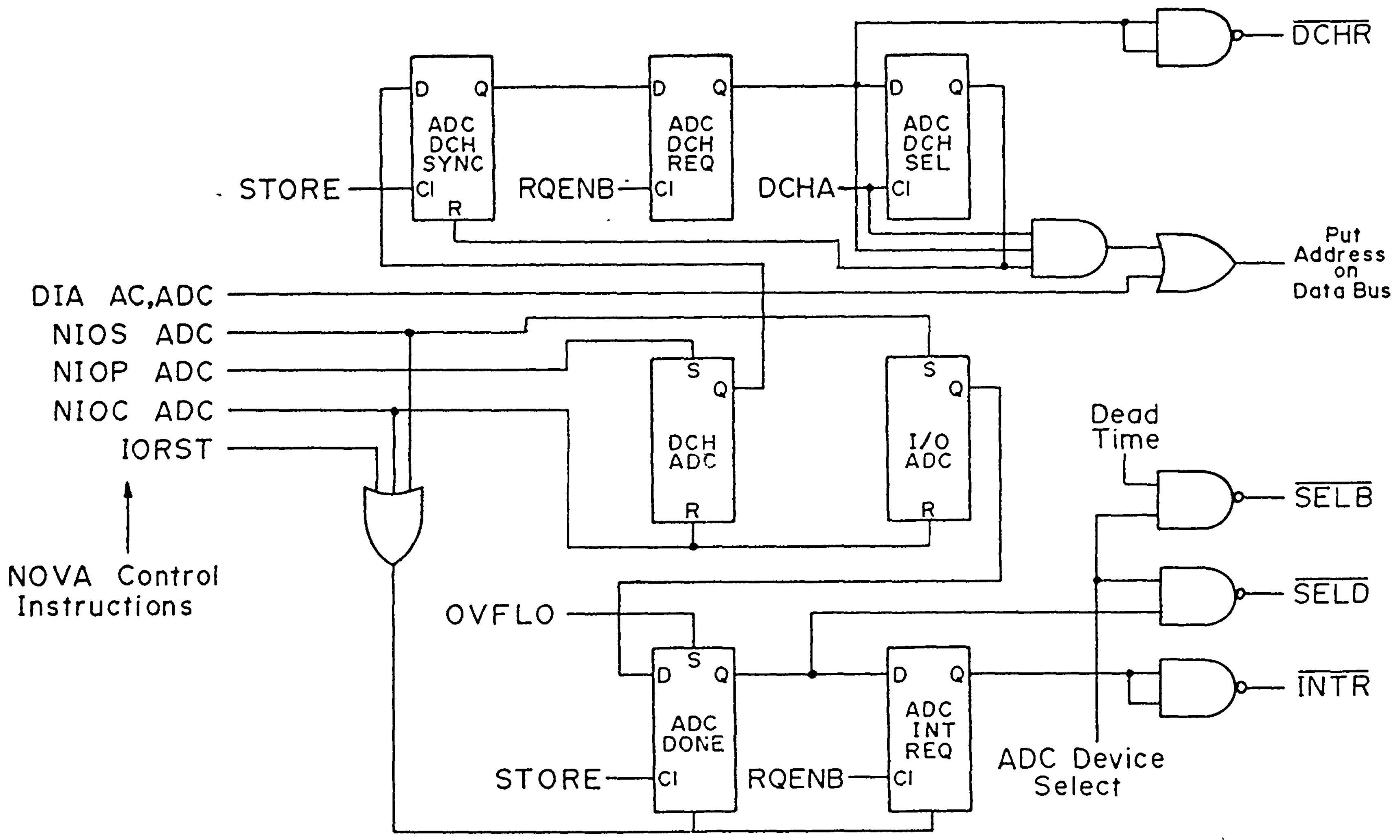


FIG. 5.4

ADC Interface Control Logic Functional Blocks and the STORE signal will then clock a 1 into the ADC DCH SYNC flip-flop. At the next free memory cycle the computer generates the RQENB signal and the ADC DCH PRQ flip-flop is clocked to a 1, thereby requesting a data channel access by pulling $\overline{\text{DCHR}}$ low. The computer acknowledges this request with a high pulse on DCHA. This signal is used by the interface to strobe the absolute address onto the data bus and reset its DCH SYNC flag, thereby terminating the process. The computer takes care of incrementing the memory word whose address is on the bus and if that incrementing causes an overflow (i.e. if the word count goes from $2^{16} - 1 =$ 65,535 to 0) then the signal OVFLO is given and an interrupt request is generated. This occurs by setting the ADC done flag and waiting for the completion of a CPU instruction cycle to generate RQENB which sets the ADC INT REQ flip-flop and pulls the $\overline{\text{INTR}}$ line low. This program interrupt can be used by the software to process the overflow by recording the channel number or decrementing the word etc. After the initial setting of the DCH ADC flip-flop that selects the data channel mode, all transfers are completed transparently to the program and require no software intervention, provided there are no overflows.

If the standard interrupt driven transfer mode has been selected by the software through the issuance of an NIOS ADC instruction that sets the I/O ADC flip-flop, the STORE signal will generate an interrupt request in the same manner as the overflow. The software must check to see which device caused the interrupt and the ADC is tested with a SKPDN ADC instruction. This instruction causes the next instruction to be skipped if the ADC DONE flag is a 1, and it can be used to branch to the ADC service routine. The service routine can read in the channel number into one of the accumulators (AC) with the instruction DIA AC,ADC. The buffer starting address can be added to the channel number using software or in fact could be done with the hardware using the base address register and the adder.

The speed of the ADC and the transfer time of the interface both determine the length of time, after an event comes in, that the system is insensitive to further inputs. This "dead time" is an important measure of the system performance. For the NS-621 ADC the dead time is equal to the rise time of the input signal plus 20 nsec x channel/number plus 3.85 μsec. The interface is set up to latch immediately onto the data from the ADC. This means that the time for the interface and computer to process the data will not add to the dead time unless a new event arrives and is encoded within the processing time. This time is less than 5.4 μsec for the data channel mode and will be at least 20 to 30 μsec using standard I/O and software.

A feature has been added to the ADC interface to allow dead time measurement and is shown in figure 5.4. The dead time signal (DT) from the NS-621 is normally at a logic 0, but goes to a logic 1 whenever the ADC is dead or insensitive to inputs. This signal can be tested by the program in the same manner as the ADC DONE flag. If DT is a 1 and the ADC device code is pulsed, the select busy ($\overline{\text{SELB}}$) line is brought low. The program checks this with a SKPBZ ADC or a SKPBN ADC instruction to produce a conditional program branch. (Note: see. reference 27 pages 2-21 to 2-26 for more detailed information on the NOVA's I/O instruction set.) The software can increment a run time counter at a fixed or random rate and at the same time increment a dead time (or conversely a live time) counter depending on the state of DT. Comparing the dead time and run time counters will give an estimation of the percentage dead time within statistical errors. These errors will decrease for longer run times and higher event rates.

Hardware on the output side consists of two subunits. The video terminal and keyboard allow the user to monitor and control the functions and status of the program as well as examine the accumulated data numerically. The graphics display allows the user to examine the data as a point plot or histogram of counts vs. channel number.

The terminal is a CT-1024 alphanumeric display system that was constructed with components purchased from Southwest Technical Products. This low cost terminal can display two pages of characters, each with 16 lines of 32 characters, on a standard video monitor. The keyboard is ASCII encoded with the standard teletype format. Data transfers to and from the terminal are handled through the Data General supplied teletype interface (TTY) at a rate of up to 75 characters per second. The interface makes use of the interrupt facility of the NOVA to allow interrupt driven software to be written. Striking any key on the keyboard will interrupt the program and signal it to read in a character. Also, after a character has been outputted and processed, an interrupt is generated, indicating to the program that it can output another character.

The graphics display interface and controller were custom designed to allow a 485 channel by 1024 point single line plot to be displayed on a standard interlaced raster scan video monitor. The only modification required of the monitor is that the raster is rotated 90^{0} counter-clockwise so that the horizontal lines now scan from the bottom of the screen to the top and the vertical trace now sweeps from left to right. Each scan line corresponds to one X-address and the position along the line is the Y-address.

The display controller is illustrated in figure 5.5 and is most easily explained by following the sequences of one complete frame of 485 X coordinates. Note that due to the interlacing it takes two vertical (i.e. left to right) sweeps to make one complete frame with the first sweep displaying all even numbered X coordinates and the second sweep all odd numbered coordinates. The sequence is as follows: At the end of a frame the 9 bit line address counter is cleared to 0 and an interrupt is generated. The program must respond to the interrupt by reading in the X line address, retrieving the corresponding Y coordinate from a memory display buffer

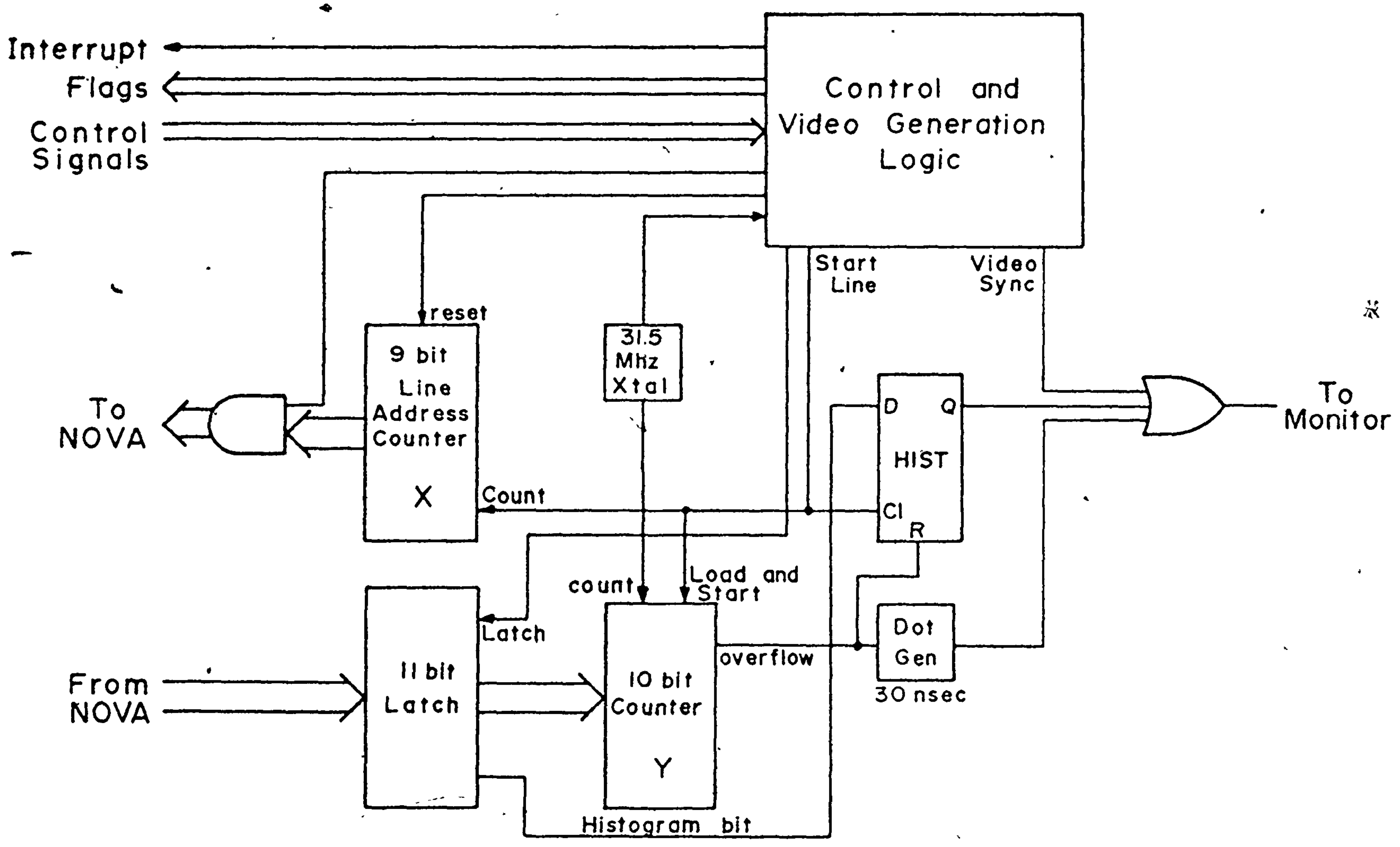


FIG. 5.5

Graphics Display Interface and Controller Principles and then outputting it to the latch. The 9 bit X address is read into the accumulator bits 7 to 15 with a DIA AC, DPY instruction where DPY = 36 and is the display's device code. The Y coordinate is outputted from bits 6 to 15 of the accumulator with a DOA AC, DPY instruction. In addition if bit Ø of the outputted word is set to a 1, the display will be in the histogram mode. When the video generation logic signals the start of a line, the Y counter is loaded with the complement of Y (i.e. -Y) and it starts to count up at 31.5 MhZ. At the same time the line address is incremenred and another interrupt is generated, thereby requesting the next point from the computer. If the histogram mode bit had been set the HIST flip-flop is also set to turn on the display beam. After Y counts (= Y X .03175 μsec) the counter overflows and a dot is flashed on to the screen. The HIST flip-flop is also reset at this time. The line scan and frame sweeps are both linear with time so the result is a linear X-Y plot.

Each scan line takes 63.5 μsec so the computer must respond within this time to maintain a steady flicker-free display. Since the display service routine requires about 27 μsec, this routine will occupy a major portion of the processor's time. This amounts to approximately 40% of the

CPU time but will not affect the operation of the ADC since the data channel has priority over the CPU in memory accesses. A breakdown of the control and video logic functions is shown in figure 5.6. A master crystal controlled oscillator is used to generate all timing and control signals. The choice of 31.5 MhZ provided a convenient count rate for the Y counter and at the same time was easily divided down to the 1.26 MhZ signal that was required by the TV sync generator. The MM 5320 sync generator is a single integrated circuit that produces all of the timing signals normally required by a television system. With a 1.26 MhZ input the frame sweep rate is exactly 60 HZ. This matches the line frequency and is very important to minimize instabilities in the display.

A scan line starts after the composite sync signal has initiated the monitor retrace. The color burst gate then clears the margin counter, clearing the carry output to 0 and re-enabling the counter. After 10 counts at .63 MhZ the counter overflows and the carry goes high, disabling the counter and triggering the START LINE pulse. The carry output will also clock the DPY DONE flag to a 1, initiating an interrupt request, provided the DPY ENABLE flag has been previously set by the software by issuing an NIOS DPY instruction. Interrupts can be inhibited by clearing the DPY ENABLE flag with an NIOC DPY

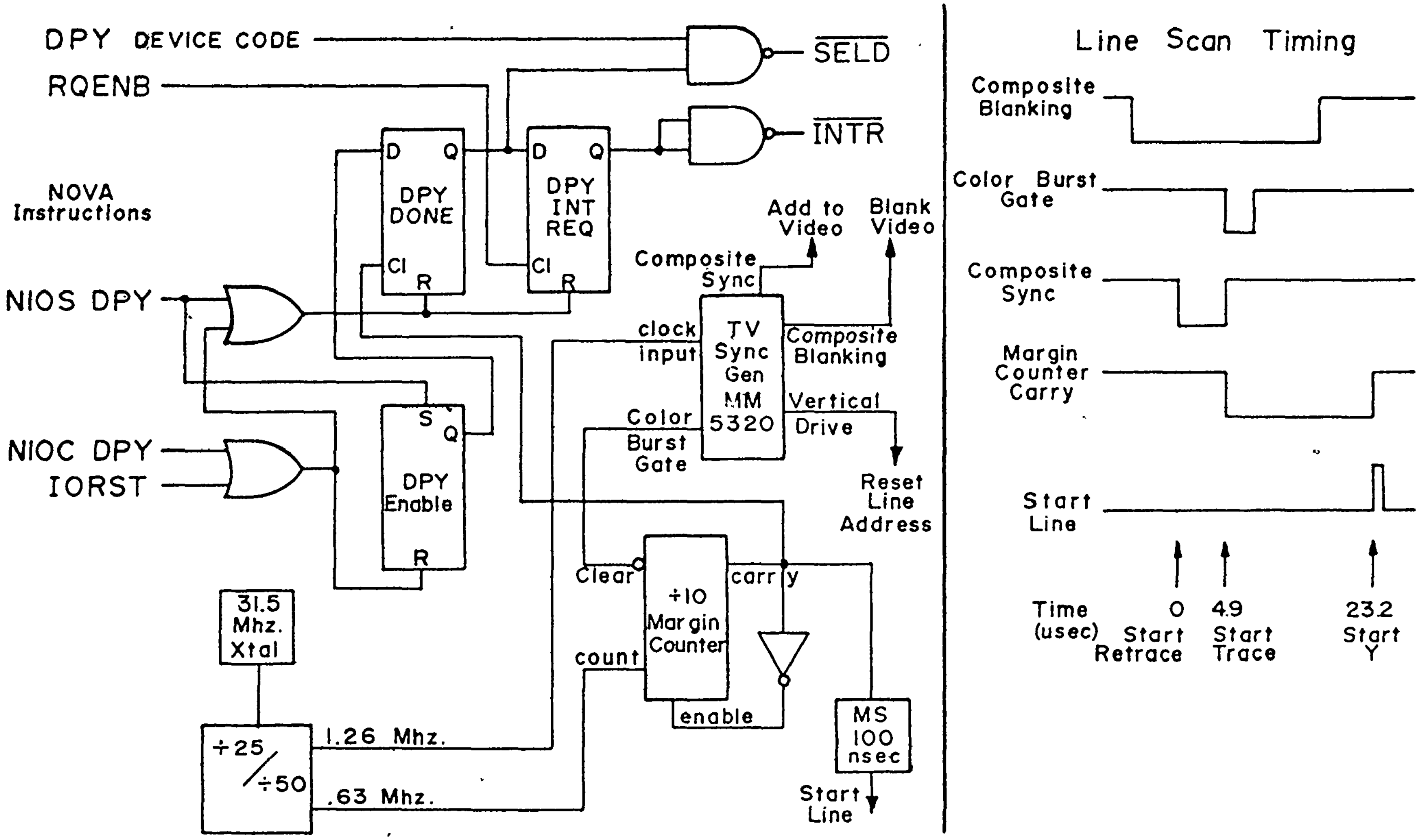


FIG. 5.6

Display Control and Video Logic

instruction. Additional outputs from the sync generator are used to blank the video during retraces (composite blanking) and synchronize the raster of the monitor (composite sync). Complete detailed schematics of the interface and controller are given in appendix C.

Another piece of hardware that is used by the system is the real time clock (RTC). This is simply a device that interrupts the NOVA every 1/60 of a second. The program uses these interrupts to initiate certain tasks as well as keep track of real and live run times for each experiment. The real time clock is supplied as part of the teletype interface and is completely described in reference 27 page 3-29.

One final piece of equipment is used by the PHA although it is not actually part of the system. A PDP-15 computer, instead of the terminal, can be connected to the NOVA's teletype interface. This allows the PHA to use the high speed paper tape reader and punch that are in the PDP-15 system. Data can also be outputted on 7 track ½ inch magnetic tape in a format that can be used by a number of large analysis programs that have been written for the CDC-6400. This off-line analysis utilizes facilities such as hard copy printers and plotters that are not otherwise available to the PHA

5.3 Software

The software for the PHA system consists of a program, called NOVADC, that resides in memory in the NOVA. This program interacts with the user to monitor and control all data and devices by performing the following functions:

- Organize experiment buffers
- Initialize and start ADC interface
- Keep track of run times
- Process channel overflows
- Service and update display
- Format output to the user.

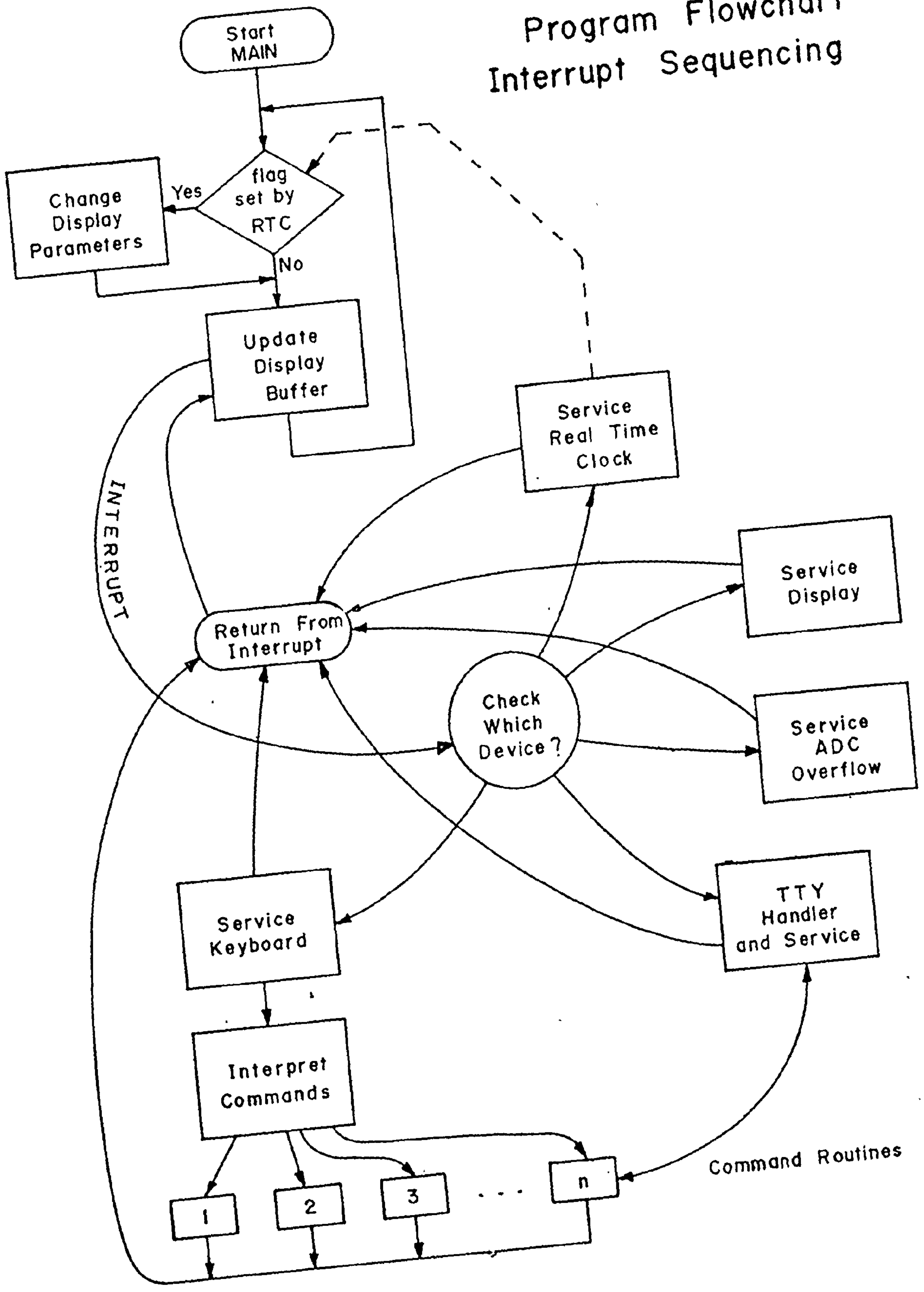
NOVADC was written in the NOVA's assembly language in an effort to produce the most efficient coding in terms of memory utilization. Each assembler instruction uses only one memory word for storage so the result is a program that not only uses fewer memory words but requires much shorter execution times than a program written in a higher level language such as Fortran or Basic. NOVADC was written using a technique called modular programming. Programs written this way are broken up into functionally separate blocks or modules which can be further separated until the program consists almost entirely of small easily debugged segments. The segments can be completely tested then linked together into the major program. The structure of the links can also be set up to allow additional modules to be easily added, thereby facilitating future expansion. Modular programming has been used for a number of years but has only recently been formalized[28]. It is a recognized technique for creating order in a large programming project and certainly in this case greatly aided in the writing of NOVADC.

The architecture of NOVADC was designed to work with experimental data that is collected and stored in fixed sized memory areas called buffers. These buffers are set up by the user through keyboard commands. The available memory (13,785 words left out of 16K) can be divided into a maximum of 16 separate buffers, each of which may range in size from 8 to 8192 channels. The buffer size, however, must be a multiple of 8 due to the restrictions of the ADC interface that ignore the least significant three bits of the base address.

