AN ABSTRACT OF THE THESIS OF

Darren W. Stevens for the degree of Master of Science in Physics presented on February 25, 1992.

Title : The Design and Architecture of An Improved Microcomputer-Controlled Perturbed Angular Correlation Spectrometer

Redacted for Privacy

Abstract approved : ______________________

John A. Gardner

An improved, integrated perturbed angular correlation (PAC) spectrometer was designed, built and exhaustively tested. PAC spectroscopy is an experimental method used for probing the structure of materials. At the core of the technique is the experimental apparatus which is used in data collection. This work was undertaken in order to improve upon the existing data collection scheme by building an integrated, computer-controlled spectrometer that would be both easy to use and reliable, so that the focus of the user could be dedicated to the physics learned from the data and not committed to the problems associated with the collection of the data. Instead of redesigning the older, existing systems, a revolutionary new design was developed and implemented. An experimental run using room-temperature HfO_2 as the sample and ^{181}Ta as the probe was performed to verify the operation of the system. Analysis of two different data sets from this run showed PAC frequencies that agreed with work done on the same material by other groups. The frequencies for the first data set were:

$$\omega_1 = 829 \pm 2 \ \ Mrad/s$$
$$\omega_2 = 1447 \pm 5 \ \ Mrad/s$$

and for the second data set were:

$$\omega_1 = 827 \pm 3 \ \ Mrad/s$$
$$\omega_2 = 1448 \pm 5 \ \ Mrad/s$$

The agreement between these two data sets and the favorable comparison of both of them to other published results indicates that the spectrometer is fully functional, although some problems associated with purchased components used in the system were discovered. The overall system performed well enough throughout extensive testing to be considered a successful venture and consequently the new system will be the basis for significant future development.

The Design and Architecture of An Improved Microcomputer-Controlled Perturbed Angular Correlation Spectrometer

by

Darren W. Stevens

A Thesis
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed February 25, 1992
Commencement June 1992

APPROVED:

Redacted for Privacy

Professor of Physics in charge of major

Redacted for Privacy

Chairman of the Department of Physics

Redacted for Privacy

Dean of the Graduate School

Date thesis is presented February 25, 1992

Typed by Darren W. Stevens for Darren W. Stevens

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the many people that have helped me on the long road to the completion of this project. First of all, I would like to thank my major professor, Dr. John A. Gardner, for introducing me to this interesting and challenging project and then helping me achieve the completion of such a complicated and detailed endeavor. I would also like to thank Walt Dillon for his many hours of electronics discussion and Tim Taylor for his help in pc-board etching.

I would also like to thank several people in my research group for the help at various points along the way, thanks to: Randy Lundquist for his help in writing the control program for the spectrometer and his writing of the astounding data analysis programs; Matthew Zacate for his extensive hardware help, his friendship, and his many conversations over Thai food; John Griffith for making the hafnium oxide sample that was used to verify the operation of the spectrometer; Eric Antonnissen for lending me the photomultiplier tube from his spectrometer when mine went kaput; Lisa Dundon for providing me with the numbers from old data that I needed for verification.

A couple of my close friends are also deserving of thanks for their friendship and support. Thanks to Neil Roberts for his encouragement at several trying times and to Scott Fuller for his friendship and the many trips to various places when I really needed a break. I would also like to thank the others in the physics department for making it an interesting place to be. I would also like to thank the good folks at Suds and Suds for the refreshing blasts of reality that I was able to squeeze in from time to time.

To my family, without whose support this project would have been an impossible task, I thank you for everything. Special thanks to mom and dad for financial and emotional support and much, much more. To my brother Dean, his wife Robin and their children Jennifer, Heather and Derrick, to my sister Linda and her husband Bob, to my sister Carol, her husband Barry and their children Emily and Brenton, I would like to say thanks for making my occasional trips home so wonderful. You are all very special to me!