The structure of NOVADC is centered around the use of the interrupt system. When enabled, this system allows the devices to request service by forcing the program to branch to a routine that will identify which device interrupted the processor, then branch to the appropriate service routine or module. With the exception of the routine called MAIN, all tasks and processes are initiated and synchronized by interrupts from the appropriate device, as shown in figure 5.7. Routine MAIN is entered when NOVADC is started and it is executed over and over. In the absence of any interrupts it has complete control of the CPU. The function of MAIN is to take data from the selected buffer in memory, scale it according to a set of variable parameters, and dump it in the special display buffer from which the display service routine gets its Y coordinates. Also, if a software flag has been set by the real time clock service routine, the switch register on the NOVA's front panel is scanned and the display parameters are changed according to the switch positions. These parameters include Y scaling, X scaling and shifting, linear or logarithmic plot and histogram or point plot. In addition a pointer can be turned on by selecting the histogram mode of display for just one channel. The switches can be used to shift the pointer position and type out the pointer channel number and contents. For details of the switch

FIG. 5.7

Program Flowchart
Interrupt Sequencing



register functions see appendix A page 2.

The interrupt is most easily seen by following the process of one interrupt, in this example from the display. Names in capitals refer to labels in the program listing that appears in appendix A. We start with the interrupt enabled and MAIN being executed. When an interrupt is requested the NOVA stores its current program counter (i.e. the address, we will want to return to) in location 0, disables the interrupt and jumps to INTR, the start of the skip chain. This segment tests each of the devices in the order of their need for immediate service and in this case will cause a branch to DSOUT, the display service routine. This routine saves the accumulator contents that MAIN was using, then reads in the X coordinate, adds to the display buffer starting address and gets the Y coordinate from memory. This is outputted to the display then the accumulator values are restored and the interrupt system re-enabled. Control is passed back to MAIN by jumping to the instruction whose address has been stored in location 0.

The rest of the device service routinesare entered in a similar manner. If the routines had saved all four of the accumulators in locations TACØ, TAC1, TAC2 and TAC3, they can return to MAIN by jumping to RESTR, the return segment of the TTY service routine. This saves possibly redundant coding. Note that all device service routines are executed with the interrupt system completely disabled. Although it is not the most efficient method of utilizing CPU time, this single priority level of interrupt (i.e. no interrupts within interrupts) was used to simplify and shorten the program.

The foregoing illustrates a typical device service routine. The following routines, although more complex, follow the same general structure in performing their functions. The real time clock service routine is used for two functions, the first of which is to keep track of run times. Since this routine is entered regularly every 1/60th of a second, by incrementing a counter the elapsed time can be measured. When this counter reaches a count of 360 (i.e. after 1/10 minutes) it is reset to 0 and channel 1 (the run time counter) of the active buffer is incremented. Also the dead time signal of the ADC is checked and if it is a Ø then channel 2 (the line time counter) of the active buffer is incremented.

The second function of the real time clock service routine is to set the display parameter update flag DCFLG. This is done by first checking the switch register to see if any update is required. If not, the update loop counter, DCNT, is set to 1 and the counter DCINT is set to 16. When an update is required DCNT is decremented and if it equals 0 the parameter update flag is set. At the same time the counter DCINT is decremented and loaded into DCNT. The result is as follows. When an update is first requested the flag DCFLG is set within 1/60th of a second. After that DCNT must be decremented 15 times before DCFLG is set. This causes a .25 second delay. The next time through, since DCINT is decremented every time, the delay is only 14 clock pulses or .233 seconds. This causes the parameter update flag to be set at an accelerating rate so the display then will appear to change slowly at first then accelerate. For the user this allows the display parameters to be finely tuned by setting, then quickly zeroing the appropriate switches. Coarse tuning is effected by setting and leaving the switches on.

The ADC overflow service routine is entered whenever any memory location has been incremented, via the data channel, and has overflowed from 65,535 counts to 0 counts. Since that channel no longer holds the true number of counts some record must be made of the overflow. This is done by simply reading in the address from the ADC, subtracting the buffer starting address to get the channel number then typing out "OVERFLOW AT CHAN" and that channel number. The ADC interface is restarted and the routine returns to MAIN.

The keyboard service routine is entered whenever the user strikes any key on the terminal's keyboard. The character is first read into accumulator Ø then checked to see if it was any of the special characters. A control R causes a jump to the start of NOVADC to re-initialize the system. An ESC or ALT MODE causes a jump to RETRN to effect a partial re-initialization. Any other character is treated as part of a command string and is stored in the command buffer CMB. A rubout (RUB), delete (DEL) or cancel (CAN) can be used to erase the last character from the command buffer and a carriage return will terminate the command string and cause a branch to the command interpreter routine. If the inputted character had just been stored in the command buffer it must be echoed to the terminal for it to appear on the alphanumeric display. This echoing is done using a subroutine called OUT that is part of the terminal output (TTY) handler and service routine. This subroutine is used for all output to the TTY and will be discussed along with the service routine. After reading in and storing a character the keyboard service routine jumps to PESTR to re-enable the interrupt and return to MAIN while waiting for the next character. If a branch had been made to the command interpreter, the interrupt system is kept disabled while the command is decoded and executed. Only after that is the command buffer cleared, the interrupt enabled and a return made to MAIN to wait for the next command.

The TTY service routine is complex because it also includes the TTY handler and output subroutine. The routines involved are most easily described by following the logic path as a string of characters is typed out. This string could come from the keyboard service routine, the ADC overflow service routine or more likely from one of the command execution routines. The sequence is flowcharted in figure 5.8 and proceeds as follows: The first character to be outputted is loaded into accumulator Ø and a jump is made to subroutine OUT. If the terminal is not busy the character is immediately

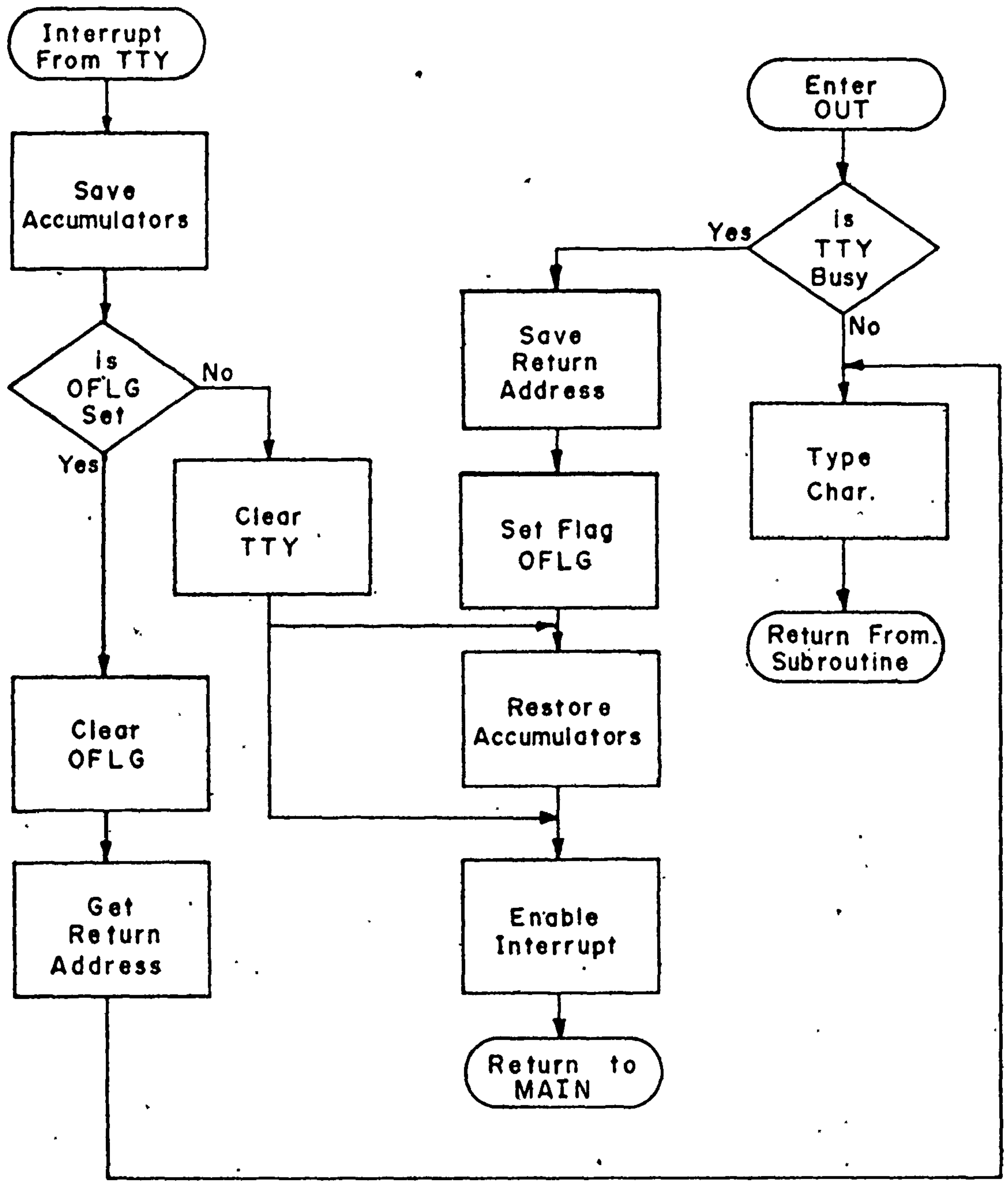


FIG. 5.8

Character Output Flowchart

outputted and the subroutine returns to get the next character. The second character is loaded into accumulator Ø and when OUT is entered the terminal is again tested. If it is busy the subroutine must wait so rather than continually looping and testing the terminal, the subroutine re-enables the interrupt system and waits in MAIN. This allows processing to continue until the terminal is finished and generates an interrupt. This interrupt sends control to the TTY service routine which will output the character and return from the subroutine. After the last character the TTY interrupt will just result in the TTY being cleared and a return being made to MAIN.

The remaining modules to be discussed are the command line decoder and command routines. When the decoder is entered the string of characters making up the command line has been stored in the command buffer. The command itself will be a string of from 1 to 5 characters followed by a space. The decoder then has the task of comparing this string to the entries in the instruction code buffer. This buffer is used to store the instruction names and starting addresses.

When a match is made the decoder jumps to the appropriate command execution routine. The structure of the command decoder allows additional commands to be easily added. All that is required is that an extra entry, consisting of the command name and starting address, must be made to the instruction code buffer and the constant that stores the total number of entries (page 0 location MIN27) must be adjusted.

The command routines themselves are executed with the interrupt system still disabled so they should be as short and as efficient as possible in order to minimize delays in servicing the display and real time clock. The routines can make use of a number of common subroutines that are used to format input and output as well as perform general housekeeping duties. These subroutines are listed in table 1 on page 4 of appendix A. When completed the routines return to MAIN by jumping to RETRN. This partially re-initializes the program to a state where it is waiting for the next command.

The actual structure of the command routine will depend greatly on what functions it performs. A typical example will be illustrated here to describe the general principles.

The command "TYPE I, N, M" is used to type out, in decimal, the counts for channels N to M inclusive of buffer number I. The first thing that the routine does is to get the values of I, N, and M in binary. The numbers had previously been typed in, in decimal, by the user and they are stored as strings of characters in the command buffer. Subroutine GTNUM is used to convert the numbers and return the values in locations INUM, NNUM and MNUM. Next, these numbers are checked for validity using subroutine CHECK. This ensures that buffer I exists and that N and M lie within the range of that buffer. If an error is detected, CHECK will type out the error code and return to RETRN to wait for another command. If there is no error, routine TYPE will then set the output in use flag OUSE. This ensures that no other routine, including the keyboard handler, tries to use the terminal while the formatted data is being outputted. After that TYPE will retrieve the data from the buffer. This is done by first getting the starting address of buffer I from the table at location .BFAD . This is a group of 16 consecutive memory locations that store all of the starting addresses. The Ith entry in this table stores the address of buffer I. There is also a table of buffer sizes that starts at location .BFSØ and is set up in a similar manner except that an additional piece of information is stored here. If bit 0 of entry I is a 1 then buffer I exists and has a non-zero size.

The retrieved data is outputted in lines consisting of a channel number and four data words. Both the channel number and data start out as single precision binary numbers and are converted to decimal by loading the binary into accumulator Ø then jumping to subroutine DCOUT. This routine then types out the string of 5 decimal digits. TYPE separates the numbers with spaces using subroutine OUT and terminates the line with a carriage return and line feed by jumping to subroutine CRLF. When the contents of channel M have finally been outputted control is passed back to MAIN and the interrupt system re-enabled by clearing the output in use flag OUSE and jumping to RETRN.

All of the foregoing modules were assembled into one complete package called NOVADC. The program requires approximately 2500 words at the bottom of memory as well as 100 words at the very top for the binary loader. The remaining memory can be utilized for spectrum buffers. The memory map in figure 5.9 shows the sizes and locations of each of the major modules.

NOVADC was written in a structured and modular fashion to allow easy changes and improvements. The program listing in appendix A gives a description of how to write and add command routines and device service routines. A full understanding of the program, however, will allow the major routines to be used, with small changes, in an experimental setup other than for pulse height analysis. An example that is detailed in appendix B shows how changing 21 instructions allows the system to be used for time interval analysis.

Flexibility and expansion have been the two main criteria in designing and programming the pulse height analyzer. It is hoped that this documentation will allow enough understanding so that the system can be maintained and upgraded to a state never approaching obsolescence.

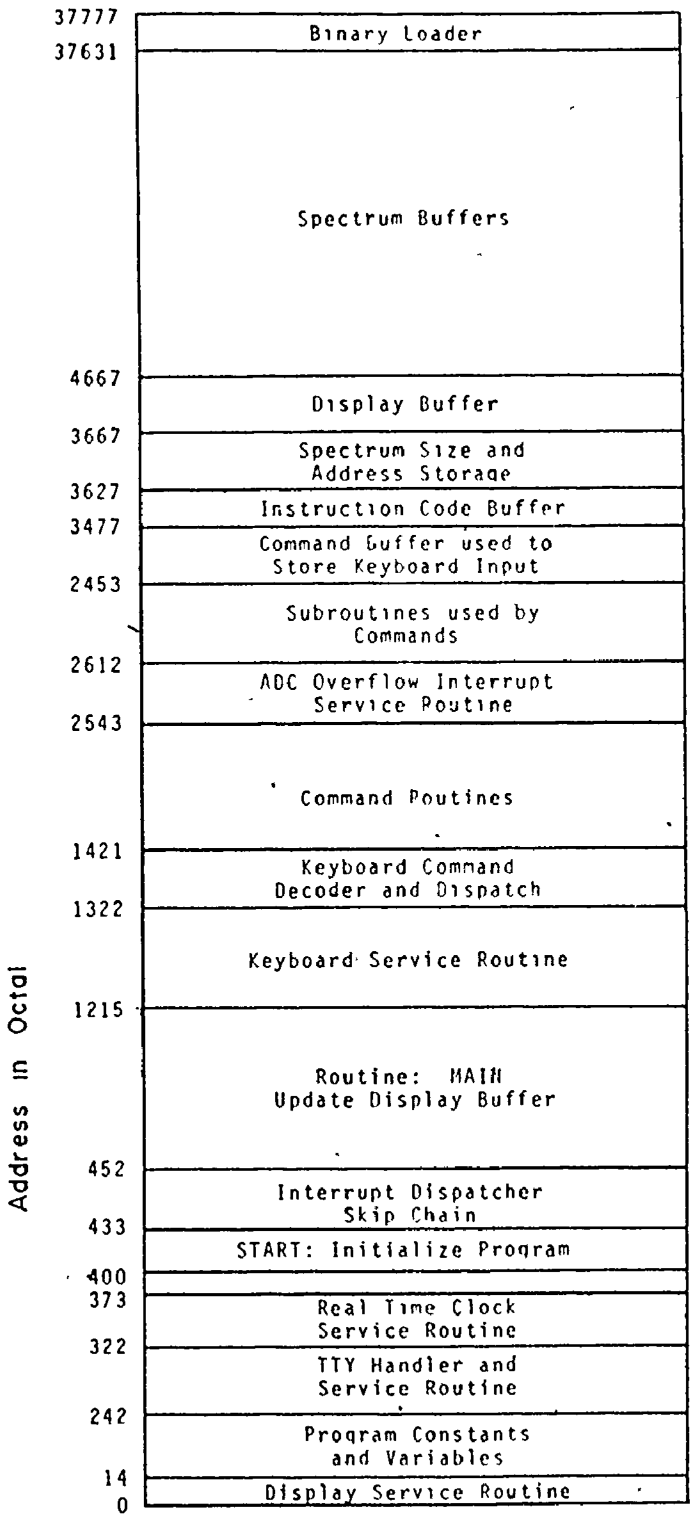

37777
Binary Loader
37631
Spectrum Buffers
4667
Display Buffer
3667
Spectrum Size and Address Storage
3627
Instruction Code Buffer
3477
Command Buffer used to Store Keyboard Input
2453
Subroutines used by Commands
2612
ADC Overflow Interrupt Service Routine
2543
Command Routines
1421
Keyboard Command Decoder and Dispatch
1322
Keyboard Service Routine
1215
Routine: MAIN Update Display Buffer
452
Interrupt Dispatcher Skip Chain
433
START: Initialize Program
400
373
Real Time Clock Service Routine
322
TTY Handler and Service Routine
242
Program Constants and Variables
14
Display Service Routine
0
Address in Octal

FIG. 5.9

NOVADC Memory Map

APPENDIX A

Program Listing

The following is a complete listing of NOVADC version 1C as put out by the NOVA assembler. The source level code for this program is stored on disc in the group's NOVA 2/10 computer system in files NOVADC1.SR, NOVADC2.SR and NOVADC3.SR. These files can be modified using the editor and the program can then be re-assembled. An absolute binary tape is loaded into the PHA using the binary loader. For more detailed information consult the appropriate Data General manuals.

```
0001  .MAIN
              ; NOVADC - PHA INTERFACE CONTROL VERSION 1C
              ; WRITTEN BY G. CORMICK  JUNE, 1975.
              ;                  REV. B  AUG. 1975.
              ;                  REV. C MAY, 1976.
              ;
              ; START AT LOCATION 400
              ; RESTART AT LOC. 377
              ;
              ; COMMANDS:
              ;  I=BUFFER NO. (1 TO 16)
              ;  N=STARTING CHANNEL
              ;  M=ENDING CHANNEL
              ;
              ; TYPE I,N,M   - LISTS CONTENTS OF BUFFER I, CHANNELS N TO M
              ; PUNCH I,N,M  - PUNCHES BINARY WITH LEADER AND TRAILER
              ; READ I,N,M   - READS BINARY PAPER TAPE
              ; CHNGE I,N    - LISTS CHANNEL N. TYPE CR TO DO NOTHING
              ;                NEW NUMBER TO CHANGE.
              ; SUM I,N,M    - SUMS CHANNELS N TO M INCLUSIVE
              ; MAX I        - TYPES CHANNEL WITH THE GREATEST CONTENT
              ; ZERO I/ALL   - SETS TO ZERO ALL CHANNELS IN BUFFER I OR ALL
              ;                CHANNELS IN ALL ACTIVE BUFFERS
              ; CLEAR I/ALL  - REMOVES BUFFER I OR ALL BUFFERS FROM ACTIVE LIST
              ; SIZE I,N     - ADDS BUFFER I OF N CHANNELS TO THE ACTIVE LIST
              ;                N MUST BE A MULTIPLE OF 8
              ; BUFF         - TYPES OUT THE STATUS OF ALL ACTIVE BUFFERS
              ; DISP I       - SETS THE BUFFER TO BE DISPLAYED
              ; STRT I       - STARTS ANALYZING INTO BUFFER I
              ; HALT         - STOPS ANALYZING
              ; CONT         - CONTINUES ANALYZING
              ;
              ;
              ; SPECIAL CHARACTERS:
              ;  ^R - RESTART
              ;  ESC OR ALT MODE - STOP PRESENT COMMAND AND RETURN TO COMMAND
              ;                    INPUT MODE
              ;  RUB OUT - ERASE LAST CHAR(S), SAME AS ESC DURING A READ.
              ;
              ; ERRORS:
              ;  0 - NOT ENOUGH SPACE LEFT IN BUFFER
              ;  1 - ADC IS OFF
              ;  2 - ILLEGAL COMMAND FORMAT
              ;  3 - BUFFER NUMBER OUT OF RANGE
              ;  4 - BUFFER INACTIVE
              ;  5 - CHANNEL NUMBERS INCORRECT
              ;  6 - ILLEGAL NUMBER
              ;  7 - NO PRESENTLY ACTIVE BUFFER
              ;  8 - THE ADC IS ALREADY ANALYZING
              ;  9 - BUFFER ALREADY EXISTS
              ;
              ;
              ; WHILE ANALYZING CHANNEL 1 IS INCREMENTED EVERY 1/10 MIN.
              ; TO GIVE A REAL TIME TOTAL. CHANNEL 2 IS ONLY INCREMENTED
              ; IF THE ADC IS NOT BUSY SO GIVES THE LIVE TIME.
              ; BOTH TIMES ARE IN MINUTES * 10.
              ;
              ;
              ;
              ;
```

```
;
;       SWITCH REGISTER FUNCTIONS
;
;    THE SWITCH REGISTER IS USED TO CONTROL THE DISPLAY BY
; ALLOWING THE USER TO EXPAND, SHIFT OR CHANGE THE MODE OF
; THE DISPLAY BY SETTING THE APPROPRIATE SWITCHES. THE
; FOLLOWING SWITCHES WHEN SET TO 1 (IE. UP) WILL PERFORM
; THE LISTED FUNCTIONS. LEAVING THE SWITCH UP WILL CAUSE
; THE FUNCTION TO BE REPEATED AT AN ACCELERATING RATE.
;
;       SW0 - RESET X & Y SCALES AND POINTER
;       SW1 - HALVE Y SCALE
;       SW2 - DOUBLE Y SCALE
;       SW3 - HALVE X SCALE
;       SW4 - DOUBLE X SCALE
;       SW5 - SHIFT X RIGHT
;       SW6 - SHIFT X LEFT
;       SW7 - GENERATE LOG DISPLAY
;       SW8 - DISPLAY IN HISTOGRAM MODE
;       SW9 - OUTPUT LONG LINES IN TYPE COMMAND
;       SW10 -
;       SW11 - TURN POINTER ON
;       SW12 - SHIFT POINTER LEFT
;       SW13 - SHIFT POINTER RIGHT
;       SW14 - TYPE OUT POINTER CHANNEL NO. AND CONTENTS
;       SW15 -
;
;
;
;
;       ADDING ROUTINES TO THE COMMAND REPERTOIRE
;
;    CERTAIN TYPES OF ROUTINES MAY BE ADDED AT THE SOURCE
; PROGRAM LEVEL TO INCREASE THE CAPABILITIES OF NOVADC. IN
; ORDER TO UNDERSTAND THE LIMITATIONS YOU MUST FIRST UNDER-
; STAND THE BASIC STRUCTURE OF THE PROGRAM.
;.   THE PROGRAM IS SET UP TO USE THE INTERRUPT SYSTEM FOR
; MOST INPUT/OUTPUT OPERATIONS. BOTH THE DISPLAY AND THE
; REAL TIME CLOCK (60 HZ.) REQUEST REGULAR INTERRUPTS SO IT
; IS IMPORTANT TO KEEP THE INTERRUPT ON AS OFTEN AS POSSIBLE.
; THE DISPLAY IN FACT REQUESTS AN INTERRUPT EVERY 63 USEC
; AND REQUIRES 25 USEC FOR SERVICING SO IT IS A MAJOR USER
; OF CPU TIME. WHEN NO INTERRUPTS ARE BEING SERVICED, THE
; SUBPROGRAM MAIN IS BEING EXECUTED AND IT CONTINUALLY
; SCANS THE SWITCH REGISTER AND UPDATES THE DISPLAY BUFFER
; DSBUF, FROM WHICH THE DISPLAY GETS ITS DATA.
;    TO EXECUTE ONE OF THE COMMANDS THE USER TYPES IT INTO
; THE KEYBOARD. THE KEYBOARD INTERRUPTS THE PROGRAM AND
; CONTROL IS TRANSFERRED TO THE KEYBOARD SERVICE ROUTINE
; .TTIN . THIS ROUTINE CHECKS FOR SPECIAL CHARACTERS, ECHOES
; THE CHARACTER AND THEN ADDS THE CHARACTER TO THE COMMAND
; LINE BUFFER .CMB . IF THE CHARACTER WAS A CARRIAGE RETURN,
; THE COMMAND IS INTERPRETED AND CONTROL IS PASSED TO THE
; APPROPRIATE ROUTINE VIA AN ADDRESS WHICH IS STORED IN
; THE INSTRUCTION CODE BUFFER, ALONG WITH THE COMMAND NAME
; ITSELF.
;    ALL OF THE COMMAND INTERPRETING IS DONE WITH THE INTER-
; RUPT OFF, AND THE ROUTINES THEMSELVES ARE EXECUTED WITH
; THE INTERRUPT OFF. THIS SIMPLIFIES THE INTERRUPT HANDLING
; BUT IT REQUIRES THE ROUTINES TO BE AS SHORT AND EFFICIENT
; AS POSSIBLE IN ORDER TO MINIMIZE THE INTERRUPT DOWN TIME.
;    AFTER CONTROL HAS BEEN PASSED TO THE APPROPRIATE ROU-
; TINE, THE COMMAND LINE IS STILL IN THE COMMAND LINE BUFFER
; .CMB . THIS ALLOWS THE ROUTINE TO ACCEPT ARGUMENTS FROM THE
; USER.
;    A NUMBER OF SUBROUTINES ARE AVAILABLE FOR THE EXTRACTION
; OF ARGUMENTS, FOR THE ANALYSIS OF BUFFERS, AND FOR THE OUT-
; PUT OF CHARACTERS AND NUMBERS. (SEE TABLE 1)
;
;    SUBROUTINE GTNUM WILL GET UP TO 3 NUMBERS, I,N,M FROM THE
; COMMAND LINE BUFFER AND WILL PUT THEM IN PAGE 0 LOCATIONS
; INUM, MNUM, NNUM. IN ADDITION I IS ASSUMED TO BE A BUFFER
; NUMBER AND IT IS CHECKED TO SEE IF IT IS VALID AND OPEN FOR
; USE. ANY ERROR WILL CAUSE AN ERROR MESSAGE TO BE PRINTED AND
; A RETURN TO MAIN. IF NO CHECK IS DESIRED, PAGE 0 LOCATION
; CKFLG MUST BE MADE NON-ZERO BEFORE GTNUM IS CALLED. A FURTHER
; ERROR CHECKING SUBROUTINE CAN BE USED. CHECK WILL EXAMINE N
; AND M AND ASSUMING THEY ARE CHANNEL NUMBERS, WILL MAKE SURE
; THAT THEY ARE CORRECT FOR BUFFER I.
;    ALL OUTPUT TO THE TELETYPE OR ALPHANUMERIC DISPLAY IS DONE
; VIA SUBROUTINES. BEFORE OUTPUTTING ANYTHING, THE OUTPUT IN
; USE FLAG MUST BE TESTED AND SET (ISZ OUSE). AFTER OUTPUTTING
; THE FLAG MUST BE CLEARED TO ZERO.
;    SINGLE CHARACTERS CAN BE SENT VIA SUBROUTINE OUT (JSR OUT
; WITH CHAR IN AC0). TEXT INFORMATION CAN BE SENT USING SUB-
; ROUTINE TEXT. TO USE THIS SUBROUTINE THE CHARACTER STRING
; MUST BE PLACED IN THE MESSAGE LIST IN TEXT AND A POINTER TO
; THAT STRING MUST BE PLACED ON PAGE 0. TO CALL THE ROUTINE LOAD
; AC2 WITH THE POINTER AND JSR @TEXT .
;    NUMBERS CAN BE OUTPUTTED IN SINGLE PRECISION DECIMAL OR
; SINGLE PRECISION OCTAL USING THE APPROPRIATE CALL, JSR @DCOUT,
; OR JSR @BINO. NUMBERS CAN ALSO BE OUTPUTTED IN DOUBLE PRECISIO
; DECIMAL IF DCOUT IS FLAGGED BY SWITCHING THE CONTENTS OF LOC-
; ATIONS DBD AND BIND, AND LOADING AC0 WITH THE HIGH ORDER BINAR
; AND AC2 WITH THE LOW ORDER BINARY AND CALLING DCOUT (JSR
; @DCOUT). LOCATIONS DBD AND BIND MUST BE RESET BEFORE OUTPUT-
; TING ANY SINGLE PRECISION NUMBERS. FOR AN EXAMPLE SEE ROUTINE
; SUM.
;
;    AFER COMPLETING THE ROUTINE CONTROL IS PASSED BACK TO MAIN
; BY JUMPING TO LOCATION 377 (JMP RETRN).
;
;
;    THE PROCEDURE FOR ADDING A ROUTINE IS AS FOLLOWS:
;
; 1) THE COMMAND NAME (5 CHARACTERS OR LESS) MUST BE ADDED TO
;    THE INSTRUCTION CODE BUFFER USING THE .TXT PSEUDO -OP.
;    IF THE COMMAND NAME IS LESS THAN 4 CHARACTERS LONG THE
;    NAME MUST BE PADDED WITH EXTRA ZEROS TO ENSURE THAT EXAC-
;    TLY 3 LOCATIONS ARE RESERVED FOR THE NAME. THE NAME IS
;    THEN FOLLOWED BY A WORD CONTAINING THE STARTING ADDRESS OF
;    YOUR ROUTINE.
;
; 2) PAGE 0 LOCATION MIN2T MUST BE SET EQUAL TO THE 2'S COMP-
;    LEMENT OF THE NUMBER OF NAMES IN THE INSTRUCTION CODE
;    BUFFER.
;
; 3) INSERT YOUR ROUTINE AFTER THE EXISTING ROUTINES.
```

```
                        ;
                        ;
                        ;    THE PROCEDURE FOR ADDING A DEVICE HANDLER IS:
                        ;
                        ; 1) ADD THE APPROPRIATE SKPDN INSTRUCTION AND JMP INSTRUCTION
                        ;    TO THE SKIP CHAIN (INTR).
                        ;
                        ; 2) PUT A POINTER TO YOUR HANDLER ON PAGE 0.
                        ;
                        ; 3) INSERT YOUR HANDLER INTO THE PROGRAM AFTER THE ADC OVER-
                        ;    FLOW HANDLER .OVER .
                        ;
                        ;    DEVICE HANDLERS SHOULD BE SET UP SO THAT THEY CANNOT BE
                        ; INTERRUPTED. (IE. ONLY 1 PRIORITY LEVEL OF INTERRUPT). ANY
                        ; WAITING SHOULD BE DONE IN MAIN WITH THE INTERRUPT ON. AFTER
                        ; AN INTERRUPT, THE HANDLER CAN USE LOCATIONS TAC0, TAC1, TAC2,
                        ; AND TAC3 TO SAVE THE ACCUMULATORS.
                        ;
                        ;
                        ;                    TABLE 1
                        ;
                        ; SUBROUTINE       ENTRY              USE
                        ;
                        ;  TEXT        JSR @TEXT       OUTPUT ASCII STRING TO TTO
                        ;              AC2=STARTING
                        ;                  ADDRESS
                        ;
                        ;  ERROR       JSR @ERR        TYPE OUT ERROR ¤
                        ;              AC0=ERROR NO.
                        ;
                        ;  TOTAL       JSR @TOTAL      ADDS UP THE BUFFER SIZES
                        ;                              AC0=TOTAL
                        ;
                        ;  GTNUM       JSR @GTNUM      EXTRACTS 3 NUMBERS FROM THE
                        ;                              COMMAND LINE INUM, MNUM, NNUM
                        ;
                        ;  DCOUT       JSR @DCOUT      OUTPUTS A BINARY NUMBER IN
                        ;              AC0=BINARY NO.  DECIMAL TO THE TTO
                        ;
                        ;  TAB         JSR @TAB        TYPES N SPACES ON THE TTO
                        ;              AC0=N
                        ;
                        ;  CHECK       JSR @CHECK      TYPES ERROR 5 AND RETURNS TO
                        ;                              MAIN IF MNUM,NNUM NOT CORRECT
                        ;
                        ;  BINO        JSR @BINO       OUTPUTS AC1 IN OCTAL TO THE
                        ;              AC1 = NO.       TTO
                        ;
                        ;
```

```
       000000 .LOC 0
00000 000453 RETM:  MAIN
00001 000434        INTR        ; ADDRESS OF INTERRUPT SERVICE
              ;
              ; DISPLAY INTERRUPT SERVICE ROUTINE
              ;
00002 050047 DSOUT: STA 2,TAC2   ; SAVE AC'S
00003 054050        STA 3,TAC3
00004 074436        DIA 3,36     ; GET X-COORD
00005 030226        LDA 2,.DSBF  ; GET BUFFER ADDRESS
00006 157000        ADD 2,3
00007 031400        LDA 2,0,3
00010 071136        DOAS 2,36    ; OUTPUT Y-COORD
00011 030047        LDA 2,TAC2   ; RESTORE AC'S
00012 034050        LDA 3,TAC3
00013 060177        INTEN        ; TURN ON INTERRUPT
00014 002000        JMP @0       ; RETURN
              ;
              ;
              ;
              ;  PROGRAM CONSTANTS AND STORAGE LOCATIONS
              ;
       000040        .LOC 40
00040 003112 .GET:  GET
00041 003130 .PUT:  PUT
00042 000000 OFLG:  0
00043 000000 OADR:  0
00044 000000 ORET:  0
00045 000000 TAC0:  0           ; TEMPORARY SAVE FOR AC'S
00046 000000 TAC1:  0
00047 000000 TAC2:  0
00050 000000 TAC3:  0
00051 003454 CMBUF: .CMB
00052 000000 CMCNT: 0
00053 000000 INUM:  0
00054 000000 NNUM:  0
00055 000000 MNUM:  0
00056 000000 CKFLG: 0
00057 000000 OUSE:  0          ; OUTPUT IN USE FLAG
00060 000000 ZFLAG: 0
00061 000000 ACTIV: 0      ; LAST BUFFER THAT WAS ANALYZED INTO
00062 004677 BUFST: .BFST
00063 037631 BUFEN: 37631      ; LAST AVAILABLE MEMORY ADDRESS
00064 000000 ANLYZ: 0      ; CURRENT BUFFER THAT IS COLLECTING DATA
00065 177740 M40:   -40
00066 000000 CNT1:  0
00067 000000 CNT2:  0
00070 000000 CNT3:  0
00071 000215 CR:    215
00072 000212 LF:    212
00073 000052 AST:   "*
00074 000300 TTOUT: .TOUT
00075 001216 TTIN:  .TTIN
00076 002544 OVER:  .OVER
00077 002631 MES1:  M1         ; TEXT POINTER LIST
00100 002636 MES2:  M2
00101 002656 MES3:  M3
00102 002663 MES4:  M4
```

```
0006   .MAIN
 00103 002672 MES5:   M5
 00104 002677 MES6:   M6
 00105 002730 MES7:   M7
 00106 000222 CNTLR:  222
 00107 000233 ESC:    233
 00110 000375 ALT:    375
 00111 000377 RUB:    377
 00112 000230 CAN:    230
 00113 000177 MASK1:  177
 00114 000136 CNTL:   "^
 00115 000122 R:      "R
 00116 000400 ST:     START
 00117 002760 TOTAL:  .TOTL
 00120 002613 TEXT:   .TEXT
 00121 000101 A:      "A
 00122 000114 L:      "L
 00123 000000 SZE:    0
 00124 000053 PLUS:   "+
 00125 003204 BIND:   .BIND
 00126 003250 DBD:    .DBD
 00127 000000 TOTL1:  0
 00130 000000 TOTL2:  0
 00131 002774 GTNUM:  .GETN
 00132 000134 BKSL:   "\
 00133 000020 P20:    20
 00134 000024 P24:    24
 00135 002746 ERR:    ERROR
 00136 003500 ASCII:  .ASCI
 00137 177774 MIN4:   -4
 00140 177773 MIN5:   -5
 00141 177760 MIN16:  -20
 00142 000040 SPACE:  40
 00143 177751 MIN27:  -27          ; LENGTH OF INSTRUCTION CODE BUFFER
 00144 000002 P2:     2
 00145 000003 P3:     3
 00146 000006 P6:     6
 00147 000005 P5:     5
 00150 000004 P4:     4
 00151 000010 P8:     10
 00152 000011 P9:     11
 00153 000012 P10:    12
 00154 000014 P12:    14
 00155 000017 P15:    17
 00156 000030 POS24:  30
 00157 000032 POS26:  32
 00160 000074 P60:    74
 00161 001000 P512:   1000
 00162 003630 BUFSZ:  .BFSZ
 00163 003650 BUFAD:  .BFAD
 00164 000000 BADR:   0
 00165 177770 MIN8:   -10
 00166 003154 TAB:    .TAB
 00167 000017 MASK2:  17
 00170 000077 MASK3:  77
 00171 077014 MASK4:  077014
 00172 000400 MASK5:  400
 00173 000100 MASK6:  100
 00174 001777 MASK7:  1777
 00175 000200 BIT8:   200
```

```
0007  .MAIN
 00176 000360 P560:   360          ; LENGTH OF LEADER
 00177 003164 CHECK:  .CHK
 00200 003172 .CKL2:  CHKL2
 00201 003120 DCOUT:  .DCOT
 00202 003361 DBIN:   .DBIN
 00203 000060 AS0:    "0
 00204 000071 AS9:    "9
 00205 000054 COMA:   ",
 00206 000000 ASC1:   0
 00207 000000 ASC2:   0
 00210 000000 ASC3:   0
 00211 000000 TAC00:  0
 00212 000000 TAC01:  0
 00213 000000 TAC02:  0
 00214 000000 TAC03:  0
 00215 003427 BINO:   .BINO
 00216 000000 YSCL:   0
 00217 000020 DCINT:  20
 00220 000000 XSCL:   0
 00221 000020 DCNT:   20
 00222 000000 XSHFT:  0
 00223 000001 DCFLG:  1
 00224 000000 DISB:   0
 00225 000000 DISBF:  0
 00226 003670 .DSBF:  DSBUF
 00227 000000 DISSZ:  0
 00230 000000 PNTA:   0
 00231 000000 DSADR:  0
 00232 000000 DLOOP:  0
 00233 000000 DSLOP:  0
 00234 000000 DSPAD:  0
 00235 000000 DSNUM:  0
 00236 000020 CLKNT:  20          ; RESET CONSTANT FOR CLOCK COUNTERS
 00237 001046 .YPNT:  YPONT
 00240 000000 RUNT:   0
 00241 000550 P550:   550         ; LOOP COUNTER FOR REAL TIMER
 00242 000453 .MAIN:  MAIN
                    ;
                    ;
                    ; SUBROUTINE OUT
                    ;
 00243 040276 OUT:    STA   0,CHAR
 00244 054277         STA   3,OUTR
 00245 063411         SKPBN TTO       ; IS TTO BUSY
 00246 000272         JMP TTTY        ; NO SO TYPE
 00247 020275         LDA 0,ECHO      ; YES SO WAIT IN MAIN FOR DONE
 00250 040043         STA 0,OADR
 00251 010042         ISZ OFLG
 00252 040211         STA 0,TAC00     ; SAVE AC'S
 00253 044212         STA 1,TAC01
 00254 050213         STA 2,TAC02
 00255 054214         STA 3,TAC03
 00256 020045 RESTR:  LDA 0,TAC0      ; RESTORE AC'S
 00257 024046         LDA 1,TAC1
 00260 030047         LDA 2,TAC2
 00261 034050         LDA 3,TAC3
 00262 060177         INTEN
 00263 002000         JMP @RETM
 00264 126420 .ECHO:  SUBZ 1,1
```

```
0008  .MAIN
 00265 044042          STA 1.OFLG
 00266 020211          LDA 0.TACO0
 00267 024212          LDA 1.TACO1
 00270 030213          LDA 2.TACO2
 00271 034214          LDA 3.TACO3
 00272 020276 TTTY:    LDA 0.CHAR
 00273 061111          DOAS 0.TTO    ; OUTPUT CHAR
 00274 002277          JMP @OUTR
 00275 000264 ECHO:    .ECHO
 00276 000000 CHAR:    0
 00277 000000 OUTR:    0
              ;
              ;
 00300 040045 .TOUT:   STA 0.TAC0    ; SAVE ACCUMULATORS
 00301 044046          STA 1.TAC1
 00302 050047          STA 2.TAC2
 00303 054050          STA 3.TAC3
 00304 020042          LDA 0.OFLG    ; CHECK SOFTWARE FLAG
 00305 101005          MOV 0.0.SNR
 00306 000312          JMP CLTTO
 00307 102400          SUB 0.0       ; OUT ROUTINE IN USE
 00310 040042          STA 0.OFLG
 00311 002043          JMP @OADR
 00312 060211 CLTTO:   NIOC TTO
 00313 000256          JMP RESTR     ; RETURN
              ;
              ; SUBROUTINE CRLF OUTPUTS A CARRIAGE RETURN AND A LINE FEED
              ;
 00314 054322 CRLF:    STA 3.CL      ; SAVE RETURN ADDRESS
 00315 020071          LDA 0.CR
 00316 004243          JSR OUT
 00317 020072          LDA 0.LF
 00320 004243          JSR OUT
 00321 002322          JMP @CL
 00322 000000 CL:      0
              ;
              ;
              ; REAL TIME CLOCK INTERRUPT SERVICE ROUTINE
              ;
 00323 040045 RTCLK:   STA 0.TAC0    ; SAVE AC0.AC1
 00324 044046          STA 1.TAC1
 00325 020064          LDA 0.ANLYZ
 00326 101005          MOV 0.0.SNR   ; IS ADC BUSY
 00327 000346          JMP RTCDS     ; UPDATE DISPLAY COUNTERS
 00330 014240 RTCL4:   DSZ RUNT      ; INCREMENT ADC TIMER
 00331 000346          JMP RTCDS
 00332 050047          STA 2.TAC2
 00333 030163          LDA 2.BUFAD
 00334 113000          ADD 0.2
 00335 031000          LDA 2.0.2
 00336 011001          ISZ 1.2       ; INCREMENT CHAN 1
 00337 000340          JMP .+1
 00340 063437          SKPBN 37      ; SKP IF ADC DEAD
 00341 011002          ISZ 2.2       ; INCREMENT CHAN 2
 00342 000343          JMP .+1
 00343 020241          LDA 0.P550
 00344 040240          STA 0.RUNT
 00345 030047          LDA 2.TAC2
 00346 060477 RTCDS:   READS 0       ; UPDATE DISPLAY
```

```
0009  .MAIN
 00347 024171          LDA 1.MASK4
 00350 123404          AND 1.0.SZR  ;SKP IF NO DISPLAY CHANGES
 00351 000363          JMP RTUPD
 00352 102520          SUBZL 0.0    ; GET +1
 00353 040221          STA 0.DCNT
 00354 020236          LDA 0.CLKNT  ; GET UPDATE COUNTER
 00355 040217          STA 0.DCINT
 00356 020045 RTCRT:   LDA 0.TAC0
 00357 024046          LDA 1.TAC1
 00360 060114          NIOS RTC     ; RESTART CLOCK
 00361 060177          INTEN
 00362 002000          JMP @0       ; RETURN
 00363 014221 RTUPD:   DSZ DCNT     ; DECREMENT UPDATE COUNTER
 00364 000356          JMP RTCRT
 00365 010223          ISZ DCFLG    ; SET UPDATE FLAG
 00366 014217          DSZ DCINT    ; REDUCE LOOP COUNT
 00367 000371          JMP .+2
 00370 010217          ISZ DCINT
 00371 020217          LDA 0.DCINT
 00372 040221          STA 0.DCNT
 00373 000356          JMP RTCRT
              ;
              ;
              ;
       000377 .LOC 377
 00377 000417 RETRN:   JMP RTRN     ; RESTART ADDRESS
 00400 062677 START:   IORST        ; RESET HARDWARE FLAGS
 00401 060114          NIOS RTC     ; RESTART RTC
 00402 060136          NIOS 36       ; RESTART DISPLAY
 00403 004314          JSR CRLF
 00404 030104          LDA 2.MES6
 00405 006120          JSR @TEXT    ; OUTPUT TITLE
 00406 004314          JSR CRLF
 00407 102400          SUB 0.0
 00410 040042          STA 0.OFLG   ; CLEAR SOFTWARE FLAGS
 00411 040060          STA 0.ZFLAG
 00412 040061          STA 0.ACTIV
 00413 040064          STA 0.ANLYZ
 00414 020242          LDA 0..MAIN
 00415 040000          STA 0.0.0
              ;
              ;
 00416 102400 RTRN:    SUB 0.0      ; CLEAR COMMAND BUFFER
 00417 040052          STA 0.CMCNT  ; AND SOFTWARE FLAGS
 00420 024134          LDA 1.P24    ; AND RETURN TO THE
 00421 044067          STA 1.CNT2   ; MAIN PROGRAM
 00422 030051          LDA 2.CMBUF
 00423 041000 LOOP2:   STA 0.0.2
 00424 151400          INC 2.2
 00425 014067          DSZ CNT2
 00426 000775          JMP LOOP2
 00427 040057          STA 0.OUSE
 00430 004314          JSR CRLF
 00431 020073          LDA 0.AST
 00432 004243          JSR OUT
 00433 000256          JMP RESTR    ; RESTORE AC'S AND RETURN
              ;
              ;
              ;
```

```
0010  .MAIN
 00434 063736 INTR:   SKPDZ 36      ; INTERRUPT SKIP CHAIN
 00435 000002         JMP DSOUT     ; SERVICE DISPLAY
 00436 063714         SKPDZ RTC
 00437 000323         JMP RTCLK     ; SERVICE REAL TIME CLOCK
 00440 063711         SKPDZ TTO
 00441 002074         JMP @TTOUT    ; SERVICE TTO
 00442 063710         SKPDZ TTI
 00443 002075         JMP @TTIN     ; SERVICE TTI
 00444 063737         SKPDZ 37
 00445 002076         JMP @OVER     ; SERVICE ADC OVERFLOW
 00446 062677         IORST         ; UNKNOWN INTERRUPT
 00447 060114         NIOS RTC      ; RESTART RTC
 00450 060136         NIOS 36       ; RESTART DISPLAY
 00451 002401         JMP @.CONT    ; CONTINUE ANALYZING
 00452 002047 .CONT:  CONT
                ;
                ;
                ; ROUTINE TO UPDATE THE CONTENTS OF THE DISPLAY BUFFER
                ;
                ;
 00453 060477 MAIN:   READS 0
 00454 101133         MOVZL# 0,0,SNC ; CHECK BIT 0 FOR RESET
 00455 000412         JMP MNL2
 00456 102440         SUBO 0,0      ; RESET X,Y
 00457 040216         STA 0,YSCL
 00460 040220         STA 0,XSCL
 00461 040222         STA 0,XSHFT
 00462 040223         STA 0,DCFLG
 00463 040230         STA 0,PNTA
 00464 020236 MNL1:   LDA 0,CLKNT
 00465 040217         STA 0,DCINT
 00466 000554         JMP UPDAT     ; REFILL THE BUFFER
 00467 024223 MNL2:   LDA 1,DCFLG   ; IS UPDATE FLAG SET
 00470 125005         MOV 1,1,SNR   ; BY RTC
 00471 000551         JMP UPDAT
 00472 126440         SUBO 1,1       ; YES SO CHANGE SCALES ETC.
 00473 044223         STA 1,DCFLG   ; RESET FLAG
 00474 024171         LDA 1,MASK4
 00475 107405         AND 0,1,SNR   ; ANY CHANGES?
 00476 000766         JMP MNL1      ; NO
 00477 101120         MOVZL 0,0     ; YES - SCAN SWITCH REG.
 00500 101123         MOVZL 0,0,SNC ; SKP IF SW1 = 1
 00501 000406         JMP MNL4
 00502 024216         LDA 1,YSCL    ; HALVE Y-SCALE
 00503 030152         LDA 2,P9
 00504 133004         ADD 1,2,SZR   ; SKP IF YSCL=-11
 00505 014216         DSZ YSCL
 00506 000401         JMP .+1
 00507 101123 MNL4:   MOVZL 0,0,SNC ; SKP IF SW2 = 1
 00510 000406         JMP MNL6
 00511 024216         LDA 1,YSCL
 00512 030152         LDA 2,P9
 00513 132404         SUB 1,2,SZR   ; SKP IF YSCL=+11
 00514 010216         ISZ YSCL
 00515 000401         JMP .+1
 00516 101123 MNL6:   MOVZL 0,0,SNC ; SKP IF SW3 = 1
 00517 000417         JMP MNL8
 00520 024227         LDA 1,DISSZ   ; GET BUFFER SIZE
                      SUB           ;
```

```
0011  .MAIN
 00522 125242          MOVOR 1,1,SZC
 00523 000403          JMP .+3
 00524 151400          INC 2,2       ; INCREMENT LOG2(SIZE)
 00525 000775          JMP .-3
 00526 024155          LDA 1,P15
 00527 146400          SUB 2,1
 00530 030220          LDA 2,XSCL
 00531 034146          LDA 3,P6
 00532 173000          ADD 3,2
 00533 146023          ADCZ 2,1,SNC ; SKP IF XSCL TOO SMALL
 00534 014220          DSZ XSCL      ; REDUCE X-SCALE
 00535 000401          JMP .+1
 00536 101123 MNL8:    MOVZL 0,0,SNC ; SKP IF SW4 - 1
 00537 000410          JMP MNL10
 00540 024220          LDA 1,XSCL
 00541 030150          LDA 2,P4
 00542 147000          ADD 2,1
 00543 030154          LDA 2,P12
 00544 146423          SUBZ 2,1,SNC
 00545 010220          ISZ XSCL      ; EXPAND X-SCALE
 00546 000401          JMP .+1
 00547 101123 MNL10:   MOVZL 0,0,SNC ; SKP IF SW5 = 1
 00550 000411          JMP MNL12
 00551 030222          LDA 2,XSHFT   ; REDUCE XSHFT BY
 00552 151005          MOV 2,2,SNR   ; 8 UNLESS IT =0
 00553 000406          JMP MNL12
 00554 024165          LDA 1,MIN8
 00555 125405          INC 1,1,SNR
 00556 000403          JMP MNL12
 00557 014222          DSZ XSHFT
 00560 000775          JMP .-3