Darren W. Stevens, February 25, 1992

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

The Design and Architecture of An Improved Microcomputer-Controlled Perturbed Angular Correlation Spectrometer

1. Introduction

Perturbed angular correlation (PAC) spectroscopy is a useful tool for research in nuclear and solid state physics. Data obtained from PAC studies gives information about the energy levels of the nuclear states of radioactive probe nuclei which can be used to deduce information about its local environment. This information can then be compared to theoretical predictions about the structure of a material, behavior of defects in the material, or various other related entities. It has advantages over other types of study in that it interrogates the local environment of the probe nuclei and thus can be used to study particular properties of materials as chosen by the exact method of study. It has been used by Professor John A. Gardner's research group to study liquid and amorphous selenium-tellurium alloys[1], zirconia ceramics[2,3], oxide ceramics[4], high T_c superconductors [5,6], defects in ceria[7], and various other materials. There have been compilations done of results obtained from PAC studies done on compounds by Lerf and Butz[8], on metals by Vianden[9], and a general list of applications by Rinneberg[10].

1.1. Theory of perturbed angular correlations

Radiations from an atomic or nuclear system are not random in their direction, but have a pattern dependent on the observed system[11]. This means that when two successive gamma rays are emitted from a single excited nucleus in the process of decaying to the ground state, there is a preferred direction for the emission of the second with respect to the emission of the first (a pattern exists). This result is what allows the whole technique of PAC to be useful; By observing these patterns, the environment of the nuclei responsible for the emissions can be deduced, and then this information can be used to understand the physical structure and other properties of the studied materials. If the decay process is repeated by many nuclei that share the same (or very nearly the same) environment and these patterns are recorded, then these observations can be used to determine the properties of the system. An example of a system that meets the similar environment criteria would be nuclei occupying the same lattice positions in a well-ordered crystalline material. Analysis of the data would reveal properties of the environment at these lattice positions and the structure of the solid could then be characterized.

The underlying theory behind the use of PAC is quite complex. In order to fully understand how the interactions within the system produce the results that are observed, one must understand electricity and magnetism, quantum mechanics, solid-state, and nuclear physics. A complete theoretical treatment of the formalism for the static electric field gradient (EFG) is given by Frauenfelder and Steffen[12]. A summary of that treatment and a complete development of the PAC theory has been done by Gaskill[13], Jaeger[14], Su[15], Weidlich[16], Fuchs[17], Schwenker[18], and Wang[19]. Only a brief theoretical development will be presented here, since this work was not undertaken as an exercise in using PAC spectroscopy, it was done to improve the experimental system. The form and notation of Jaeger[20] and Su[21] will be adopted in this discussion and their work will be summarized.

Consider the intermediate state of the nuclei that undergo the decay cascade mentioned previously. During its finite lifetime τ, the nuclei will interact with any external or internal electromagnetic fields that act upon them. The interaction with these fields will cause the nuclei to precess and the angular distribution of the second gamma ray with respect to the first will be changed from the case where there are no fields acting. By measuring the angular correlation for the system in which the fields are acting and comparing it to the expected angular correlation with no fields acting, information about the interaction can be obtained. By learning about the interaction, the environment of the nuclei can be determined and the properties of the sample can be characterized. The only interactions that must be taken into account for the most popular

PAC probes is the interaction of the electric quadrupole moment of the nucleus with the electric field gradient. The reason for this is that (i) the most common PAC probes have closed electronic shells (spherical symmetry) and therefore no electric or magnetic dipole moments, (ii) the materials that are studied are usually non-magnetic, and (iii) higher order electronic terms are negligible. It will be assumed that this is the case in the rest of this discussion.