 00561 101123 MNL12:   MOVZL 0,0,SNC ; SKP IF SW6 = 1
 00562 000432          JMP MNL14
 00563 024220          LDA 1,XSCL
 00564 030227          LDA 2,DISSZ
 00565 034157          LDA 3,POS26   ; ALLOW FOR THE BLANK 30
 00566 173000          ADD 3,2       ; CHANNELS ON THE DISPLAY
 00567 034161          LDA 3,P512
 00570 125113          MOVL# 1,1,SNC
 00571 000405          JMP .+5
 00572 175120          MOVZL 3,3     ; XSCL IS -VE
 00573 125405          INC 1,1,SNR
 00574 000410          JMP MNL13
 00575 000775          JMP .-3
 00576 124405          NEG 1,1,SNR   ; XSCL IS +VE
 00577 000405          JMP MNL13     ; XSCL IS 0
 00600 034161          LDA 3,P512
 00601 175220          MOVZR 3,3
 00602 125404          INC 1,1,SZR
 00603 000776          JMP .-2
 00604 024222 MNL13:   LDA 1,XSHFT
 00605 167000          ADD 3,1
 00606 146422          SUBZ 2,1,SZC ; SKP IF NOT AT END
 00607 000405          JMP MNL14     ; OF BUFFER
 00610 024165          LDA 1,MIN8
 00611 010222          ISZ XSHFT     ; SHIFT BUFFER
 00612 125404          INC 1,1,SZR
 00613 000776          JMP .-2
 00614 101300 MNL14:   MOVS 0,0
```

```
0012  .MAIN
 00615 101220          MOVZR 0.0
 00616 101223          MOVZR 0.0.SNC ; SKP IF SW13 = 1
 00617 000411          JMP MNL16
 00620 030137          LDA 2.MIN4
 00621 034161          LDA 3.P512
 00622 024230          LDA 1.PNTA
 00623 166432          SUBZ# 3.1.SZC ; INCREMENT POINTER
 00624 000404          JMP MNL16     ; ADDRESS UNLESS IT
 00625 010230          ISZ PNTA      ; IS >= 512
 00626 151404          INC 2.2.SZR
 00627 000773          JMP .-5
 00630 101223 MNL16:   MOVZR 0.0.SNC ; SKP IF SW12 = 1
 00631 000411          JMP UPDAT
 00632 030137          LDA 2.MIN4
 00633 024230          LDA 1.PNTA
 00634 125005          MOV 1.1.SNR
 00635 000405          JMP UPDAT
 00636 151405          INC 2.2.SNR
 00637 000403          JMP UPDAT
 00640 014230          DSZ PNTA       ; SHIFT POINTER LEFT
 00641 000775          JMP .-3
 00642 020225 UPDAT:   LDA 0.DISBF   ; GET 1ST ADDRESS
 00643 024222          LDA 1.XSHFT
 00644 123000          ADD 1.0
 00645 040231          STA 0.DSADR
 00646 024161          LDA 1.P512
 00647 044232          STA 1.DLOOP
 00650 030226          LDA 2..DSBF   ; GET DISPLAY BUFFER ADDRESS
 00651 050234          STA 2.DSPAD
 00652 024220          LDA 1.XSCL
 00653 125133          MOVZL# 1.1.SNC ; SKP IF XSCL<0
 00654 125005          MOV 1.1.SNR   ; SKP IF XSCL NOT= 0
 00655 000426          JMP MNL40     ; XSCL <= 0
 00656 124400          NEG 1.1       ; XSCL IS +VE
 00657 152520          SUBZL 2.2     ; SET AC2 = 1
 00660 151120          MOVZL 2.2     ; GET THE NUMBER OF
 00661 125404          INC 1.1.SZR   ; DISPLAY WORDS PER
 00662 000776          JMP .-2       ; CHANNEL
 00663 050235          STA 2.DSNUM
 00664 022231 MNL20:   LDA 0.@DSADR ; GET CHANNEL CONTENTS
 00665 006237          JSR @.YPNT    ; GET Y VALUE
 00666 030235          LDA 2.DSNUM
 00667 150400          NEG 2.2
 00670 050233          STA 2.DSLOP
 00671 004452 MNL22:   JSR POINT
 00672 042234          STA 0.@DSPAD ; PUT Y IN DISPLAY
 00673 010234          ISZ DSPAD     ; DSNUM TIMES
 00674 014232          DSZ DLOOP
 00675 000402          JMP .+2
 00676 000462          JMP PNTR
 00677 010233          ISZ DSLOP
 00700 000771          JMP MNL22
 00701 010231          ISZ DSADR
 00702 000762          JMP MNL20
 00703 152520 MNL40:   SUBZL 2.2     ; GET THE NUMBER OF
 00704 155000          MOV 2.3       ; CHANNELS PER DISPLAY
 00705 125400          INC 1.1       ; WORD
 00706 136415          SUB# 1.3.SNR
 00707 000403          JMP MNL42
```

```
0013  .MAIN .
 00710 151120          MOVZL 2,2
 00711 000774          JMP .-4
 00712 050235 MNL42:   STA 2,DSNUM
 00713 102400 MNL44:   SUB 0,0
 00714 176400          SUB 3,3
 00715 030235          LDA 2,DSNUM
 00716 150400          NEG 2,2
 00717 026231          LDA 1,@DSADR
 00720 123022          ADDZ 1,0,SZC ; ADD UP DSNUM
 00721 175400          INC 3,3      ; CHANNELS
 00722 010231          ISZ DSADR
 00723 151404          INC 2,2,SZR
 00724 000774          JMP .-4
 00725 024220          LDA 1,XSCL
 00726 125005          MOV 1,1,SNR
 00727 000405          JMP MNL46
 00730 175220          MOVZR 3,3    ; DIVIDE SUM BY
 00731 101200    )     MOVR 0,0     ; DSNUM FOR AVERAGE
 00732 125404          INC 1,1,SZR
 00733 000775          JMP .-3
 00734 006237 MNL46:   JSR @.YPNT
 00735 004406          JSR POINT
 00736 042234          STA 0,@DSPAD
 00737 010234          ISZ DSPAD
 00740 014232          DSZ DLOOP
 00741 000752          JMP MNL44
 00742 000416          JMP PNTR
 00743 064477 POINT:   READS 1
 00744 030133          LDA 2,P20
 00745 147405          AND 2,1,SNR    ; SKIP IF POINTER ON
 00746 001400          JMP 0,3
 00747 024226          LDA 1,.DSBF
 00750 030230          LDA 2,PNTA
 00751 133000          ADD 1,2
 00752 024234          LDA 1,DSPAD    ; IS THIS LOC THE ONE
 00753 132414          SUB# 1,2,SZR   ; THAT IS BEING POINTED TO
 00754 001400          JMP 0,3
 00755 101120          MOVZL 0,0      ; YES SO SET BIT 0
 00756 101240          MOVOR 0,0
 00757 001400          JMP 0,3
 00760 024144 PNTR:    LDA 1,P2
 00761 060477          READS 0
 00762 107404          AND 0,1,SZR  ; IS SW14 SET
 00763 000404          JMP MNL49
 00764 102400          SUB 0,0      ; NO SO CLEAR POINTER
 00765 040460          STA 0,PNTF   ; TYPE FLAG
 00766 002242          JMP @.MAIN
 00767 020057 MNL49:   LDA 0,OUSE   ; YES SO TYPE OUT
 00770 101004          MOV 0,0,SZR  ; CHANNEL NO. IF
 00771 002242          JMP @.MAIN   ; OUTPUT NOT BUSY
 00772 020453          LDA 0,PNTF   ; AND POINTER TYPE
 00773 101004          MOV 0,0,SZR  ; FLAG IS NOT SET
 00774 002242          JMP @.MAIN
 00775 010450          ISZ PNTF     ; SET FLAG
 00776 010057          ISZ OUSE
 00777 030230          LDA 2,PNTA
 01000 024220          LDA 1,XSCL
 01001 125113          MOVL# 1,1,SNC
 01002 000405          JMP .+5
```

```
0014  .MAIN
 01003 151120          MOVZL 2.2      ; XSCL IS -VE
 01004 125405          INC 1.1.SNR
 01005 000407          JMP MNL50
 01006 000775          JMP .-3
 01007 124405          NEG 1.1.SNR     ; XSCL IS +VE
 01010 000404          JMP MNL50       ; XSCL IS 0
 01011 151220          MOVZR 2.2
 01012 125404          INC 1.1.SZR
 01013 000776          JMP .-2
 01014 024222 MNL50:   LDA 1.XSHFT
 01015 147000          ADD 2.1         ; COMPUTE CHANNEL NO.
 01016 030163          LDA 2.BUFAD
 01017 020224          LDA 0.DISB
 01020 113000          ADD 0.2         ; GET ADDRESS OF BUFFER
 01021 035000          LDA 3.0.2
 01022 137000          ADD 1.3
 01023 021400          LDA 0.0.3       ;GET CHANNEL CONTENTS
 01024 044231          STA 1.DSADR
 01025 060277          INTDS
 01026 030416          LDA 2..WAIT     ; SET UP INTERRUPT
 01027 050000          STA 2.0.0       ; WAITING LOOP
 01030 006201          JSR @DCOUT      ; OUTPUT CONTENTS
 01031 030103          LDA 2.MES5
 01032 006120          JSR @TEXT       ; TYPE "AT CHAN."
 01033 020231          LDA 0.DSADR
 01034 006201          JSR @DCOUT      ; OUTPUT CHANNEL
 01035 004314          JSR CRLF
 01036 102400          SUB 0.0
 01037 040057          STA 0.OUSE
 01040 060177          INTEN
 01041 002242          JMP @.MAIN
 01042 000401 WAIT:    JMP .+1
 01043 000777          JMP .-1
 01044 001042 .WAIT:   WAIT
 01045 000000 PNTF:    0
                ;
                ; ROUTINE TO CALCULATE Y DISPLAYED
                ;
 01046 054432 YPONT:   STA 3.YPRET    ; SAVE RETURN
 01047 064477          READS 1
 01050 030172          LDA 2.MASK5
 01051 133404          AND 1.2.SZR     ; IS SW7 = 1
 01052 000427          JMP LOG         ; YES SO LOG DISPLAY
 01053 024216          LDA 1.YSCL      ; NO SO LINEAR
 01054 125113          MOVL# 1.1.SNC ; SKP I YSCL <0
 01055 000405          JMP YPL2
 01056 101220          MOVZR 0.0       ; DIVIDE Y BY 2
 01057 125404          INC 1.1.SZR
 01060 000776          JMP .-2
 01061 000406          JMP YPSET
 01062 124405 YPL2:    NEG 1.1.SNR
 01063 000404          JMP YPSET       ; YSCL =0
 01064 101120          MOVZL 0.0     ; YSCL IS +VE
 01065 125404          INC 1.1.SZR
 01066 000776          JMP .-2
 01067 030174 YPSET:   LDA 2.MASK7      ; TRUNCATE TO 10 BITS
 01070 143400          AND 2.0
 01071 030175          LDA 2.BIT8
 01072 064477          READS 1
```

```
0015  .MAIN
 01073 133405          AND 1.2.SNR    ; IS SW8 ON
 01074 002404          JMP @YPRET
 01075 101100          MOVL 0.0       ; YES SO HISTOGRAM
 01076 101240          MOVOR 0.0      ; SET BIT 0
 01077 002401          JMP @YPRET
 01100 000000 YPRET: 0
              ;
              ;
              ; ROUTINE TO GENERATE LOG FUNCTION FOR
              ; LOG DISPLAY
              ;
              ; THIS ROUTINE USES THE APPROXIMATION:
              ;    LOG10(X) = C1*Z + C3*Z^3 + C5*Z^5
              ;
              ;       WHERE:     Z = (X-1)/(X+1)
              ;                  C1 = .8690286
              ;                  C3 = .2773839
              ;                  C5 = .2543275
              ;
              ;       AND:       1/SQR(10)  <= X <=  SQR(10)
              ;
              ;    LOG2(X) = LOG2(10)*LOG10(X)
              ;
              ; X IS FIRST NORMALIZED TO THE RANGE 1 TO 2
              ; BY SHIFTING LEFT AND COUNTING SHIFTS
              ; NOTE: THE POSITION OF THE BINARY POINT
              ;       CHANGES IN ORDER TO MAINTAIN
              ;       ACCURACY WITH SINGLE PRECISION
              ;       BUT IT IS USUALLY NEAR THE LEFT
              ;       OF THE 16 BIT WORD
              ;        THE POLYNOMIAL WAS FACTORED
              ;       BEFORE BEING WORKED OUT.
              ;
              ; LOG2(X) IS SCALED SO THAT LOG2(65536)
              ; BECOMES 1024
              ;
              ; ENTER WITH X IN AC0
              ; EXIT WITH LOG IN AC0
              ;
              ;
 01101 024155 LOG:   LDA 1.P15
 01102 044464        STA 1.LOGA     ; INITIALIZE NORMALIZER
 01103 101132 LOGL2: MOVZL# 0.0.SZC ; IS MSB = 1
 01104 000404        JMP LOGL4      ; YES
 01105 101120        MOVZL 0.0      ; NO SO SHIFT X
 01106 014460        DSZ LOGA       ; AND REDUCE A
 01107 000774        JMP LOGL2
 01110 024456 LOGL4: LDA 1.LOGA
 01111 125300        MOVS 1.1       ; SHIFT A
 01112 125220        MOVZR 1.1
 01113 125220        MOVZR 1.1
 01114 044452        STA 1.LOGA
 01115 101100        MOVL 0.0       ; ZERO MSB OF X
 01116 101220        MOVZR 0.0
 01117 111240        MOVOR 0.2      ; GET Y=X+1
 01120 101220        MOVZR 0.0      ; GET X-1
 01121 126440        SUBO 1.1       ; DIVIDE AC0/AC2
 01122 034141        LDA 3.MIN16    ; GET LOOP COUNTER
 01123 125120        MOVZL 1.1      ; SHIFT LOW DIVIDEND
```

```
0016  .MAIN
 01124 101100 LOGD2: MOVL 0.0       ; SHIFT HIGH DIVIDEND
 01125 142412        SUB# 2.0.SZC ; DOES DIVISOR GO IN
 01126 142400        SUB 2.0      ; YES
 01127 125100        MOVL 1.1     ; SHIFT LOW DIVIDEND
 01130 175404        INC 3.3.SZR
 01131 000773        JMP LOGD2
 01132 125220        MOVZR 1.1    ; DONE
 01133 125220        MOVZR 1.1
 01134 125220        MOVZR 1.1
 01135 044432        STA 1.LOGZ     ; SAVE Z
 01136 131000        MOV 1.2
 01137 004435        JSR MPY        ; GET Z*Z
 01140 004447        JSR LGSFT      ; SHIFT LEFT 3 PLACES (DP)
 01141 040427        STA 0.ZSQ      ; SAVE Z^2
 01142 105000        MOV 0.1
 01143 030430        LDA 2.CP5      ; GET 1ST CONSTANT
 01144 004430        JSR MPY
 01145 004442        JSR LGSFT
 01146 030424        LDA 2.CP3
 01147 113000        ADD 0.2        ;ADD TO POLYNOMIAL
 01150 024420        LDA 1.ZSQ
 01151 004423        JSR MPY
 01152 004435        JSR LGSFT
 01153 030416        LDA 2.CP1
 01154 113000        ADD 0.2        ; ADD TO POLYNOMIAL
 01155 024412        LDA 1.LOGZ
 01156 004416        JSR MPY
 01157 101220        MOVZR 0.0
 01160 101220        MOVZR 0.0
 01161 101220        MOVZR 0.0
 01162 101220        MOVZR 0.0
 01163 024403        LDA 1.LOGA
 01164 123000        ADD 1.0      ; ADD NORMALIZATION FACTOR
 01165 000702        JMP YPSET
                    ;
                    ;
 01166 000000 LOGA:  0
 01167 000000 LOGZ:  0
 01170 000000 ZSQ:   0
 01171 056141 CP1:   056141        ; CP1 = C1*LOG2(10)
 01172 016574 CP3:   016574
 01173 015421 CP5:   015421
                    ;
                    ; MULTIPLY AC1*AC2
                    ;
 01174 102460 MPY:   SUBC 0.0
 01175 054411        STA 3.MPYR    ; SAVE RETURN
 01176 034141        LDA 3.MIN16
 01177 125203 MPYL2: MOVR 1.1.SNC ; CHECK NEXT MULTIPLIER BIT
 01200 101201        MOVR 0.0.SKP ; 0 SO SHIFT
 01201 143220        ADDZR 2.0    ; 1 SO ADD MULTIPLICAND AND SHIFT
 01202 175404        INC 3.3.SZR
 01203 000774        JMP MPYL2
 01204 125260        MOVCR 1.1     ; SHIFT IN LAST LOW BIT
 01205 002401        JMP @ MPYR    ; RETURN
 01206 000000 MPYR:  0
                    ;
                    ; ROUTINE TO SHIFT AC0,AC1 3 PLACES
                    ;
```

```
0017  .MAIN
 01207 101120 LGSFT: MOVZL 0,0
 01210 125100        MOVL 1,1
 01211 101120        MOVZL 0,0
 01212 125100        MOVL 1,1
 01213 101120        MOVZL 0,0
 01214 125100        MOVL 1,1
 01215 001400        JMP 0,3     ; RETURN
                .EOT
```

```
                   ;
                   ;
 01216 040045 .TTIN: STA 0,TAC0    ; TTI SERVICE ROUTINE
 01217 044046        STA 1,TAC1    ; SAVE AC'S
 01220 050047        STA 2,TAC2
 01221 054050        STA 3,TAC3
 01222 060610        DIAC 0,TTI    ; GET CHAR
 01223 024106        LDA 1,CNTLR
 01224 106404        SUB 0,1,SZR   ; IS IT ^R
 01225 000411        JMP INL2
 01226 102400 LNH:   SUB 0,0       ; RESTART
 01227 040042        STA 0,OFLG
 01230 020114        LDA 0,CNTL
 01231 004243        JSR OUT
 01232 020115        LDA 0,R
 01233 004243        JSR OUT
 01234 004314        JSR CRLF
 01235 002116        JMP @ST       ; GOTO START
 01236 024107 INL2:  LDA 1,ESC
 01237 106404        SUB 0,1,SZR   ; IS IT ESC
 01240 000406        JMP INL6
 01241 102420 INL4:  SUBZ 0,0      ; YES
 01242 040057        STA 0,OUSE
 01243 040060        STA 0,ZFLAG
 01244 004314        JSR CRLF
 01245 000377        JMP RETRN     ; GO BACK TO COMMAND INPUT
 01246 024110 INL6:  LDA 1,ALT
 01247 106405        SUB 0,1,SNR   ; IS IT ALT MODE
 01250 000771        JMP INL4      ; YES
 01251 024057        LDA 1,OUSE
 01252 125005        MOV 1,1,SNR
 01253 000402        JMP INL8
 01254 000256        JMP RESTR     ; IGNORE INPUT
 01255 024071 INL8:  LDA 1,CR
 01256 106404        SUB 0,1,SZR   ; IS IT CARR RET
 01257 000403        JMP INL10
 01260 004314        JSR CRLF
 01261 000442        JMP CMND
 01262 024111 INL10: LDA 1,RUB
 01263 106404        SUB 0,1,SZR   ; IS IT RUBOUT
 01264 000415        JMP INL12
 01265 020052 INL11: LDA 0,CMCNT   ; YES SO BACKUP COMMAND
 01266 101005        MOV 0,0,SNR   ; BUFFER COUNTER
 01267 000256        JMP RESTR     ; NOTHING TO ERASE
 01270 030051        LDA 2,CMBUF
 01271 113000        ADD 0,2
 01272 102400        SUB 0,0
 01273 041000        STA 0,0,2     ; CLEAR ERASED CHAR
 01274 014052        DSZ CMCNT
 01275 000401        JMP .+1
 01276 020132        LDA 0,BKSL    ; TYPE BACK SLASH
 01277 004243        JSR OUT
 01300 000256        JMP RESTR
 01301 024112 INL12: LDA 1,CAN
 01302 106405        SUB 0,1,SNR   ; IS IT CANCEL
 01303 000762        JMP INL11
 01304 004243        JSR OUT       ; ADD CHAR TO BUFFER
 01305 030052        LDA 2,CMCNT
 01306 024134        LDA 1,P24
```

```
0019  .MAIN
 01307 132032         ADCZ# 1.2.SZC ; SKIP IF CMCNT < 24
 01310 000411         JMP INER
 01311 024113         LDA 1.MASK1
 01312 123400         AND 1.0
 01313 030052         LDA 2.CMCNT
 01314 024051         LDA 1.CMBUF
 01315 133000         ADD 1.2
 01316 041000         STA 0.0.2
 01317 010052         ISZ CMCNT    ; ADVANCE CHAR COUNTER
 01320 000256         JMP RESTR
 01321 020144 INER:   LDA 0.P2      ; LOAD AC0 WITH 2
 01322 002135         JMP @ERR
                ;
                ;
                ;
                ; ROUTINE TO INTERPRET COMMANDS
                ;
 01323 020052 CMND:   LDA 0.CMCNT
 01324 101005         MOV 0.0.SNR
 01325 000377         JMP RETRN     ; NO COMMAND SO RETURN
 01326 030051         LDA 2.CMBUF
 01327 021000         LDA 0.0.2     ; GET CHAR FROM BUFFER
 01330 105300         MOVS 0.1
 01331 151400         INC 2.2
 01332 034142         LDA 3.SPACE
 01333 021000         LDA 0.0.2     ; GET 2ND CHAR
 0133 1415       SUB# 0.3.SNR ; TEST FOR SPACE
 01335 000422         JMP CML1      ; SPACE MEANS END OF WORD
 01336 107300         ADDS 0.1      ; PACK 2ND CHAR
 01337 044206         STA 1.ASC1
 01340 151400         INC 2.2
 01341 021000         LDA 0.0.2     ; GET 3RD CHAR
 01342 105300         MOVS 0.1
 01343 151400         INC 2.2
 01344 021000         LDA 0.0.2     ; GET 4TH CHAR
 01345 116415         SUB# 0.3.SNR ; TEST FOR SPACE
 01346 000415         JMP CML2
 01347 107300         ADDS 0.1      ; PACK 2ND WORD
 01350 044207         STA 1.ASC2
 01351 151400         INC 2.2
 01352 021000         LDA 0.0.2     ; GET 5TH CHAR
 01353 116415         SUB# 0.3.SNR ; TEST FOR SP
 01354 000412         JMP CML3
 01355 105300         MOVS 0.1      ; NOT A SP
 01356 000411         JMP CML4
 01357 102400 CML1:   SUB 0.0       ; ZERO CHARS 2-6
 01360 107300         ADDS 0.1
 01361 044206         STA 1.ASC1
 01362 126400         SUB 1.1
 01363 102400 CML2:   SUB 0.0          ; ZERO CHARS 4-6
 01364 107300         ADDS 0.1
 01365 044207         STA 1.ASC2
 01366 126400 CML3:   SUB 1.1.      ; ZERO CHARS 5-6
 01367 102400 CML4:   SUB 0.0       ; ZERO CHAR 6
 01370 107300         ADDS 0.1
 01371 044210         STA 1.ASC3
 01372 030136         LDA 2.ASCII ; COMPARE COMMANDS
 01373 020143         LDA 0.MIN27
                      STA    1
```

```
0020  .MAIN
 01375 020206 CMLOP: LDA 0.ASC1
 01376 025000        LDA 1.0.2
 01377 122404        SUB 1.0.SZR
 01400 000411        JMP CMAGN
 01401 020207        LDA 0.ASC2
 01402 025001        LDA 1.1.2
 01403 122404        SUB 1.0.SZR
 01404 000405        JMP CMAGN
 01405 020210        LDA 0.ASC3
 01406 025002        LDA 1.2.2
 01407 122405        SUB 1.0.SNR
 01410 003003        JMP @3.2      ; MATCH SO EXCECUTE
 01411 151400 CMAGN: INC 2.2
 01412 151400        INC 2.2
 01413 151400        INC 2.2
 01414 151400        INC 2.2
 01415 010066        ISZ CNT1      ; 27 TRIES YET?
 01416 000757        JMP CMLOP
 01417 102520        SUBZL 0.0     ; YES SO NO GO
 01420 101120        MOVZL 0.0     ; SET AC0=2
 01421 002135        JMP @ERR
                    ;
                    ;
                    ; ROUTINE TO TYPE OUT BUFFER CONTENTS
                    ;
 01422 006131 TYPE:  JSR @GTNUM    ; GET I.N.M
 01423 006177        JSR @CHECK
 01424 010057        ISZ OUSE      ; SET OUTPUT IN USE FLAG
 01425 004314        JSR CRLF
 01426 064477        READS 1
 01427 030173        LDA 2.MASK6
 01430 133400        AND 1.2
 01431 050451        STA 2.SW9
 01432 020054 TPL4:  LDA 0.NNUM
 01433 006201        JSR @DCOUT    ; TYPE OUT CHANNEL NO.
 01434 020446        LDA 0.SW9
 01435 101004        MOV 0.0.SZR
 01436 000406        JMP TPL5
 01437 020144        LDA 0.P2
 01440 006166        JSR @TAB
 01441 020137        LDA 0.MIN4    ; SET UP SHORT LINE COUNTER
 01442 040066        STA 0.CNT1
 01443 000405        JMP TYPL6
 01444 020150 TPL5:  LDA 0.P4
 01445 006166        JSR @TAB
 01446 020165        LDA 0.MIN8    ; SET UP LINE COUNTER
 01447 040066        STA 0.CNT1
 01450 020142 TYPL6: LDA 0.SPACE
 01451 024431        LDA 1.SW9
 01452 125004        MOV 1.1.SZR
 01453 004243        JSR OUT
 01454 020163        LDA 0.BUFAD
 01455 034053        LDA 3.INUM
 01456 117000        ADD 0.3
 01457 031400        LDA 2.0.3
 01460 024054        LDA 1.NNUM
 01461 133000        ADD 1.2
 01462 021000        LDA 0.0.2
 01463 006201        JSR @DCOUT    ; OUTPUT CHAN. CONTENTS
```

```
0021  .MAIN
 01464 020055          LDA 0.MNUM
 01465 024054          LDA 1.NNUM
 01466 106405          SUB 0.1.SNR   ; HAS CHAN M BEEN OUTPUTTED
 01467 000406          JMP TYPEN     ; YES SO END
 01470 010054          ISZ NNUM
 01471 010066          ISZ CNT1
 01472 000756          JMP TYPL6
 01473 004314          JSR CRLF      ; OUTPUT NEXT LINE
 01474 000736          JMP TPL4
 01475 004314 TYPEN:   JSR CRLF
 01476 004314          JSR CRLF
 01477 102400          SUB 0.0
 01500 040057          STA 0.OUSE
 01501 000377          JMP RETRN
 01502 000000 SW9:     0
              ;
              ;
              ;
              ; ROUTINE TO PUNCH CHANNELS N TO M IN PDP
              ; COMPATIBLE BINARY
              ;
 01503 006131 PUNCH:   JSR @GTNUM    ; GET I.N.M
 01504 006177          JSR @CHECK    ; CHECK N.M FOR ERRORS
 01505 010057          ISZ OUSE      ;SET TTO IN USE FLAG
 01506 063077          HALT          ; WAIT FOR PUNCH ON
 01507 004451          JSR BLANK
 01510 020053 PUNL4:   LDA 0.INUM
 01511 034163          LDA 3.BUFAD
 01512 117000          ADD 0.3
 01513 031400          LDA 2.0.3
 01514 020054          LDA 0.NNUM
 01515 113000          ADD 0.2
 01516 021000          LDA 0.0.2
 01517 101220          MOVZR 0.0
 01520 101220          MOVZR 0.0
 01521 101220          MOVZR 0.0
 01522 101220          MOVZR 0.0
 01523 101320          MOVZS 0.0
 01524 024167          LDA 1.MASK2
 01525 123400          AND 1.0       ; MASK OFF 4 LS BITS
 01526 024175          LDA 1.BIT8
 01527 123000          ADD 1.0
 01530 004243          JSR OUT       ; OUTPUT 4 MS BITS
 01531 021000          LDA 0.0.2
 01532 101120          MOVZL 0.0
 01533 101120          MOVZL 0.0
 01534 101320          MOVZS 0.0
 01535 024170          LDA 1.MASK3   ; MASK OFF 6 LS BITS
 01536 123400          AND 1.0
 01537 024175          LDA 1.BIT8
 01540 123000          ADD 1.0
 01541 004243          JSR OUT       ; OUTPUT NEXT 6 BITS
 01542 021000          LDA 0.0.2
 01543 024170          LDA 1.MASK3
 01544 123400          AND 1.0
 01545 024175          LDA 1.BIT8
 01546 123000          ADD 1.0
 01547 004243          JSR OUT       ; OUTPUT LAST 6 BITS
 01550 010054          ISZ NNUM
```

```
0022  .MAIN
 01551 020054         LDA 0.NNUM
 01552 024055         LDA 1.MNUM
 01553 122023         ADCZ 1.0.SNC ; SKP IF FINISHED
 01554 000734         JMP PUNL4
 01555 004403         JSR BLANK
 01556 063077         HALT          ; WAIT FOR PUNCH OFF
 01557 000377         JMP RETRN
 01560 054410 BLANK:  STA 3.BLNK    ; TRAILER AND LEADER SUBR.
 01561 020176         LDA 0.P360
 01562 040067         STA 0.CNT2
 01563 102400         SUB 0.0
 01564 004243 BLOOP:  JSR OUT
 01565 014067         DSZ CNT2
 01566 000776         JMP .-2
 01567 002401         JMP @BLNK
 01570 000000 BLNK:   0
              ;
              ;
              ; READ ROUTINE READS PAPER TAPE WRITTEN IN
              ; PDP-15 BINARY FORMAT
              ;
 01571 006131 READ:   JSR @GTNUM    ; GET I.N.M
 01572 006177         JSR @CHECK    ; CHECK N.M FOR ERRORS
 01573 030163         LDA 2.BUFAD
 01574 024053         LDA 1.INUM
 01575 133000         ADD 1.2
 01576 025000         LDA 1.0.2
 01577 030054         LDA 2.NNUM
 01600 133000         ADD 1.2
 01601 060277         INTDS         ; TURN OFF INTERRUPT
 01602 060110         NIOS TTI      ; START READER
 01603 004432 RDL2:   JSR IN        ; GET 1ST CHAR
 01604 024167         LDA 1.MASK2
 01605 123400         AND 1.0
 01606 101120         MOVZL 0.0     ; PACK CHAR
 01607 101120         MOVZL 0.0
 01610 101120         MOVZL 0.0
 01611 105120         MOVZL 0.1
 01612 004423         JSR IN        ; GET 2ND CHAR
 01613 034170         LDA 3.MASK3
 01614 163700         ANDS 3.0
 01615 101220         MOVZR 0.0
 01616 101220         MOVZR 0.0
 01617 107000         ADD 0.1
 01620 004415         JSR IN        ; GET 3RD CHAR
 01621 034170         LDA 3 MASK3
 01622 163400         AND 3.0
 01623 107000         ADD 0.1       ; PACK LAST CHAR
 01624 045000         STA 1.0.2
 01625 151400         INC 2.2
 01626 010054         ISZ NNUM
 01627 020054         LDA 0.NNUM
 01630 024055         LDA 1.MNUM
 01631 122023         ADCZ 1.0.SNC ; SKIP IF FINISHED
 01632 000751         JMP RDL2      ; GET ANOTHER CHANNEL
 01633 060210 RDL4:   NIOC TTI
 01634 000377         JMP RETRN
 01635 054412 IN:     STA 3.INR
 01636 063610         SKPDN TTI
```

```
0023  .MAIN
 01637 000777           JMP .-1
 01640 060510           DIAS 0,TTI
 01641 101005           MOV 0,0,SNR
 01642 000774           JMP IN+1       ; IGNORE LEADER
 01643 034111           LDA 3,RUB
 01644 116405           SUB 0,3,SNR
 01645 000766           JMP RDL4       ;RUBOUT SO ABORT READ
 01646 002401           JMP @INR
 01647 000000 INR:      0
                ;
                ;
                ;
                ; ROUTINE TO CHANGE CHANNEL N
                ;
 01650 006131 CHNGE:  JSR @GTNUM     ; GET I,N
 01651 024054           LDA 1,NNUM