In PAC spectroscopy, the observation of the first gamma ray emission from the nuclear decay (through cascade $I_i \rightarrow I \rightarrow I_f$), selects a subset of all of the decaying nuclei in the sample. This must be the case, because only certain spin orientations can produce gamma ray emissions in the direction of a detector. As a result of this, the second gamma ray from these nuclei must be emitted in an angular distribution that is correlated with the direction of the first. The probability that a nucleus decays by the emission of two gamma rays, γ_1 and γ_2, in the directions $\mathbf{k}_1$ and $\mathbf{k}_2$ into solid angles $d\Omega_1$ and $d\Omega_2$ is defined as:

$$W(\mathbf{k}_1, \mathbf{k}_2, t)\, d\Omega_1 d\Omega_2$$

By using quantum mechanics to describe the transition from $I_i \rightarrow I \rightarrow I_f$ it can be shown that:

$$W(\mathbf{k}_1, \mathbf{k}_2) = \sum_{k=0,\, even}^{k_{max}} A_{kk}\, P_k(\cos\theta)$$

is the angular correlation function for free nuclei. In this equation, A_{kk} are numbers that depend only on the spins of the nuclear states and $P_k(\cos\theta)$ are the Legendre polynomials where θ is the angle between the gamma ray emissions.

If a static extranuclear perturbation field is taken into account, after similar analysis it can be shown that:

$$W(\mathbf{k}_1, \mathbf{k}_2) = \sum_{k=0,\, even}^{k_{max}} A_{kk}\, G_{kk}\, P_k(\cos\theta)$$

is the resulting perturbed angular correlation function. The assumption made in the derivation of this equation is that the sample is made up of many crystals that are randomly oriented. The resulting perturbation function G_{kk} can then be understood as the average of the perturbation function for a single crystal. This perturbation function can be expressed as:

$$G_{kk}(t) = \sum_{n} S_{nn}^{kk} + \sum_{n \neq n'} S^{kk} \cos\left[\frac{(E_n - E_{n'})\,t}{\hbar}\right]$$

where the coefficients S^{kk} depend on the nuclear states in the transition. The energies E_n and $E_{n'}$ are the energy eigenvalues for the interaction Hamiltonian. It is important to note the time independent term $\sum_{n} S_{nn}^{kk}$. Due to the existence of this term, the angular correlation of a polycrystalline sample is never completely destroyed by the static perturbing fields. For this reason, the term has historically been called the "hard-core".

In order to fully understand the perturbation function, the energy eigenvalues for the interaction Hamiltonian must be found. Solving for the energy eigenvalues of the interaction Hamiltonian is quite complicated and only the results will be shown here. As was stated earlier, the only interaction that must be taken into account is the interaction of the electric quadrupole moment of the nucleus with the electric field gradient (EFG). To simplify things, the principle axis system is chosen for the EFG in such a way that $|V_{xx}| \leq |V_{yy}| \leq |V_{zz}|$ $(V \equiv \text{EFG})$. For external charges the Poisson equation reduces to the Laplace equation $V_{xx} + V_{yy} + V_{zz} = 0$ so that η , which is defined by:

$$\eta = \frac{V_{xx} - V_{yy}}{V_{zz}}$$

is restricted to $0 \leq \eta \leq 1$. In the above definition, η is called the asymmetry parameter.

For static, axially-symmetric EFG's the quadrupole Hamiltonian is diagonal and results in energy eigenvalues:

$$E_m = \hbar\omega_Q[3m^2 - I(I+1)]$$

where the quadrupole frequency ω_Q is defined as:

$$\omega_Q = \frac{eQV_{zz}}{4I(2I - 1)\hbar}$$

where Q is the the quadrupole moment of the nucleus. For the case where the electric field gradient is not axially symmetric, the Hamiltonian is not diagonal. In this situation, the Hamiltonian must be diagonalized in order to solve for the eigenvalues, a process which can be quite difficult. If the system is restricted to I = 5/2 , which is the case for the PAC probe used in this work, then the diagonalization results in :

$$E_{\pm\frac{5}{2}} = 2\alpha\hbar\omega_Q\cos[\frac{1}{3}\cos^{-1}\beta]$$

$$E_{\pm\frac{3}{2}} = -2\alpha\hbar\omega_Q\cos[\frac{1}{3}(\pi + \cos^{-1}\beta]$$

$$E_{\pm\frac{1}{2}} = -2\alpha\hbar\omega_Q\cos[\frac{1}{3}(\pi - \cos^{-1}\beta]$$

where

$$\alpha = \sqrt{\frac{28}{3}(\eta^2 + 3)} \qquad \text{and} \qquad \beta = \frac{80(1 - \eta^2)}{\alpha^3}$$