 01652 006200           JSR @.CKL2     ; MAKE SURE N>=SIZE
 01653 030163           LDA 2,BUFAD
 01654 020053           LDA 0,INUM
 01655 113000           ADD 0,2
 01656 021000           LDA 0,0,2
 01657 030054           LDA 2,NNUM
 01660 113000           ADD 0,2
 01661 021000           LDA 0,0,2       ; GET CONTENTS OF
 01662 006201           JSR @DCOUT      ; CHANNEL N & OUTPUT
 01663 050164           STA 2,BADR      ; SAVE CHANNEL ADDRESS
 01664 102520           SUBZL 0,0
 01665 101140           MOVOL 0,0       ; GENERATE +3
 01666 006166           JSR @TAB
 01667 102440           SUBO 0,0
 01670 024134           LDA 1,P24
 01671 044067           STA 1,CNT2
 01672 030051           LDA 2,CMBUF
 01673 041000 CHNL2:  STA 0,0,2       ; CLEAR COMMAND BUFFER
 01674 151400           INC 2,2
 01675 014067           DSZ CNT2
 01676 000775           JMP CHNL2
 01677 040052           STA 0,CMCNT
 01700 040067           STA 0,CNT2
 01701 010057           ISZ OUSE        ; SET OUTPUT IN USE FLAG
 01702 063610 CHNWT:  SKPDN TTI
 01703 000777           JMP .-1
 01704 060610           DIAC 0,TTI
 01705 024106           LDA 1,CNTLR
 01706 106405           SUB 0,1,SNR
 01707 002116           JMP @ST         ; CHAR WAS ^R SO RESTART
 01710 024107           LDA 1,ESC
 01711 106405           SUB 0,1,SNR
 01712 000377           JMP RETRN       ; CHAR WAS ESC SO RETURN
 01713 024110           LDA 1,ALT
 01714 106405           SUB 0,1,SNR
 01715 000377           JMP RETRN       ; CHAR WAS ALT MODE
 01716 024111           LDA 1,RUB
 01717 106404           SUB 0,1,SZR
 01720 000416           JMP CHNL4       ; ADD CHAR TO BUFFER
 01721 020067           LDA 0,CNT2      ; CHAR WAS RUBOUT
 01722 101005           MOV 0,0,SNR     ; NO CHARS YET - IGNORE RUB
 01723 000757           JMP CHNWT
 01724 014067           DSZ CNT2
```

```
0024  .MAIN
 01725 000401             JMP .+1
 01726 020132             LDA 0,BKSL
 01727 004243             JSR OUT       ; OUTPUT "\"
 01730 102400             SUB 0,0
 01731 024067             LDA 1,CNT2
 01732 030051             LDA 2,CMBUF
 01733 133000             ADD 1,2
 01734 041000             STA 0,0,2     ; ZERO ERASED CHAR
 01735 000745             JMP CHNWT      ; WAIT FOR NEXT CHAR
 01736 004243 CHNL4: JSR OUT
 01737 024071             LDA 1,CR
 01740 106404             SUB 0,1,SZR
 01741 000413             JMP CHNL6      ; ADD CHAR TO BUFFER
 01742 004314             JSR CRLF       ; CHAR WAS RETURN
 01743 020067             LDA 0,CNT2
 01744 101005             MOV 0,0,SNR
 01745 000377             JMP RETRN      ; NO CHARS - DO NOTHING
 01746 102400             SUB 0,0
 01747 040057             STA 0,OUSE
 01750 040067             STA 0,CNT2
 01751 006202             JSR @DBIN      ; CONVERT TO BINARY
 01752 046164             STA 1,@BADR    ; STORE NO.
 01753 000377             JMP RETRN
 01754 024113 CHNL6: LDA 1,MASK1
 01755 123400             AND 1,0
 01756 024203             LDA 1,AS0
 01757 030204             LDA 2,AS9
 01760 142033             ADCZ# 2,0,SNC ; SKIP IF >9
 01761 106032             ADCZ# 0,1,SZC ; SKIP IF >=0
 01762 000407             JMP CHNER
 01763 030051             LDA 2,CMBUF
 01764 024067             LDA 1,CNT2
 01765 133000             ADD 1,2
 01766 041000             STA 0,0,2
 01767 010067             ISZ CNT2
 01770 000712             JMP CHNWT
 01771 020146 CHNER: LDA 0,P6        ; ERROR 6
 01772 002135             JMP @ERR
                    ;
                    ;
                    ; ROUTINE TO SUM (INTEGRATE) CHANNELS N TO M
                    ; IN BUFFER I
                    ;
 01773 006131 SUM:   JSR @GTNUM    ; GET I,N,M
 01774 006177        JSR @CHECK    ; CHECK N,M FOR ERRORS
 01775 102400        SUB 0,0
 01776 040127        STA 0,TOTL1
 01777 040130        STA 0,TOTL2
 02000 034163        LDA 3,BUFAD
 02001 024053        LDA 1,INUM
 02002 137000        ADD 1,3
 02003 031400        LDA 2,0,3     ; GET ADDRESS OF BUFFER I
 02004 024054        LDA 1,NNUM
 02005 133000        ADD 1,2
 02006 034055        LDA 3,MNUM
 02007 021000 SUMLP: LDA 0,0,2     ; GET CONTENTS OF CHANNEL
 02010 024127        LDA 1,TOTL1
 02011 107022        ADDZ 0,1,SZC
 02012 010130        ISZ TOTL2
```

```
0025  .MAIN
 02013 044127         STA 1,TOTL1
 02014 010054         ISZ NNUM
 02015 151400         INC 2,2
 02016 020054         LDA 0,NNUM
 02017 162023         ADCZ 3,0,SNC   ; SKIP IF FINISHED
 02020 000767         JMP SUMLP
 02021 024126         LDA 1,DBD     ; SET DCOUT TO
 02022 020125         LDA 0,BIND    ; OUTPUT IN
 02023 044125         STA 1,BIND    ; DOUBLE PRECISION
 02024 040126         STA 0,DBD
 02025 020130         LDA 0,TOTL2
 02026 030127         LDA 2,TOTL1
 02027 006201         JSR @DCOUT    ; OUTPUT TOTAL
 02030 024126         LDA 1,DBD     ; RESET DCOUT
 02031 020125         LDA 0,BIND    ; FOR SINGLE
 02032 040126         STA 0,DBD     ; PRECISION
 02033 044125         STA 1,BIND
 02034 004314         JSR CRLF
 02035 000377         JMP RETRN
                ;
                ;
                ;
                ; ROUTINE TO HALT ANALYZING
                ;
 02036 060237 HLT:   NIOC 37       ; CLEAR ADC INTERFACE
 02037 020064         LDA 0,ANLYZ
 02040 101005         MOV 0,0,SNR
 02041 000377         JMP RETRN
 02042 006201         JSR @DCOUT    ; OUTPUT BUFFER NO.
 02043 004314         JSR CRLF
 02044 102400         SUB 0,0
 02045 040064         STA 0,ANLYZ
 02046 000377         JMP RETRN
                ;
                ;
                ; ROUTINE TO CONTINUE AFTER A HALT
                ;
 02047 020061 CONT:  LDA 0,ACTIV
 02050 101004         MOV 0,0,SZR
 02051 000404         JMP CONL2
 02052 020146         LDA 0,P6      ; NO ACTIVE BUFFER
 02053 101400         INC 0,0       ; ERROR 7
 02054 002135         JMP @ERR
 02055 040053 CONL2:  STA 0,INUM
 02056 040064         STA 0,ANLYZ
 02057 006201         JSR @DCOUT
 02060 004314         JSR CRLF
 02061 000410         JMP STRL2
                ;
                ;
                ; ROUTINE TO START ADC
                ;
 02062 006131 STRT:  JSR @GTNUM    ; GET I
 02063 020064         LDA 0,ANLYZ
 02064 101005         MOV 0,0,SNR   ; IS ADC ACTIVE ALREADY
 02065 000404         JMP STRL2
 02066 020165         LDA 0,MIN8    ; YES ERROR 8
 02067 100400         NEG 0,0
                      JMP @ERR
```

```
0026  .MAIN
 02071 020241 STRL2: LDA 0,P550    ; RESET MINUTES COUNTER
 02072 040240        STA 0,RUNT
 02073 020053        LDA 0,INUM
 02074 040061        STA 0,ACTIV
 02075 040064        STA 0,ANLYZ
 02076 030163        LDA 2,BUFAD
 02077 113000        ADD 0,2
 02100 021000        LDA 0,0,2     ; GET BUFFER ADDRESS
 02101 061337        DOAP 0,37     ; OUTPUT TO INTERFACE
 02102 000377        JMP RETRN
                ;
                ;
                ; ROUTINE TO FIND THE CHANNEL WITH THE MAXIMUM
                ; NUMBER OF COUNTS
                ;
 02103 006131 MAX:   JSR @GTNUM    ; GET I
 02104 034163        LDA 3,BUFAD
 02105 020053        LDA 0,INUM
 02106 117000        ADD 0,3
 02107 031400        LDA 2,0,3     ; GET ADDRESS OF BUFF I
 02110 102400        SUB 0,0
 02111 040067        STA 0,CNT2
 02112 021000 MAXL2: LDA 0,0,2
 02113 040442        STA 0,MAXN
 02114 020067        LDA 0,CNT2
 02115 040441        STA 0,MAXCH
 02116 034162        LDA 3,BUFSZ
 02117 024053        LDA 1,INUM
 02120 137000        ADD 1,3
 02121 025400        LDA 1,0,3     ; GET SIZE OF BUFF I
 02122 125100        MOVL 1,1
 02123 135220        MOVZR 1,3
 02124 174400        NEG 3,3
 02125 174000        COM 3,3       ; DECREMENT AC3
 02126 010067 MAXL4: ISZ CNT2
 02127 151400        INC 2,2
 02130 021000        LDA 0,0,2     ; GET NEXT CHANNEL
 02131 024424        LDA 1,MAXN
 02132 122033        ADCZ# 1,0,SNC ; IS IT LARGEST YET
 02133 000404        JMP MAXL6
 02134 040421        STA 0,MAXN    ; YES
 02135 020067        LDA 0,CNT2
 02136 040420        STA 0,MAXCH
 02137 020067 MAXL6: LDA 0,CNT2
 02140 162433        SUBZ# 3,0,SNC ; SKIP IF DONE
 02141 000765        JMP MAXL4
 02142 010057        ISZ OUSE
 02143 020412        LDA 0,MAXN
 02144 006201        JSR @DCOUT
 02145 030103        LDA 2,MES5
 02146 006120        JSR @TEXT
 02147 020407        LDA 0,MAXCH
 02150 006201        JSR @DCOUT
 02151 004314        JSR CRLF
 02152 102400        SUB 0,0
 02153 040057        STA 0,OUSE
 02154 000377        JMP RETRN
 02155 000000 MAXN:  0
 02156 000000 MAXCH: 0
```

```
                    ;
                    ;
                    ;
                    ; ROUTINE TO SET THE BUFFER CONTENTS TO ZERO
                    ;
 02157 004402 ZERO:    JSR .ZER
 02160 000377          JMP RETRN
 02161 054461 .ZER:    STA 3.ZERET   ; SAVE RETURN ADDRESS
 02162 030051          LDA 2.CMBUF
 02163 021000 ZERL2:   LDA 0.0.2
 02164 024142          LDA 1.SPACE
 02165 151400          INC 2.2
 02166 106414          SUB# 0.1.SZR ; IS CHAR A SPACE
 02167 000774          JMP ZERL2     ; NO TRY NEXT CHAR
 02170 021000          LDA 0.0.2
 02171 024121          LDA 1.A
 02172 106414          SUB# 0.1.SZR ; SKIP IF "A"
 02173 000426          JMP ZERI
 02174 151400          INC 2.2
 02175 021000          LDA 0.0.2
 02176 024122          LDA 1.L
 02177 122415          SUB# 1.0.SNR ; SKIP IF NOT "L"
 02200 000404          JMP ZERL4
 02201 102520 ZERR2:   SUBZL 0.0
 02202 101120          MOVZL 0.0     ; ERROR 2
 02203 002135          JMP @ERR
 02204 151400 ZERL4:   INC 2.2
 02205 021000          LDA 0.0.2
 02206 122414          SUB# 1.0.SZR ; SKIP IF "L"
 02207 000772          JMP ZERR2
 02210 030062          LDA 2.BUFST   ;"ALL" SO ZERO
 02211 034063          LDA 3.BUFEN   ; ENTIRE BUFFER
 02212 102400          SUB 0.0
 02213 041000 ZERL6:   STA 0.0.2     ; ZERO BUFFER LOC.
 02214 151400          INC 2.2
 02215 172033          ADCZ# 3.2.SNC ; END OF BUFFER YET?
 02216 000775          JMP ZERL6     ; NO
 02217 040053          STA 0.INUM
 02220 002422          JMP @ZERET
 02221 006131 ZERI:    JSR @GTNUM    ; ZERO BUFFER I
 02222 034163          LDA 3.BUFAD
 02223 024053          LDA 1.INUM
 02224 137000          ADD 1.3
 02225 031400          LDA 2.0.3     ; GET BUFFER ADDRESS
 02226 034162          LDA 3.BUFSZ
 02227 137000          ADD 1.3
 02230 021400          LDA 0.0.3     ; GET BUFFER SIZE
 02231 101100          MOVL 0.0
 02232 115220          MOVZR 0.3     ; MASK OFF BIT 0
 02233 157000          ADD 2.3       ; GET BUFFER END ADDRESS
 02234 102400          SUB 0.0
 02235 041000 ZERL8:   STA 0.0.2
 02236 151400          INC 2.2
 02237 172033          ADCZ# 3.2.SNC ; END OF BUFFER YET?
 02240 000775          JMP ZERL8     ; NO
 02241 002401          JMP @ZERET
 02242 000000 ZERET:   0
                    ;
                    ;
```

```
                        ; ROUTINE TO CLEAR BUFFERS FROM THE ACTIVE LIST
                        ;
 02243 004716 CLEAR:    JSR .ZER
 02244 020053           LDA 0,INUM
 02245 101004           MOV 0,0,SZR
 02246 000414           JMP CLL4        ; CLEAR I ONLY
 02247 030162           LDA 2,BUFSZ     ; CLEAR ALL
 02250 034062           LDA 3,BUFST
 02251 102400           SUB 0,0
 02252 041000 CLL2:     STA 0,0,2
 02253 151400           INC 2,2
 02254 172033           ADCZ# 3,2,SNC ; BUFFERS CLEARED YET?
 02255 000775           JMP CLL2
 02256 060237 CLL3:     NIOC 37         ; YES
 02257 040061           STA 0,ACTIV
 02260 040064           STA 0,ANLYZ
 02261 000377           JMP RETRN
 02262 020053 CLL4:     LDA 0,INUM
 02263 030163           LDA 2,BUFAD
 02264 113000           ADD 0,2
 02265 025000           LDA 1,0,2       ; GET BUFFER ADDRESS
 02266 044164 CLL5:     STA 1,BADR
 02267 030162           LDA 2,BUFSZ
 02270 113000           ADD 0,2
 02271 035000           LDA 3,0,2       ; GET BUFFER SIZE
 02272 175100           MOVL 3,3
 02273 175220           MOVZR 3,3
 02274 054123           STA 3,SZE
 02275 167000           ADD 3,1         ; GET NEXT BUFFER ADDRESS
 02276 020065           LDA 0,M40
 02277 040066           STA 0,CNT1
 02300 102400           SUB 0,0
 02301 040067           STA 0,CNT2
 02302 020067 CLL6:     LDA 0,CNT2
 02303 034163           LDA 3,BUFAD
 02304 117000           ADD 0,3
 02305 031400           LDA 2,0,3
 02306 132405           SUB 1,2,SNR     ; IS THIS NEXT ADDRESS
 02307 000405           JMP SHFT
 02310 010067           ISZ CNT2        ; NO  TRY NEXT BUFFER
 02311 010066           ISZ CNT1
 02312 000770           JMP CLL6
 02313 000434           JMP CLL12       ; ALL BUFFERS TRIED
 02314 020067 SHFT:     LDA 0,CNT2      ; SHIFT ADJACENT BUFFER
 02315 030162           LDA 2,BUFSZ     ; DOWN
 02316 113000           ADD 0,2
 02317 035000           LDA 3,0,2
 02320 175100           MOVL 3,3
 02321 175220           MOVZR 3,3
 02322 054066           STA 3,CNT1
 02323 054070           STA 3,CNT3
 02324 024123           LDA 1,SZE
 02325 030164           LDA 2,BADR
 02326 135000 MOVE:     MOV 1,3
 02327 157000           ADD 2,3
 02330 021400           LDA 0,0,3       ; MOVE FROM OLD SPOT
 02331 041000           STA 0,0,2       ; TO NEW SPOT
 02332 151400           INC 2,2
 02333 014070           DSZ CNT3
```

```
0029  .MAIN
 02334 000772          JMP MOVE
 02335 020067          LDA 0,CNT2    ; UPDATE BUFFER ADDRESS
 02336 030163          LDA 2,BUFAD
 02337 113000          ADD 0,2
 02340 020164          LDA 0,BADR
 02341 025000          LDA 1,0,2
 02342 041000          STA 0,0,2
 02343 020066          LDA 0,CNT1    ; GET NEW ADDRESS
 02344 107000          ADD 0,1       ; TO CHECK FOR
 02345 020067          LDA 0,CNT2
 02346 000720          JMP CLL5
 02347 020053 CLL12:   LDA 0,INUM
 02350 030162          LDA 2,BUFSZ
 02351 113000          ADD 0,2
 02352 126400          SUB 1,1
 02353 045000          STA 1,0,2     ; CLEAR SIZE
 02354 045020          STA 1,20,2    ; CLEAR ADDRESS
 02355 024061          LDA 1,ACTIV
 02356 106405          SUB 0,1,SNR
 02357 000677          JMP CLL3      ; CLEAR ADC
 02360 000377          JMP RETRN
                       .EOT
```

```
                      ;
                      ; ROUTINE TO OUTPUT ACTIVE BUFFERS AND
                      ; THEIR ADDRESSES AND STATUS
                      ;
02361 010057 BUFF:    ISZ OUSE        ; SET OUTPUT IN USE FLAG
02362 030100          LDA 2,MES2
02363 006120          JSR @TEXT       ; OUTPUT HEADER
02364 004314          JSR CRLF
02365 030105          LDA 2,MES7
02366 006120          JSR @TEXT
02367 004314          JSR CRLF
02370 102520          SUBZL 0,0
02371 040067          STA 0,CNT2
02372 030162          LDA 2,BUFSZ
02373 050066          STA 2,CNT1
02374 030066 BUFL2:   LDA 2,CNT1
02375 025001          LDA 1,1,2       ; GET BUFFER SIZE
02376 125103          MOVL 1,1,SNC    ; SKP IF BUFFER EXISTS
02377 000443          JMP BUFL8+1
02400 125220          MOVZR 1,1
02401 044123          STA 1,SZE
02402 020067          LDA 0,CNT2      ; GET BUFFER NO.
02403 006201          JSR @DCOUT
02404 102520          SUBZL 0,0
02405 006166          JSR @TAB        ; OUTPUT 3 SPACES
02406 020123          LDA 0,SZE
02407 006201          JSR @DCOUT      ; OUTPUT SIZE
02410 020151          LDA 0,P8
02411 006166          JSR @TAB        ; OUTPUT 8 SPACES
02412 030066          LDA 2,CNT1
02413 025021          LDA 1,21,2      ; GET ADDRESS OF BUFFER
02414 006215          JSR @BINO
02415 024067          LDA 1,CNT2
02416 020064          LDA 0,ANLYZ
02417 106414          SUB# 0,1,SZR    ; IS BUFFER NO.=ANLYZ
02420 000413          JMP BUFL6
02421 102520          SUBZL 0,0       ; YES SO TYPE A
02422 006166          JSR @TAB
02423 020121          LDA 0,A
02424 004243          JSR OUT
02425 020061          LDA 0,ACTIV
02426 106414          SUB# 0,1,SZR    ; SKP IF BUFFER ACTIVE
02427 000412          JMP BUFL8
02430 020122 BUFL4:   LDA 0,L
02431 004243          JSR OUT
02432 000407          JMP BUFL8
02433 020061 BUFL6:   LDA 0,ACTIV
02434 106414          SUB# 0,1,SZR
02435 000404          JMP BUFL8
02436 102520          SUBZL 0,0
02437 006166          JSR @TAB
02440 000770          JMP BUFL4
02441 004314 BUFL8:   JSR CRLF
02442 010067          ISZ CNT2
02443 010066          ISZ CNT1
02444 020067          LDA 0,CNT2
02445 024133          LDA 1,P20   ; +20
02446 122033          ADCZ# 1,0,SNC ; CHECKED ALL BUFFERS?
02447 000725          JMP BUFL2       ; NO GO BACK
```

```
0031  .MAIN
 02450 006117 BFL10: JSR @TOTAL    ; COMPUT TOTAL
 02451 024063        LDA 1,BUFEN   ; MEMORY IN USE
 02452 030062        LDA 2,BUFST
 02453 146400        SUB 2,1
 02454 106400        SUB 0,1       ; CALCULATE
 02455 121000        MOV 1,0       ; MEMORY LEFT
 02456 006201        JSR @DCOUT
 02457 030102        LDA 2,MES4
 02460 006120        JSR @TEXT     ; TYPE "WORDS LEFT"
 02461 004314        JSR CRLF
 02462 004314        JSR CRLF
 02463 000377        JMP RETRN
                ;
                ;
                ;
                ; ROUTINE TO ADD A BUFFER OF SIZE N TO
                ; THE ACTIVE LIST
                ;
 02464 010056 SIZE:  ISZ CKFLG     ; INHIBIT ERROR CHECK
 02465 006131        JSR @GTNUM
 02466 030162        LDA 2,BUFSZ
 02467 020053        LDA 0,INUM
 02470 113000        ADD 0,2
 02471 021000        LDA 0,0,2     ; GET BUFSZ(I)
 02472 101103        MOVL 0,0,SNC ; IS BIT 0 SET
 02473 000403        JMP SZL2
 02474 020152        LDA 0,P9      ; YES - ERROR 9
 02475 002135        JMP @ERR
 02476 006117 SZL2:  JSR @TOTAL
 02477 024063        LDA 1,BUFEN
 02500 030062        LDA 2,BUFST
 02501 146400        SUB 2,1
 02502 106400        SUB 0,1       ; COMPUTE MEMORY
 02503 040066        STA 0,CNT1    ; AVAILABLE
 02504 030054        LDA 2,NNUM
 02505 102400        SUB 0,0
 02506 132032        ADCZ# 1,2,SZC ; IS N>AVAILABLE
 02507 002135        JMP @ERR      ; YES - ERROR 0
 02510 030162        LDA 2,BUFSZ
 02511 020053        LDA 0,INUM
 02512 113000        ADD 0,2
 02513 024054        LDA 1,NNUM
 02514 125100        MOVL 1,1
 02515 125240        MOVOR 1,1     ; SET BIT 0 ON BUFSZ(I)
 02516 045000        STA 1,0,2     ; STORE IN BUFFER LIST
 02517 020062        LDA 0,BUFST
 02520 024066        LDA 1,CNT1
 02521 123000        ADD 1,0
 02522 101400        INC 0,0       ; COMPUTE ADDRESS
 02523 030163        LDA 2,BUFAD
 02524 024053        LDA 1,INUM
 02525 133000        ADD 1,2
 02526 041000        STA 0,0,2
 02527 000377        JMP RETRN
                ;
                ;
                ; ROUTINE TO SET THE BUFFER TO BE DISPLAYED
                ;
 02530 006131 DISP:  JSR @GTNUM    ; GET I
```

```
0032  .MAIN
 02531 024053          LDA 1,INUM
 02532 044224          STA 1,DISB
 02533 030162          LDA 2,BUFSZ
 02534 133000          ADD 1,2
 02535 021000          LDA 0,0,2     ; GET SIZE OF BUFFER
 02536 101100          MOVL 0,0
 02537 101220          MOVZR 0,0     ; CLEAR BIT 0
 02540 040227          STA 0,DISSZ
 02541 021020          LDA 0,20,2    ; GET ADDRESS
 02542 040225          STA 0,DISBF
 02543 000377          JMP RETRN
                ;
                ;
                ; DEVICE HANDLER FOR THE ADC OVERFLOW FLAG
                ;
 02544 040045 .OVER: STA 0,TAC0    ; SAVE
 02545 044046        STA 1,TAC1    ; ACCUMULATORS
 02546 050047        STA 2,TAC2
 02547 054050        STA 3,TAC3
 02550 060637 OVR2:  DIAC 0,37     ; GET NUMBER
 02551 060337        NIOP 37       ; RESTART ADC
 02552 030163        LDA 2,BUFAD
 02553 024064        LDA 1,ANLYZ
 02554 133000        ADD 1,2
 02555 025000        LDA 1,0,2     ; GET BUFFER ADDRESS
 02556 122400        SUB 1,0       ; COMPUTE CHANNEL NO.
 02557 040433        STA 0,TOV4
 02560 020057        LDA 0,OUSE    ; SAVE OUTPUT FLAGS
 02561 101004        MOV 0,0,SZR
 02562 000256        JMP RESTR
 02563 040424        STA 0,TOV1
 02564 020043        LDA 0,OADR
 02565 040423        STA 0,TOV2
 02566 020042        LDA 0,OFLG
 02567 040422        STA 0,TOV3
 02570 030101        LDA 2,MES3    ; OUTPUT "OVERFLOW"
 02571 006120        JSR @TEXT     ; "AT CHAN"
 02572 030103        LDA 2,MES5
 02573 006120        JSR @TEXT
 02574 020416        LDA 0,TOV4
 02575 006201        JSR @DCOUT
 02576 004314        JSR CRLF
 02577 020410        LDA 0,TOV1    ; RESTORE OUTPUT FLAGS
 02600 040057        STA 0,OUSE
 02601 020407        LDA 0,TOV2
 02602 040043        STA 0,OADR
 02603 020406        LDA 0,TOV3
 02604 040042        STA 0,OFLG
 02605 000401        JMP .+1
 02606 000256        JMP RESTR
 02607 000000 TOV1:  0
 02610 000000 TOV2:  0
 02611 000000 TOV3:  0
 02612 000000 TOV4:  0
                ;
                ;
                ;
                ; SUBROUTINES
                ;
```

```
                    ; SUBROUTINE TEXT OUTPUTS ASCII STRINGS
                    ; ENTER WITH AC2 = ADDRESS OF STRING
                    ;
02613 054415 .TEXT: STA 3,TXTR      ; SAVE RETURN ADDRESS
02614 024111        LDA 1,RUB       ; MASK = 377
02615 021000 TXT2:  LDA 0,0,2
02616 123405        AND 1,0,SNR
02617 002411        JMP @TXTR       ; RETURN
02620 004243        JSR OUT
02621 021000        LDA 0,0,2
02622 101300        MOVS 0,0        ; UNPACK NEXT CHAR
02623 123405        AND 1,0,SNR
02624 002404        JMP @TXTR       ; RETURN
02625 004243        JSR OUT
02626 151400        INC 2,2
02627 000766        JMP TXT2
02630 000000 TXTR:  0
                    ;
                    ; MESSAGE LIST
             M1:    .TXT "  ERROR "
02631 020040
02632 051105
02633 047522
02634 020122
02635 000000
             M2:    .TXT " BUFFER  SIZE  STARTING ADDRESS"
02636 041040
02637 043125
02640 042506
02641 020122
02642 051440
02643 055111
02644 020105
02645 051440
02646 040524
02647 052122
02650 047111
02651 020107
02652 042101
02653 051104
02654 051505
02655 000123
             M3:    .TXT "OVERFLOW "
02656 053117
02657 051105
02660 046106
02661 053517
02662 000040
             M4:    .TXT "  WORDS LEFT"
02663 020040
02664 047527
02665 042122
02666 020123
02667 042514
02670 052106
02671 000000
             M5:    .TXT " AT CHAN."
02672 040440
02673 020124
```

```
 0034  .MAIN
  02674 044103
  02675 047101
  02676 000056
                  M6:     .TXT " PULSE HEIGHT ANALYZER CONTROL  <015><012>   VERSIC
C"
  02677 050040
  02700 046125
  02701 042523
  02702 044040
  02703 044505
  02704 044107
  02705 020124
  02706 047101
  02707 046101
  02710 055131
  02711 051105
  02712 041440
  02713 047117
  02714 051124
  02715 046117
  02716 020040
  02717 005015
  02720 020040
  02721 053040
  02722 051105
  02723 044523
  02724 047117
  02725 030440
  02726 041455
  02727 000000
                  M7:     .TXT "                        (OCTAL)"
  02730 020040
  02731 020040
  02732 020040
  02733 020040
  02734 020040
  02735 020040
  02736 020040
  02737 020040
  02740 020040
  02741 020040
  02742 047450
  02743 052103
  02744 046101
  02745 000051
                  ;
                  ;
                  ; SUBROUTINE ERROR
                  ; ENTER WITH ERROR NUMBER IN AC0
                  ;
  02746 024203 ERROR: LDA 1.ASO
  02747 123000        ADD 1.0
  02750 040407        STA 0.ERNUM
  02751 030077        LDA 2.MES1
  02752 006120        JSR @TEXT
  02753 020404        LDA 0.ERNUM
  02754 004243        JSR OUT
  02755 004314        JSR CRLF
  02756 000377        JMP RETRN
  02757 000000 ERNUM: 0
```

```
0035  .MAIN
                    ;
                    ;
                    ; SUBROUTINE TOTAL ADDS UP ALL OF THE
                    ; BUFFER SIZES AND RETURNS THE SUM IN AC0
                    ;
02760 020141 .TOTL: LDA 0,MIN16
02761 040067        STA 0,CNT2
02762 030162        LDA 2,BUFSZ
02763 102400        SUB 0,0
02764 025000 TOTLP: LDA 1,0,2      ; GET SIZE AND
02765 125100        MOVL 1,1       ; MASK OFF BIT 0
02766 125220        MOVZR 1,1
02767 123000        ADD 1,0        ; ADD TO TOTAL
02770 151400        INC 2,2
02771 010067        ISZ CNT2
02772 000772        JMP TOTLP
02773 001400        JMP 0,3        ; RETURN
                    ;
                    ;
                    ; SUBROUTINE GTNUM  GETS I,N,M FROM THE
                    ; COMMAND LINE BUFFER
                    ;
02774 054515 .GETN: STA 3,GETRT
02775 102400        SUB 0,0
02776 040052        STA 0,CMCNT
02777 030051 GTNL2: LDA 2,CMBUF
03000 010052        ISZ CMCNT
03001 020052        LDA 0,CMCNT
03002 113000        ADD 0,2
03003 021000        LDA 0,0,2      ; GET CHAR
03004 024203        LDA 1,AS0
03005 034204        LDA 3,AS9
03006 162033        ADCZ# 3,0,SNC ; SKP IF CHAR >9
03007 106032        ADCZ# 0,1,SZC ; SKP IF CHAR >=0
03010 000767        JMP GTNL2      ; NOT A DIGIT
03011 006202        JSR @DBIN      ; DIGIT SO GET NUMBER
03012 044053        STA 1,INUM
03013 024205        LDA 1,COMA
03014 106405        SUB 0,1,SNR    ; IS CHAR A ","
03015 000412        JMP GTNL6
03016 101005        MOV 0,0,SNR    ; NO  IS IT NULL
03017 000404        JMP GTNL4
03020 102520 GTER2: SUBZL 0,0      ; NOT NULL - ERROR 2
03021 101120        MOVZL 0,0
03022 002135        JMP @ERR
03023 102400 GTNL4: SUB 0,0
03024 040054        STA 0,NNUM     ; ZERO N,M
03025 040055        STA 0,MNUM
03026 000437        JMP NUMCK
03027 020052 GTNL6: LDA 0,CMCNT    ; GET NEXT CHAR
03030 030051        LDA 2,CMBUF
03031 113000        ADD 0,2
03032 021000        LDA 0,0,2
03033 024203        LDA 1,AS0
03034 034204        LDA 3,AS9
03035 162033        ADCZ# 3,0,SNC ; IS 0<=CHAR<=9
03036 106032        ADCZ# 0,1,SZC
03037 000761        JMP GTER2     ; NO ERROR 2
03040 006202        JSR @DBIN
```

```
0036  .MAIN
 03041 044054          STA 1.NNUM
 03042 024205          LDA 1.COMA
 03043 106405          SUB 0.1.SNR   ; IS CHAR "."
 03044 000406          JMP GTNL8
 03045 101004          MOV 0.0.SZR   ; NO  IS IT NULL
 03046 000752          JMP GTER2     ; NOT NULL
 03047 102400          SUB 0.0       ; ZERO M
 03050 040055          STA 0.MNUM
 03051 000414          JMP NUMCK
 03052 020052 GTNL8:   LDA 0.CMCNT   ; GET NEXT CHAR
 03053 030051          LDA 2.CMBUF
 03054 113000          ADD 0.2
 03055 021000          LDA 0.0.2
 03056 024203          LDA 1.AS0
 03057 034204          LDA 3.AS9
 03060 162033          ADCZ# 3.0.SNC ; IS CHAR A DIGIT
 03061 106032          ADCZ# 0.1.SZC
 03062 000736          JMP GTER2     ; NO
 03063 006202          JSR @DBIN
 03064 044055          STA 1.MNUM
 03065 020053 NUMCK:   LDA 0.INUM
 03066 126520          SUBZL 1.1     ; +1
 03067 030133          LDA 2.P20     ; +20
 03070 142033          ADCZ# 2.0.SNC ; SKP IF I>20
 03071 122433          SUBZ# 1.0.SNC ; SKP IF I>=1
 03072 000726          JMP GTER2     ; BUFFER NO. INCORRECT
 03073 020056          LDA 0.CKFLG   ; SKIP IF ERROR 4
 03074 101004          MOV 0.0.SZR   ; SHOULD BE CHECKED FOR
 03075 000411          JMP NMCK2
 03076 030162          LDA 2 BUFSZ
 03077 020053          LDA 0.INUM
 03100 113000          ADD 0.2
 03101 021000          LDA 0.0.2
 03102 101102          MOVL 0.0.SZC   ; IS BIT 0 SET
 03103 000403          JMP NMCK2
 03104 020150          LDA 0 P4      ; NO  SO BUFFER
 03105 002135          JMP @ERR      ; INACTIVE - ERROR 4
 03106 102400 NMCK2:   SUB 0.0
 03107 040056          STA 0.CKFLG
 03110 002401          JMP @GETRT
 03111 000000 GETRT:   0
 03112 030051 GET:     LDA 2.CMBUF
 03113 020052          LDA 0.CMCNT
 03114 113000          ADD 0.2
 03115 021000          LDA 0.0.2
 03116 010052          ISZ CMCNT
 03117 001400          JMP 0.3
              ;
              ;
              ; SUBROUTINE DCOUT  OUTPUTS DECIMAL NUMBERS WITH
              ; LEADING ZEROS BLANKED
              ; WHEN USED WITH PUT
              ; ENTER WITH BINARY NO. IN ACO
              ;
 03120 054407 .DCOT:   STA 3.DCRET   ; SAVE RETURN ADDRESS
 03121 126400          SUB 1.1
 03122 044060          STA 1.ZFLAG   ; SET UP FLAG AND
 03123 044070          STA 1.CNT3
 03124 105000          MOV 0.1
```

```
0037  .MAIN
 03125 006125         JSR @BIND
 03126 002401         JMP @DCRET
 03127 000000 DCRET:  0
                ;
 03130 054423 PUT:    STA 3,PTRET
 03131 024124         LDA 1,PLUS
 03132 106405         SUB 0,1,SNR   ; SKP IF CHAR NOT= "+"
 03133 000415         JMP PUTL6
 03134 024060         LDA 1,ZFLAG
 03135 125004         MOV 1,1,SZR
 03136 000413         JMP PUTL8
 03137 024203         LDA 1,ASO
 03140 106405         SUB 0,1,SNR   ; SKP IF CHAR NOT= "0"
 03141 000403         JMP PUTL2
 03142 010060         ISZ ZFLAG
 03143 000406         JMP PUTL8
 03144 010070 PUTL2:  ISZ CNT3
 03145 024070         LDA 1,CNT3
 03146 034147         LDA 3,P5
 03147 1643         SUBZ 3,1,SNC
 03150 020142 PUTL6:  LDA 0,SPACE   ; BLANK OUT CHAR
 03151 004243 PUTL8:  JSR OUT
 03152 002401         JMP @PTRET
 03153 000000 PTRET:  0
                ;
                ;
                ; SUBROUTINE TAB  OUTPUTS N SPACES
                ; WHERE AC0 = N
                ;
 03154 054407 .TAB:   STA 3 TABRT   ; SAVE RETURN ADDRESS
 03155 104400         NEG 0,1
 03156 020142 TBLOP:  LDA 0,SPACE
 03157 004243         JSR OUT
 03160 125404         INC 1,1,SZR
 03161 000775         JMP TBLOP
 03162 002401         JMP @TABRT
 03163 000000 TABRT:  0
                ;
                ;
                ; SUBROUTINE CHECK
                ;
 03164 020054 .CHK:   LDA 0,NNUM
 03165 024055         LDA 1,MNUM
 03166 106432         SUBZ# 0,1,SZC ; SKP IF N>M
 03167 000403         JMP CHKL2
 03170 020147 CKER5:  LDA 0,P5      ; ERROR 5
 03171 002135         JMP @ERR
 03172 020053 CHKL2:  LDA 0,INUM
 03173 030162         LDA 2,BUFSZ
 03174 113000         ADD 0,2
 03175 021000         LDA 0,0,2     ; GET SIZE OF BUFFER I
 03176 101100         MOVL 0,0
 03177 101220         MOVZR 0,0     ; MASK OFF BIT 0
 03200 125400         INC 1,1
 03201 122433         SUBZ# 1,0,SNC
 03202 000766         JMP CKER5     ; M>SIZE+1
 03203 001400         JMP 0,3       ; RETURN
                ;
                ;
```

```
                     ; I/O CONVERSION ROUTINES
                     ;
 03204 054431 .BIND:   STA 3,.ED03       ; SAVE RETURN
 03205 050427          STA 2,.ED02       ; SAVE AC2
 03206 040425          STA 0,.ED00       ; SAVE AC0
 03207 034440          LDA 3,.ED30       ; ADDRESS OF POWER OF TEN TABLE
 03210 054434          STA 3,.ED10       ; INITIALIZE POINTER
 03211 020124          LDA 0,PLUS        ; NO, IT IS POSITIVE; GET PLUS
 03212 044433 .ED97:   STA 1,.ED11       ; SAVE N
 03213 006041          JSR @.ED40        ; PUT OUT SIGN OR DIGIT
 03214 024431          LDA 1,.ED11       ; GET CURRENT VALUE OF N
 03215 036427          LDA 3,@.ED10      ; GET CURRENT  POWER OF TEN
 03216 010426          ISZ .ED10         ; BUMP POINTER
 03217 161005          MOV 3,0,SNR
 03220 000407          JMP .ED98         ; PUT OUT NULL
 03221 020203          LDA 0,ASO         ; GET ASCII "0"
 03222 166422 .ED99:   SUBZ 3,1,SZC      ; DOES POWER OF TEN GO IN?
 03223 101401          INC 0,0,SKP       ; YES, BUMP RESULT DIGIT
 03224 167001          ADD 3,1,SKP       ; NO, RESTORE PREVIOUS VALUE
 03225 000775          JMP .ED99         ; CONTINUE SUBTRACTING
 03226 000764          JMP .ED97         ; PUT OUT DIGIT
 03227 006041 .ED98:   JSR @.ED40        ; PUT OUT NULL WORD
 03230 020403          LDA 0,.ED00       ; RESTORE AC0
 03231 030403          LDA 2,.ED02       ; RESTORE AC2
 03232 002403          JMP @.ED03        ; RETURN
 03233 000000 .ED00:   0                 ; SAVE AC0
 03234 000000 .ED02:   0                 ; SAVE AC2
 03235 000000 .ED03:   0                 ; SAVE AC3
       000012          .RDX 10
 03236 023420 .ED05:   10000             ; POWER OF TEN TABLE
 03237 001750          1000              ; 10**3
 03240 000144          100               ; 10**2
 03241 000012          10                ; 10**1
 03242 000001          1                 ; 10**0
 03243 000000          0                 ; END OF TABLE INDICATION
       000010          .RDX 8
 03244 000000 .ED10:   0                 ; ADDRESS OF CURRENT POWER OF
                                         ; TEN ENTRY
 03245 000000 .ED11:   0                 ; RUNNING SUM WORD
 03246 000055 .ED20:   "-                ; ASCII "-"
 03247 003236 .ED30:   .ED05             ; ADDRESS OF POWER OF TEN TABLE
       000041 .ED40=.PUT                 ; PAGE ZERO PUT CHARACTER
                                         ; ROUTINE ADDRESS
                     ;
                     ;
                     ; DOUBLE PRECISION BINARY TO DECIMAL CONVERSION
                     ;
 03250 054454 .DBD:    STA 3,.FD03       ; SAVE RETURN
 03251 040452          STA 0,.FD00       ; SAVE AC0
 03252 020477          LDA 0,.FD30       ; POINT TO HIGH ORDER POWER IN
                                         ; TABLE
 03253 040503          STA 0,.FD12
 03254 020124          LDA 0,PLUS        ; ASSUME "+"
 03255 125113          MOVL# 1,1,SNC
 03256 000405          JMP .FD99         ; YES, WAS POSITIVE
 03257 150404          NEG 2,2,SZR       ; NO, NEGATIVE
 03260 124001          COM 1,1,SKP
 03261 124400          NEG 1,1
 03262 020475          LDA 0,.FD21       ; GET "-"
```

```
0039  .MAIN
 03263 044470 .FD99:  STA 1..FD10        ; SAVE ABS(NUMBER)
 03264 050470         STA 2..FD10+1
 03265 006041         JSR @.FD40         ; PUT OUT SIGN OR DIGIT
 03266 024465         LDA 1..FD10        ; RESTORE ABS(NUMBER)
 03267 030465         LDA 2..FD10+1
 03270 020470         LDA 0..FD22        ; GET OCTAL 57
 03271 040464         STA 0..FD11        ; COUNT IT UP IN STORAGE
 03272 034464         LDA 3..FD12        ; CURRENT POINTER TO POWER OF
                                         ; 10 TABLE
 03273 021401 .FD98:  LDA 0.1.3          ; LOW ORDER WORD
 03274 101005         MOV 0.0.SNR        ; TEST FOR END OF TABLE
 03275 000423         JMP .FD97          ; DONE
 03276 112420         SUBZ 0.2
 03277 021400         LDA 0.0.3          ; HIGH ORDER WORD
 03300 101003         MOV 0.0.SNC
 03301 106001         ADC 0.1.SKP
 03302 106400         SUB 0.1
 03303 010452         ISZ .FD11          ; COUNT UP DIGIT
 03304 125113         MOVL# 1.1.SNC      ; TEST FOR <0
 03305 000766         JMP .FD98          ; KEEP SUBTRACTING

 03306 021401         LDA 0.1.3          ; RESTORE POSITIVE VALUE
 03307 113022         ADDZ 0.2.SZC
 03310 125400         INC 1.1
 03311 021400         LDA 0.0.3
 03312 107000         ADD 0.1
 03313 175400         INC 3.3            ; BUMP AC3 TO NEXT TABLE ENTRY
 03314 175400         INC 3.3
 03315 054441         STA 3..FD12
 03316 020437         LDA 0..FD11        ; GET DIGIT
 03317 000744         JMP .FD99          ; PUT IT OUT
 03320 006041 .FD97:  JSR @.FD40         ; PUT OUT NULL
 03321 020402         LDA 0..FD00        ; RESTORE AC0
 03322 002402         JMP @.FD03         ; RETURN
 03323 000000 .FD00:  0                  ; SAVE AC0
 03324 000000 .FD03:  0                  ; SAVE RETURN
 03325 035632 .FD05:  35632              ; 10**9
 03326 145000         145000
 03327 002765         2765               ; 10**8
 03330 160400         160400
 03331 000230         230                ; 10**7
 03332 113200         113200
 03333 000017         17                 ; 10**6
 03334 041100         41100
 03335 000001         1                  ; 10**5
 03336 103240         103240
       000012         .RDX 10
 03337 000000         0                  ; 10**4
 03340 023420         10000
 03341 000000         0                  ; 10**3
 03342 001750         1000
 03343 000000         0                  ; 10**2
 03344 000144         100
 03345 000000         0                  ; 10**1
 03346 000012         10
 03347 000000         0                  ; 10**0
 03350 000001         1
 03351 003325 .FD30:  .FD05    ; POINTER TO CONVERSION TABLE
 03352 000000         0        ; END OF TABLE INDICATION
```

```
0040   .MAIN
       000010         .RDX 8
       000002 .FD10:  .BLK 2             ; SAVE CURRENT DOUBLE WORD
 03355 000000 .FD11:  0                  ; COUNT UP DIGIT WORD
 03356 000000 .FD12:  0                  ; POINTER TO POWER OF TEN ENTRY
 03357 000055 .FD21:  "-                 ; ASCII "-"
 03360 000057 .FD22:  57                 ; ASCII "0" -1
       000041 .FD40=.PUT                 ; PAGE 0 PUT CHARACTER ADDRESS
              ;
              ;
              ; SINGLE PRECISION DECIMAL TO BINARY CONVERSION
              ;
 03361 054442 .DBIN:  STA 3,.EC03        ; SAVE AC3
 03362 050440         STA 2,.EC02        ; SAVE AC2
 03363 102400         SUB 0,0
 03364 040440         STA 0,.EC10        ; CLEAR SIGN WORD
 03365 040440         STA 0,.EC11        ; CLEAR SUM WORD
 03366 006040         JSR @.EC40         ; GET A CHARACTER
 03367 024124         LDA 1,PLUS         ; TEST FOR "+"
 03370 106405         SUB 0,1,SNR
 03371 000405         JMP .EC97          ; YES
 03372 024434         LDA 1,.EC21        ; NO, TEST FOR "-"
 03373 106404         SUB 0,1,SZR
 03374 000403         JMP .EC96          ; NO EXPLICIT SIGN
 03375 010427         ISZ .EC10          ; SET FLAG WORD FOR NEGATIVE
                                         ; NUMBER
 03376 006040 .EC97:  JSR @.EC40         ; GET ANOTHER CHARACTER
 03377 024203 .EC96:  LDA 1,AS0          ; ASCII "0"
 03400 030204         LDA 2,AS9          ; ASCII "9"
 03401 142033         ADCZ# 2,0,SNC      ; SKIP IF > 9
 03402 106032         ADCZ# 0,1,SZC      ; SKIP IF >= 0
 03403 000406         JMP .EC95          ; NOT A DIGIT, THERFORE A BREAK
                                         ; CHARACTER
 03404 122400         SUB 1,0            ; REDUCE DIGIT TO 0-9 BINARY
                                         ; RANGE
 03405 024420         LDA 1,.EC11        ; SUM WORD
 03406 004406         JSR .EC50          ; MULTIPLY BY 10 AND ADD
 03407 044416         STA 1,.EC11        ; SAVE SUM
 03410 000766         JMP .EC97          ; GET NEXT CHARACTER

 03411 024414 .EC95:  LDA 1,.EC11        ; RESULT TO AC1
 03412 030410         LDA 2,.EC02        ; RESTORE AC2
 03413 002410         JMP @.EC03
              ; ROUTINE TO MULTIPLY AC1 BY 10 AND ADD AC0
 03414 131120 .EC50:  MOVZL 1,2          ; N*2
 03415 151120         MOVZL 2,2          ; N*4
 03416 147000         ADD 2,1            ; N*5
 03417 125120         MOVZL 1,1          ; N*5*2 = N*10
 03420 107000         ADD 0,1            ; ADD AC0
 03421 001400         JMP 0,3            ; SUCCESS RETURN
 03422 000000 .EC02:  0                  ; SAVE AC2
 03423 000000 .EC03:  0                  ; SAVE AC3
 03424 000000 .EC10:  0                  ; FLAG WORD FOR SIGN OF RESULT
 03425 000000 .EC11:  0                  ; RUNNING SUM WORD
 03426 000055 .EC21:  "-                 ; ASCII "-"
                                         ; ENTRY
       000040 .EC40=.GET                 ; ADDRESS OF GET CHARACTER
                                         ; ROUTINE
       000041 .EC41=.PUT                 ; ADDRESS OF PUT CHARACTER
                                         ; ROUTINE
```

```
                  ;
                  ;
                  ; SINGLE PRECISION BINARY TO OCTAL CONVERSION
                  ;
 03427 054774 .BINO:  STA 3,.EC03       ; SAVE RETURN
 03430 050772         STA 2,.EC02       ; *SAVE AC2
 03431 152621         SUBZR 2,2,SKP     ; 100000 TO AC2
 03432 146401 .EF99:  SUB 2,1,SKP       ; DECREASE CURRENT DIGIT BY 1
 03433 020420 .EF98:  LDA 0,.EF20       ; GET OCTAL 57
 03434 101400         INC 0,0           ; FORM ASCII OUTPUT DIGIT
 03435 146533         SUBZL# 2,1,SNC    ; - IMPLIES DIGIT COMPLETE
 03436 000774         JMP .EF99         ; NOT DONE, SUBTRACT 1 FROM
                                        ; CURRENT DIGIT
 03437 050413         STA 2,.EF10       ; SAVE SUBTRACT CONSTANT
 03440 004243         JSR OUT           ; PUT OUT A DIGIT
 03441 030411         LDA 2,.EF10       ; RESTORE SUBTRACT CONSTANT
 03442 151220         MOVZR 2,2         ; POSITION "1" FOR NEXT
                                        ; OCTAL DIGIT
 03443 151220         MOVZR 2,2
 03444 151224         MOVZR 2,2,SZR
 03445 000766         JMP .EF98         ; NOT DONE
 03446 141000         MOV 2,0
 03447 004243         JSR OUT           ; PUT OUT NULL CHARACTER
 03450 030752         LDA 2,.EC02       ; *RESTORE AC2
 03451 002752         JMP @.EC03        ; RETURN

 03452 000000 .EF10:  0                 ; SAVE LOCATION FOR
                                        ; SUBTRACT CONSTANT
 03453 000057 .EF20:  57                ; ASCII CONSTANT

       000041 .EF40=.PUT                 ; PAGE ZERO ADDRESS OF
                                        ; PUT CHARACTER ROUTINE
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ; BUFFERS
                  ;
                  ; COMMAND BUFFER
                  ;
       000024 .CMB:  .BLK 24
                  ;
                  ; INSTRUCTION CODE BUFFER
                  ;
                  .ASCI: .TXT "TYPE".
 03500 054524
 03501 042520
 03502 000000
 03503 001422              TYPE
                           .TXT "T<000><000><000><000>"
 03504 000124
 03505 000000
 03506 000000
 03507 001422              TYPE
                           .TXT "PUNCH"
 03510 052520
 03511 041516
```

```
0042  .MAIN
 03512 000110
 03513 001503          PUNCH
                       .TXT "P<000><000><000><000>"
 03514 000120
 03515 000000
 03516 000000
 03517 001503          PUNCH
                       .TXT "READ"
 03520 042522
 03521 042101
 03522 000000
 03523 001571          READ
                       .TXT "R<000><000><000><000>"
 03524 000122
 03525 000000
 03526 000000
 03527 001571          READ
                       .TXT "CHNGE"
 03530 044103
 03531 043516
 03532 000105
 03533 001650          CHNGE
                       .TXT "C<000><000><000><000>"
 03534 000103
 03535 000000
 03536 000000
 03537 001650          CHNGE
                       .TXT "SUM<000><000>"
 03540 052523
 03541 000115
 03542 000000
 03543 001773          SUM
                       .TXT "HALT"
 03544 040510
 03545 052114
 03546 000000
 03547 002036          HLT
                       .TXT "H<000><000><000><000>"
 03550 000110
 03551 000000
 03552 000000
 03553 002036          HLT
                       .TXT "CONT"
 03554 047503
 03555 052116
 03556 000000
 03557 002047          CONT
                       .TXT "MAX<000><000>"
 03560 040515
 03561 000130
 03562 000000
 03563 002103          MAX
                       .TXT "M<000><000><000><000>"
 03564 000115
 03565 000000
 03566 000000
 03567 002103          MAX
                       .TXT "ZERO"
 03570 042532
```

```
0043  .MAIN
 03571 047522
 03572 000000
 03573 002157         ZERO
                      .TXT "Z<000><000><000><000>"
 03574 000132
 03575 000000
 03576 000000
 03577 002157         ZERO
                      .TXT "BUFF"
 03600 052502
 03601 043106
 03602 000000
 03603 002361         BUFF
                      .TXT "B<000><000><000><000>"
 03604 000102
 03605 000000
 03606 000000
 03607 002361         BUFF
                      .TXT "STRT"
 03610 052123
 03611 052122
 03612 000000
 03613 002062         STRT
                      .TXT "CLEAR"
 03614 046103
 03615 040505
 03616 000122
 03617 002243         CLEAR
                      .TXT "SIZE"
 03620 044523
 03621 042532
 03622 000000
 03623 002464         SIZE
                      .TXT "DISP"
 03624 044504
 03625 050123
 03626 000000
 03627 002530         DISP
                 ;
                 ; BUFFER SIZES AND ADDRESSES
                 ;
       000020 .BFSZ: .BLK 20
       000020 .BFAD: .BLK 20
                 ;
                 ;  DISPLAY BUFFER
                 ;
       001000 DSBUF: .BLK 1000 .
                 ;
                 ; STORAGE BUFFER STARTS AT AN
                 ; EVEN MULTIPLE OF 8
                 ;
       004677   .LOC .&177770+7
 04677 000000 .BFST!  0
                 ;
       000400 .END START
```

```
  0044   .MAIN
A        000121
ACTIV    000061
ALT      000110
ANLYZ    000064
AS0      000203
AS9      000204
ASC1     000206
ASC2     000207
ASC3     000210
ASCII    000136
AST      000073
BADR     000164
BFL10    002450
BIND     000125
BINO     000215
BIT8     000175
BKSL     000132
BLANK    001560
BLNK     001570
BLOOP    001564
BUFAD    000163
BUFEN    000063
BUFF     002361
BUFL2    002374
BUFL4    002430
BUFL6    002433
BUFL8    002441
BUFST    000062
BUFSZ    000162
CAN      000112
CHAR     000276
CHECK    000177
CHKL2    003172
CHNER    001771
CHNGE    001650
CHNL2    001673
CHNL4    001736
CHNL6    001754
CHNWT    001702
CKER5    003170
CKFLG    000056
CL       000322
CLEAR    002243
CLKNT    000236
CLL12    002347
CLL2     002252
CLL3     002256
CLL4     002262
CLL5     002266
CLL6     002302
CLTTO    000312
CMAGN    001411
CMBUF    000051
CMCNT    000052
CML1     001357
CML2     001363
CML3     001366
CML4     001367
CMLOP    001375
```

```
  0045   .MAIN
CMND     001323
CNT1     000066
CNT2     000067
CNT3     000070
CNTL     000114
CNTLR    000106
COMA     000205
CONL2    002055
CONT     002047
CP1      001171
CP3      001172
CP5      001173
CR       000071
CRLF     000314
DBD      000126
DBIN     000202
DCFLG    000223
DCINT    000217
DCNT     000221
DCOUT    000201
DCRET    003127
DISB     000224
DISBF    000225
DISP     002530
DISSZ    000227
DLOOP    000232
DSADR    000231
DSBUF    003670
DSLOP    000233
DSNUM    000235
DSOUT    000002
DSPAD    000234
ECHO     000275
ERNUM    002757
ERR      000135
ERROR    002746
ESC      000107
GET      003112
GETRT    003111
GTER2    003020
GTNL2    002777
GTNL4    003023
GTNL6    003027
GTNL8    003052
GTNUM    000131
HLT      002036
IN       001635
INER     001321
INL10    001262
INL11    001265
INL12    001301
INL2     001236
INL4     001241
INL6     001246
INL8     001255
INR      001647
INTR     000434
INUM     000053
L        000122
```

```
  0046   .MAIN
LF       000072
LGSFT    001207
LNH      001226
LOG      001101
LOGA     001166
LOGD2    001124
LOGL2    001103
LOGL4    001110
LOGZ     001167
LOOP2    000423
M1       002631
M2       002636
M3       002656
M4       002663
M40      000065
M5       002672
M6       002677
M7       002730
MAIN     000453
MASK1    000113
MASK2    000167
MASK3    000170
MASK4    000171
MASK5    000172
MASK6    000173
MASK7    000174
MAX      002103
MAXCH    002156
MAXL2    002112
MAXL4    002126
MAXL6    002137
MAXN     002155
MES1     000077
MES2     000100
MES3     000101
MES4     000102
MES5     000103
MES6     000104
MES7     000105
MIN16    000141
MIN27    000143
MIN4     000137
MIN5     000140
MIN8     000165
MNL1     000464
MNL10    000547
MNL12    000561
MNL13    000604
MNL14    000614
MNL16    000630
MNL2     000467
MNL20    000664
MNL22    000671
MNL4     000507
MNL40    000703
MNL42    000712
MNL44    000713
MNL46    000734
MNL49    000767
```

```
  0047   .MAIN
MNL50    001014
MNL6     000516
MNL8     000536
MNUM     000055
MOVE     002326
MPY      001174
MPYL2    001177
MPYR     001206
NMCK2    003106
NNUM     000054
NUMCK    003065
OADR     000043
OFLG     000042
ORET     000044
OUSE     000057
OUT      000243
OUTR     000277
OVER     000076
OVR2     002550
P10      000153
P12      000154
P15      000155
P2       000144
P20      000133
P24      000134
P3       000145
P360     000176
P4       000150
P5       000147
P512     000161
P550     000241
P6       000146
P60      000160
P8       000151
P9       000152
PLUS     000124
PNTA     000230
PNTF     001045
PNTR     000760
POINT    000743
POS24    000156
POS26    000157
PTRET    003153
PUNCH    001503
PUNL4    001510
PUT      003130
PUTL2    003144
PUTL6    003150
PUTL8    003151
R        000115
RDL2     001603
RDL4     001633
READ     001571
RESTR    000256
RETM     000000
RETRN    000377
RTCDS    000346
RTCL4    000330
RTCLK    000323
```

```
  0048   .MAIN
RTCRT    000356
RTRN     000416
RTUPD    000363
RUB      000111
RUNT     000240
SHFT     002314
SIZE     002464
SPACE    000142
ST       000116
START    000400
STRL2    002071
STRT     002062
SUM      001773
SUMLP    002007
SW9      001502
SZE      000123
SZL2     002476
TAB      000166
TABRT    003163
TAC0     000045
TAC1     000046
TAC2     000047
TAC3     000050
TAC00    000211
TAC01    000212
TAC02    000213
TAC03    000214
TBLOP    003156
TEXT     000120
TOTAL    000117
TOTL1    000127
TOTL2    000130
TOTLP    002764
TOV1     002607
TOV2     002610
TOV3     002611
TOV4     002612
TPL4     001432
TPL5     001444
TTIN     000075
TTOUT    000074
TTTY     000272
TXT2     002615
TXTR     002630
TYPE     001422
TYPEN    001475
TYPL6    001450
UPDAT    000642
WAIT     001042
XSCL     000220
XSHFT    000222
YPL2     001062
YPONT    001046
YPRET    001100
YPSET    001067
YSCL     000216
ZERET    002242
ZERI     002221
ZERL2    002163
```

```
   0049   .MAIN
 ZERL4    002204
 ZERL6    002213
 ZERL8    002235
 ZERO     002157
 ZERR2    002201
 ZFLAG    000060
 ZSQ      001170
 .ASCI    003500
 .BFAD    003650
 .BFST    004677
 .BFSZ    003630
 .BIND    003204
 .BINO    003427
 .CHK     003164
 .CKL2    000200
 .CMB     003454
 .CONT    000452
 .DBD     003250
 .DBIN    003361
 .DCOT    003120
 .DSBF    000226
 .EC02    003422
 .EC03    003423
 .EC10    003424
 .EC11    003425
 .EC21    003426
 .EC40    000040
 .EC41    000041
 .EC50    003414
 .EC95    003411
 .EC96    003377
 .EC97    003376
 .ECHO    000264
 .ED00    003233
 .ED02    003234
 .ED03    003235
 .ED05    003236
 .ED10    003244
 .ED11    003245
 .ED20    003246
 .ED30    003247
 .ED40    000041
 .ED97    003212
 .ED98    003227
 .ED99    003222
 .EF10    003452
 .EF20    003453
 .EF40    000041
 .EF98    003433
 .EF99    003432
 .FD00    003323
 .FD03    003324
 .FD05    003325
 .FD10    003353
 .FD11    003355
 .FD12    003356
 .FD21    003357
 .FD22    003360
 .FD30    003351
```

```
  0050  .MAIN
.FD40   000041
.FD97   003320
.FD98   003273
.FD99   003263
.GET    000040
.GETN   002774
.MAIN   000242
.OVER   002544
.PUT    000041
.TAB    003154
.TEXT   002613
.TOTL   002760
.TOUT   000300
.TTIN   001216
.WAIT   001044
.YPNT   000237
.ZER    002161
```

APPENDIX B

Program Modification Example

This section illustrates the general usefulness of NOVADC by describing a set of software changes that can be made to turn the PHA into a time interval analyzer. The particular experiment that spawned this trial was the attempt to track down a very low intensity background component that was appearing in a ^{3}He neutron spectrometer. The count rate from this peak was only about 17 per hour and it was thought that an analysis of the interarrival time distribution would show whether the source was purely random as in a nuclear process or whether regularities pointed to electronic noise in the processing circuitry of this particular experiment.

Basically the only hardware required was something to interrupt the computer every time an event of the appropriate energy was detected. This was done simply by putting an SCA window around the peak and feeding this SCA output to the ADC. Software then set up the interface to interrupt the computer whenever an event came into the ADC. This interrupt gave control to the ADC overflow service routine. This routine was completely modified to measure the time between events using the real time clock. This was done by reading, then zeroing, the run time for the buffer that was being used. Existing software, including the real time clock service routine updated the run time so that only change required was to increase the update frequency by changing the real timer loop count (location 241). The time between events was then used as a channel number and the contents of that channel were incremented, producing a histogram of probability vs. time interval in the buffer.

Only 21 locations were changed so the majority of NOVADC remained the same and could be used to display and output the data in the normal fashion. A list of the required changes is given on the following page.

Incidentally, the distribution was found to be exponential indicating a statistically random source for the background component.

```
                                .TITL ADCPATCH
                        ; PATCH TO ALLOW TIME SPECTRUM ANALYSIS
                        ; WITH THE NOVA PHA RUNNING UNDER NOVADC V-3C
         000241                 .LOC 241
 00241   000074                 74
         002101                 .LOC 2101
 02101   061137                 DOAS 0,37
         002550                 .LOC 2550
 02550   034437                 LDA 3 TOV1          ; GET ADR OF CHAN 1
 02551   175400                 INC 3 3
 02552   025400                 LDA 1,0,3           ; GET CHAN 1
 02553   102400                 SUB 0,0
 02554   041400                 STA 0,0,3           ; CLEAR CHAN 1
 02555   020433                 LDA 0,TOV2          ; 3
 02556   030433                 LDA 2,TOV3          ; 4095
 02557   106433                 SUBZ# 0,1,SNC
 02560   105000                 MOV 0,1
 02561   146432                 SUBZ# 2,1,SZC
 02562   145000                 MOV 2,1
 02563   030424                 LDA 2,TOV1
 02564   133000                 ADD 1,2
 02565   011000                 ISZ 0,2             ; INC APPROPRIATE CHAN
 02566   060137                 NIOS 37             ; RESTART ADC
 02567   000254                 JMP 254
                        ;
         002607                 .LOC 2607
 02607   026700  TOV1:          26700
 02610   000003  TOV2:          3
 02611   007777  TOV3:          7777
                                .END
```

APPENDIX C

Detailed Logic Diagrams

The following schematics are included as an aid to software designers who wish to fully utilize the capabilities of the ADC and graphics display interfaces. For complete wiring diagrams and updates consult the group technician (K. Chin, Dept. of Physics, Room 120, Nuclear Research Building).

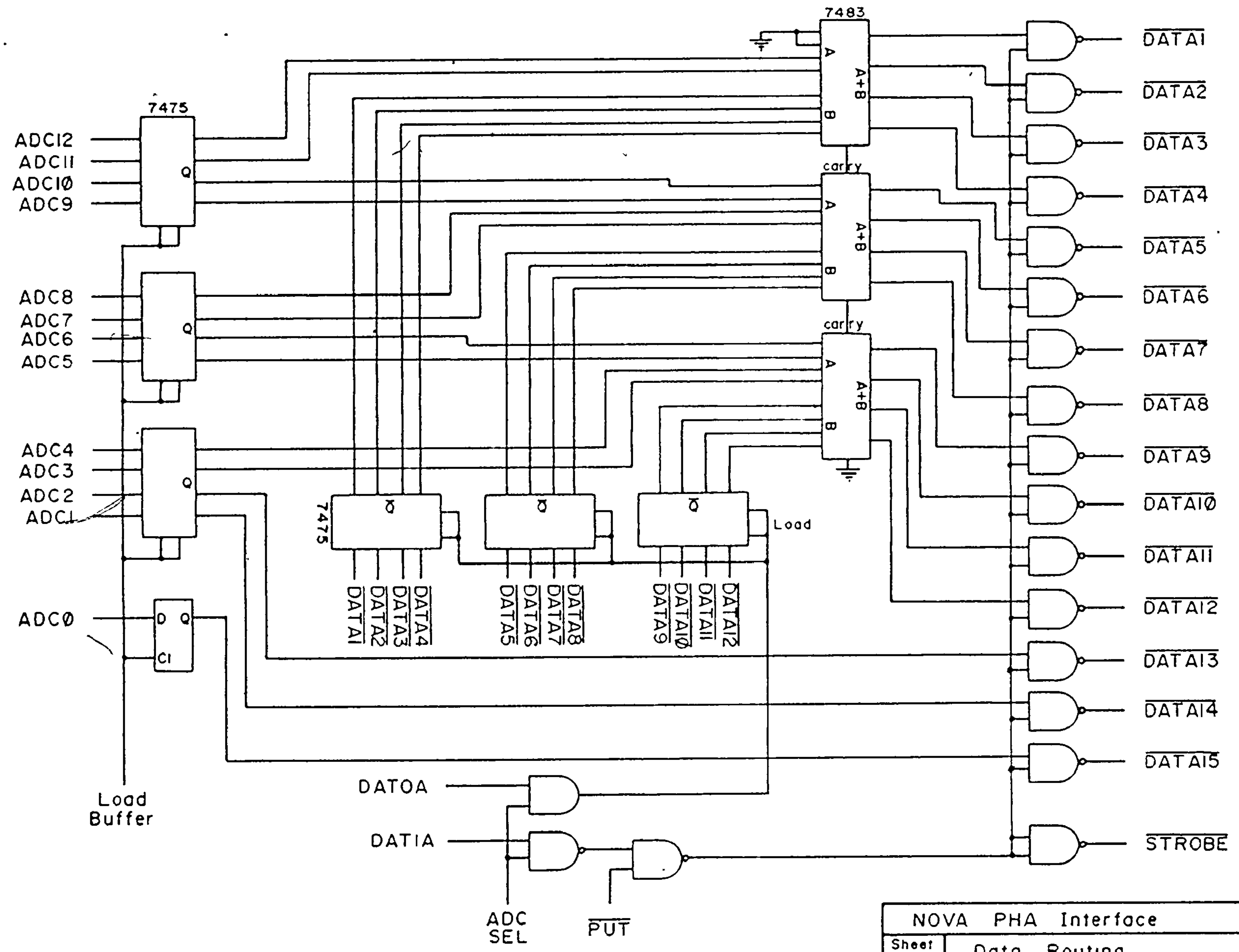

7475
7483
ADC12
ADC11
ADC1Ø
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADCØ
Q
D
Cl
Load Buffer
A
B
A+B
carry
Load
$\overline{Q}$
$\overline{\text{DATA1}}$
$\overline{\text{DATA2}}$
$\overline{\text{DATA3}}$
$\overline{\text{DATA4}}$
$\overline{\text{DATA5}}$
$\overline{\text{DATA6}}$
$\overline{\text{DATA7}}$
$\overline{\text{DATA8}}$
$\overline{\text{DATA9}}$
$\overline{\text{DATA1Ø}}$
$\overline{\text{DATA11}}$
$\overline{\text{DATA12}}$
$\overline{\text{DATA13}}$
$\overline{\text{DATA14}}$
$\overline{\text{DATA15}}$
$\overline{\text{STROBE}}$
DATOA
DATIA
ADC SEL
$\overline{\text{PUT}}$

FIG. C1

NOVA PHA Interface	
Sheet 2 of 4	Data Routing
	Designed by G Cormick Sept 75

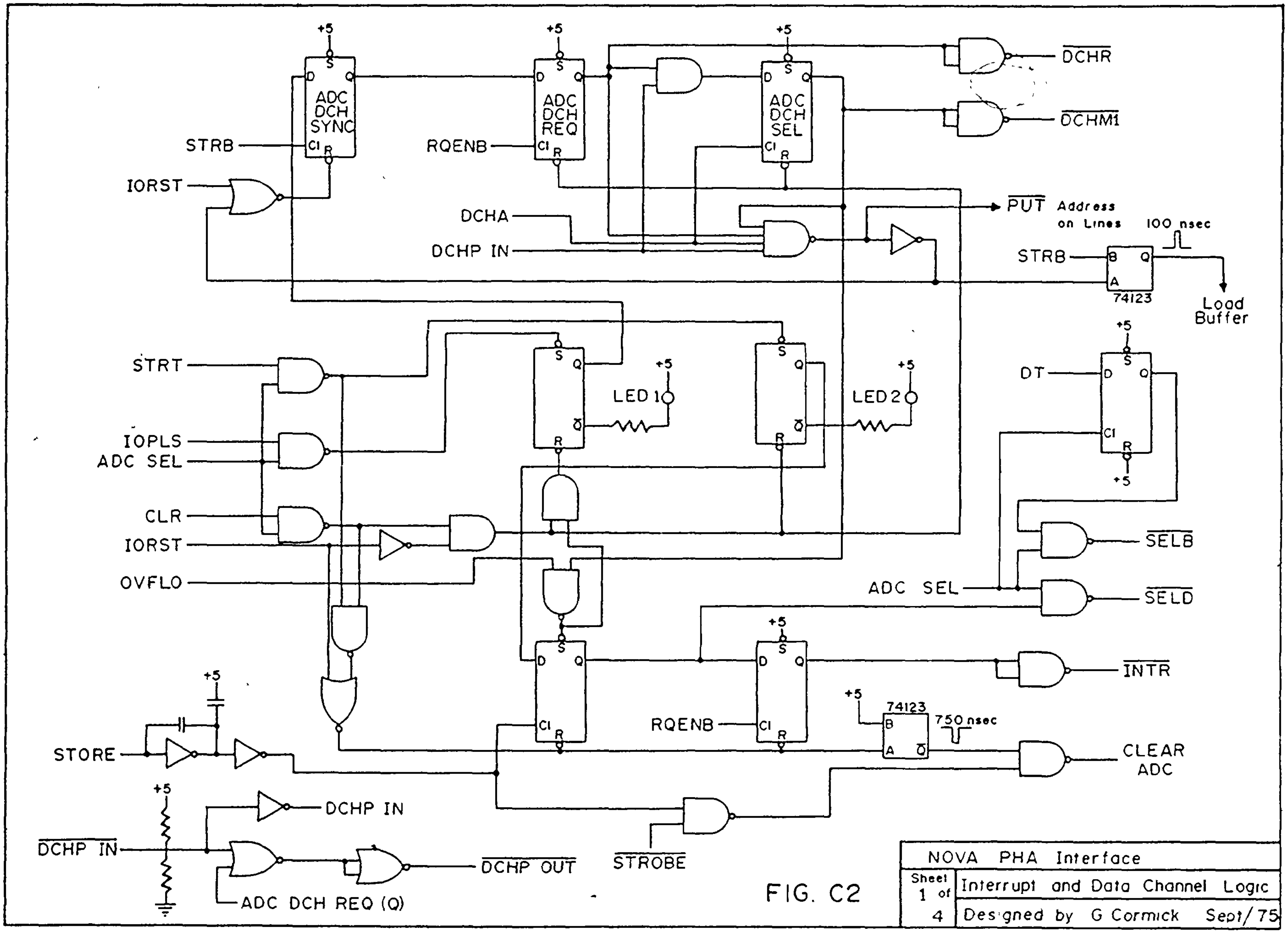


FIG. C2

NOVA PHA Interface	
Sheet 1 of 4	Interrupt and Data Channel Logic
	Designed by G Cormick Sept/75

Bin Connections Viewed From the Back

Amphenol Pin #	A	Top	B
	Gnd	43	Gnd
		42	$\overline{\text{DATA0}}$
13	ADC12	41	$\overline{\text{DATA1}}$
12	ADC11	40	$\overline{\text{DATA2}}$
11	ADC10	39	$\overline{\text{DATA3}}$
10	ADC9	38	$\overline{\text{DATA4}}$
9	ADC8	37	$\overline{\text{DATA5}}$
8	ADC7	36	$\overline{\text{DATA6}}$
7	ADC6	35	$\overline{\text{DATA7}}$
6	ADC5	34	DATOA
5	ADC4	33	DATOB
4	ADC3	32	DATIA
3	ADC2	31	DATIB
2	ADC1	30	STRT
1	ADC0	29	CLR
		28	IOPLS
15	STORE	27	IORST
16	CLEAR	26	$\overline{\text{SELB}}$
14	DT	25	$\overline{\text{SELD}}$
		24	$\overline{\text{INTR}}$
		23	RQENB
	+5 V.	22	+5 V.
		21	DCHA
		20	$\overline{\text{DCHB}}$
		19	$\overline{\text{DCHM10}}$
		18	$\overline{\text{DCHM11}}$
		17	DCHI
		16	$\overline{\text{DATA8}}$
		15	$\overline{\text{DATA9}}$
	$\overline{\text{DCHP IN}}$	14	$\overline{\text{DATA10}}$
	$\overline{\text{DCHP OUT}}$	13	$\overline{\text{DATA11}}$
		12	$\overline{\text{DATA12}}$
	$\overline{\text{INTP IN}}$	11	$\overline{\text{DATA13}}$
	$\overline{\text{INTP OUT}}$	10	$\overline{\text{DATA14}}$
		9	$\overline{\text{DATA15}}$
		8	DCHO
	-12 V.	7	-12 V.
	DATIC	6	DEV SEL
	DATOC	5	
	INTA	4	
	OVFLO	3	
	MSKO	2	$\overline{\text{STROBE}}$
	Gnd	1	Gnd

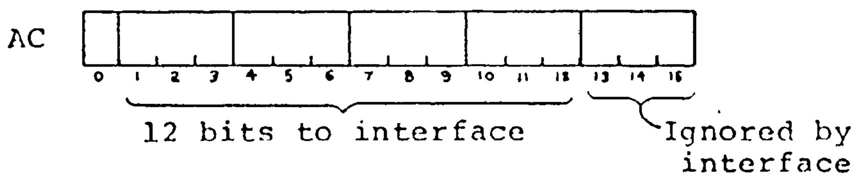
Bottom

FIG. C3

ADC Interface	
sheet 3 of 4	Back Panel Connections
	Designed by G. Cormick Sept/75

Device Code ADC = 37

NIOP ADC	060337	Enable ADC for Data Channel Accesses
NIOS ADC	060137	Enable ADC for Accumulator I/O
NIOC ADC	060237	Disable ADC
IORST	062677	Disable ADC and Reset all Devices
DOA 0,ADC	061037	Output Base Address from Accumulator 0 bits 1 - 15



DIA 0,ADC	060437	Input Channel Address to Accumulator 0 bits 1 - 15

FIG. C4

NOVA ADC Interface	
sheet 4 of 4	Instruction Set
	Designed by G. Cormick Sept/75

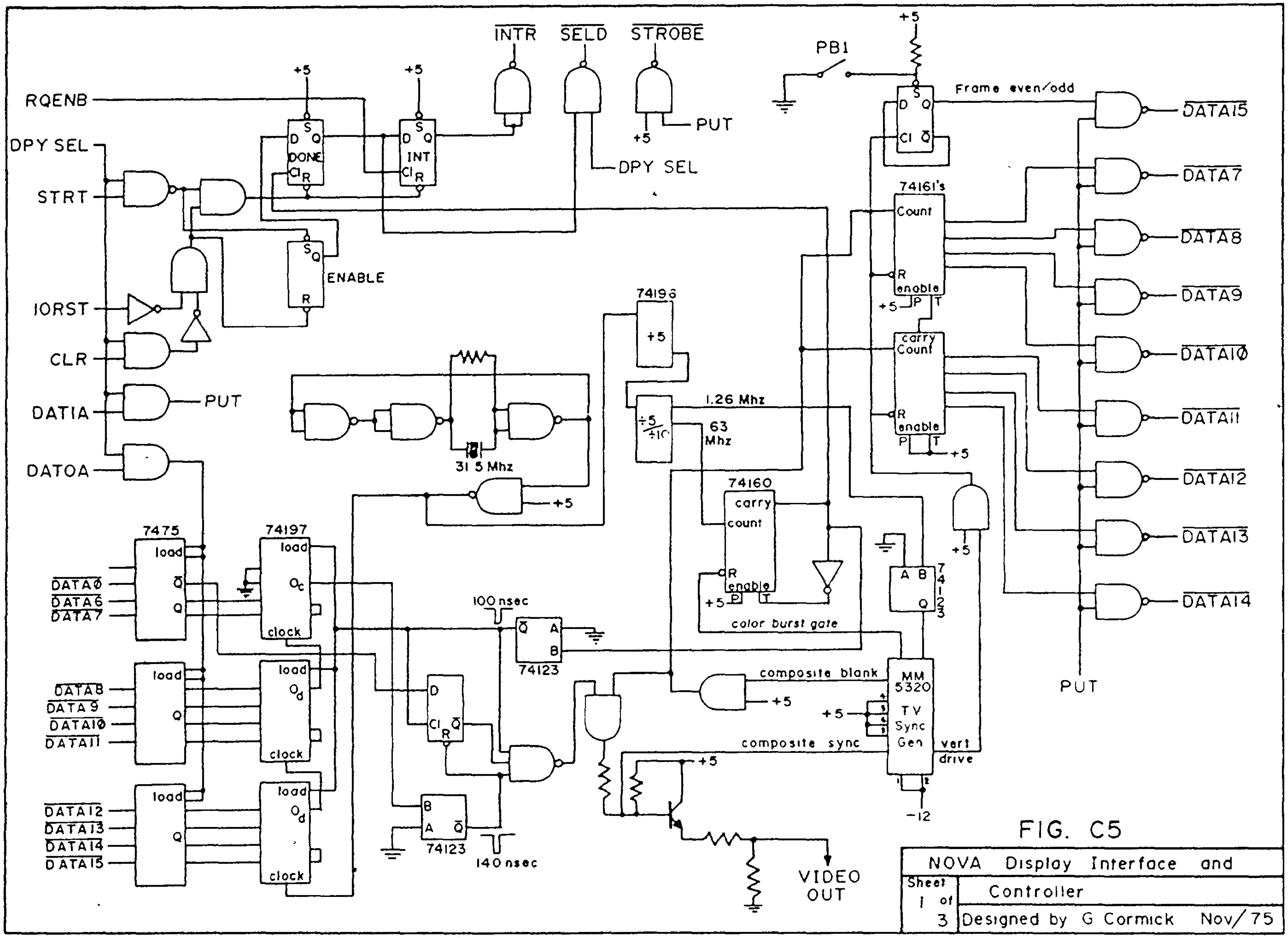

RQENB
DPY SEL
STRT
IORST
CLR
DATIA
DATOA
PUT
+5
DONE
INT
ENABLE
$\overline{\text{INTR}}$
$\overline{\text{SELD}}$
$\overline{\text{STROBE}}$
DPY SEL
PB1
Frame even/odd
74161's
Count
carry Count
enable
74196
÷5 ÷10
1.26 Mhz
63 Mhz
31 5 Mhz
74160
color burst gate
composite blank
composite sync
MM 5320 TV Sync Gen
vert drive
-12
74123
100nsec
140nsec
7475
74197
load
clock
VIDEO OUT
$\overline{\text{DATA0}}$ $\overline{\text{DATA6}}$ $\overline{\text{DATA7}}$
$\overline{\text{DATA8}}$ $\overline{\text{DATA9}}$ $\overline{\text{DATA10}}$ $\overline{\text{DATA11}}$
$\overline{\text{DATA12}}$ $\overline{\text{DATA13}}$ $\overline{\text{DATA14}}$ $\overline{\text{DATA15}}$
$\overline{\text{DATA15}}$ $\overline{\text{DATA7}}$ $\overline{\text{DATA8}}$ $\overline{\text{DATA9}}$ $\overline{\text{DATA10}}$ $\overline{\text{DATA11}}$ $\overline{\text{DATA12}}$ $\overline{\text{DATA13}}$ $\overline{\text{DATA14}}$

FIG. C5

NOVA Display Interface and	
Sheet 1 of 3	Controller
	Designed by G Cormick Nov/75

Bin Connections Viewed from the Back

A	Top	B
Gnd	43	Gnd
	42	$\overline{\text{DATA0}}$
	41	$\overline{\text{DATA1}}$
	40	$\overline{\text{DATA2}}$
	39	$\overline{\text{DATA3}}$
	38	$\overline{\text{DATA4}}$
	37	$\overline{\text{DATA5}}$
	36	$\overline{\text{DATA6}}$
	35	$\overline{\text{DATA7}}$
PB1	34	DATOA
	33	DATOB
	32	DATIA
	31	DATIB
VIDEO OUT	30	STRT
	29	CLR
	28	IOPLS
	27	IORST
	26	$\overline{\text{SELB}}$
	25	$\overline{\text{SELD}}$
	24	$\overline{\text{INTR}}$
	23	PQENB
+5 V.	22	+5 V.
	21	DCMA
	20	$\overline{\text{DCMR}}$
	19	$\overline{\text{DCM10}}$
	18	$\overline{\text{DCM11}}$
	17	DCHI
	16	$\overline{\text{DATA8}}$
	15	$\overline{\text{DATA9}}$
$\overline{\text{DCHP IN}}$	14	$\overline{\text{DATA10}}$
$\overline{\text{DCHP OUT}}$	13	$\overline{\text{DATA11}}$
	12	$\overline{\text{DATA12}}$
$\overline{\text{INTP IN}}$	11	$\overline{\text{DATA13}}$
$\overline{\text{INTP OUT}}$	10	$\overline{\text{DATA14}}$
	9	$\overline{\text{DATA15}}$
	8	DCHO
-12 V.	7	-12 V.
DATIC	6	DEV SEL
DATOC	5	
INTA	4	
OVFLO	3	
$\overline{\text{MSKO}}$	2	$\overline{\text{STROBE}}$
Gnd	1	Gnd

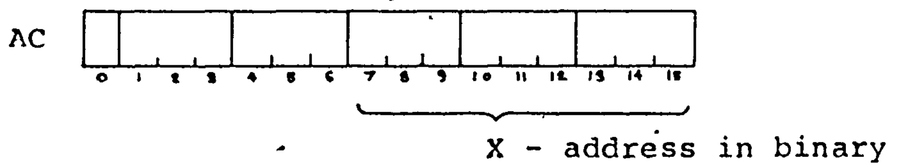
Bottom

FIG. C6

Display Interface	
sheet 2 of 3	Back Panel Connections
	Designed by G Cormick Nov/75

Device Code DPY = 36

NIOS DPY	060136	Enable Interrupt From Display
NIOC DPY	060236	Disable Interrupts
IORST	062677	Disable Interrupts and Reset all Devices
DIA 0,DPY	060436	Read 9 bit X - Address from Display into Accumulator 0 bits 7 - 15



DOA 0,DPY	061 036	Output 10 bit Y - Coordinate from Accumulator 0 bits 6 - 15

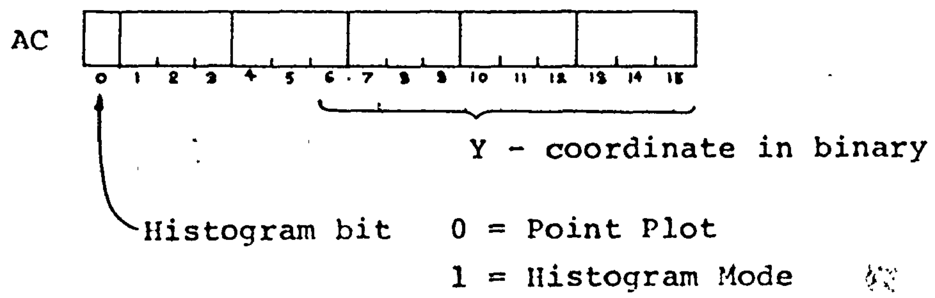


FIG. C7

NOVA Display Interface	
sheet 3 of 3	Instruction Set
	Designed by G. Cormick Nov/75

REFERENCES

1. L. Nicol, A. M. Lopez, A. Robertson, W. V. Prestwich, T. J. Kennett, Nucl. Instr. and Meth. 81,(1970),263.

2. F. Ishaq and T. J. Kennett, Can. J. Phys. 50,(1972), 3090 - 3099.

3. A. Colenbrander and T. J. Kennett, Can. J. Phys. 53, (1975), 236 - 250.

4. V. J. Thomson, W. V. Prestwich, T. J. Kennett, Can. J. Phys. 54,(1976), 383 - 389.

5. L. Nichol and T. J. Kennett, Can. J. Phys. 49,(1971), 1461.

6. A. H. Colenbrander and T. J. Kennett, Nucl. Instr. and Meth. 116,(1974), 237 - 249.

7. A. H. Colenbrander and T. J. Kennett, Nucl. Instr. and Meth. 116,(1974), 251 - 265.

8. B. A. Euler and S. N. Kaplan, IEEE. Trans. Nucl. Sci. NS-17,(1970), 81.

9. A. Robertson, G. C. Cormick, T. J. Kennett, W. V. Prestwich, Nucl. Instr. and Meth. 127,(1975), 373.

10. G, T. Ewan and A. J. Tavendale, Can. J. Phys. 42, (1964), 2286.

11. J. Kantele and P. Suominen, Nucl. Instr. and Meth. 41,(1966), 41.

12. V. J. Orphan and N. C. Rasmussen, Nucl. Instr. and Meth. 48,(1967), 282.

13. U. Tamm, W. Michaelis, P. Coussieu, Nucl. Instr. and Meth. 48,(1967), 301.

14. K. L. Swinth, L. D. Phillip and N. C. Hoitink, IEEE. Trans. on Nucl. Sci. NS-15, No. 3, (1968, 486.

15. H. L. Malm, IEEE. Trans. on Nucl. Sci. NS-13, No. 3, (1966), 285.

16. G. T. Ewan, R. L. Graham, I. K. MacKenzie, IEEE. Trans. on Nucl. Sci. NS-13, No. 3, (1966), 297.

17. M. G. Strauss and R. N. Larsen, Nucl. Instr. and Meth. 56, (1967).

18. G. Panagiotopoulos, M.Sc. Thesis, McMaster Univ. (1969).

19. R. D. Evans, The Atomic Nucleus, McGraw Hill, 1955, chapter 21, p. 616.

20. F. Adams and R. Dams, Applied Gamma Ray Spectrometry, Pergamon Press, Oxford, 1970.

21. P. W. Nicholson, Nuclear Electronics, John Wiley and Sons, New York, 1974.

22. G. W. Hutchinson and G. G. Scarrott, Phil. Mag. 42, (1951), 792.

23. P. W. Byington and C. W. Johnstone, Institute of Radio Engineers Convention Record, 3, (1955), 204.

24. R. W. Schumann and J. P. McMahon, Rev. of Sci. Instr, Vol. 27, No. 9, (1956), 675.

25. R. Spinrad, IEEE. Trans. on Nucl. Sci, NS-11, No. 3, (1964), 324.

26. J. A. Jones, IEEE. Trans. on Nucl. Sci, NS-14, No. 1, (1967), 576.

27. William English, How to Use the NOVA Computers, Data General Corporation, Southboro, Mass. 1974.

28. J. Maynard, Modular Programming, Auerbach Publishers Inc, N. Y. 1972.