Recall that: $(I = \frac{5}{2} \rightarrow m = \pm\frac{5}{2}, \pm\frac{3}{2}, \pm\frac{1}{2})$

By substituting these into the equation for G_{kk}, the expression can be now be directly linked to the energy states of the nucleus and the EFG at the probe nuclei. This is the fundamental basis for PAC; by extracting the G_{kk} function for the experimental data, information about the environment of the probe is also obtained.

Before closing this discussion, there are a few conclusions that should be drawn. One observation is that the $\pm m$ energy levels for both cases (symmetric and asymmetric EFGs) are doubly degenerate. This results in (2I+1)/2 distinct energy levels for half-integer I, and to I+1 distinct energy levels for integer I. For the case I=5/2, there are three distinct frequencies (depending on m - shown on the previous page) observed when the data is analyzed, corresponding to the differences between these energy levels. These three frequencies are referred to as the PAC frequencies. For the asymmetric EFG case, the energy levels (and therefore the PAC frequencies) depend on the degree of the asymmetry (η). The last conclusion that should be mentioned is that the three PAC frequencies must obey the sum rule, $\omega_1 = \omega_2 + \omega_3$.

It should also be noted that the possibility exists that the electromagnetic fields in the local environment of the probe may change as a function of time. If the fields change during the lifetime of the intermediate state, then the electric quadrupole interaction becomes time-dependent and therefore very complicated. By studying this, there is quite a bit to be learned about such things as the jump rate of defects, correlation times, and the trapping energy. The system can (and usually does) include both static and time-dependent perturbing fields. The derivations for time-dependent perturbations are beyond the scope of this discussion and will not be presented here.

1.2. Experimental perturbed angular correlation spectroscopy

PAC spectroscopy is an experimental method that allows for detailed study of the structural properties of various materials. As its name implies, it involves establishing the angular correlation between the emissions of two (or more) gamma rays from an excited nucleus as it decays to the ground state. Obviously, the first requirement in studying a material involves the presence of radioactive probe nuclei in the material. If a probe species is not naturally present or cannot be manufactured by a process such as irradiation, than it can be artificially introduced by one of many different methods. Once the probe nuclei have been introduced into the sample, the sequence for a decay of interest would be for the parent nucleus to decay by some mechanism, usually electron capture or beta decay, into a daughter nucleus which is left in an excited state. This nucleus then decays to the ground state by the emission of a single gamma ray or a cascade of gamma rays. It is the latter emission scheme that is useful for this experimental method.

It is necessary to carefully pick the probe nuclei so that certain criteria are fulfilled for this type of work. The first, and most important, is that the gamma rays are emitted in a cascade in such a way that the intermediate state of the nucleus exhibits a half-life that is suitable for the resolution of the time measuring device. The gamma rays must also have energies that are within the range of common detectors. There are several radioactive probe nuclei that satisfy these criteria (see appendix A). The next restrictions on the choice of probe depend directly on the physical characteristics of the material to be studied and the specific properties of the material that are of interest. For example, if the material to be studied is a crystalline structure, the charge on the probe atom and the physical size of the atom must also be compatible with the lattice site in which the probe is to be introduced. This is very important if the results are to be correctly interpreted, since analysis of the angular correlation gives information about the local environment in which the probe nuclei are placed. By knowing the site (or sites) where the probe nuclei are located, predictions and simulations of the results can be made, thereby confirming or disproving theoretical models about the material.

The data collection for this experimental procedure can be performed in many different ways. The first essential pieces of equipment are the detectors, which are normally photomultiplier (pm) tubes (see figure 1.1). The basic operation of a pm tube is very simple. A gamma ray is incident upon the scintillation crystal and excites the crystal lattice. The lattice then produces visible light in the process of relaxation. These

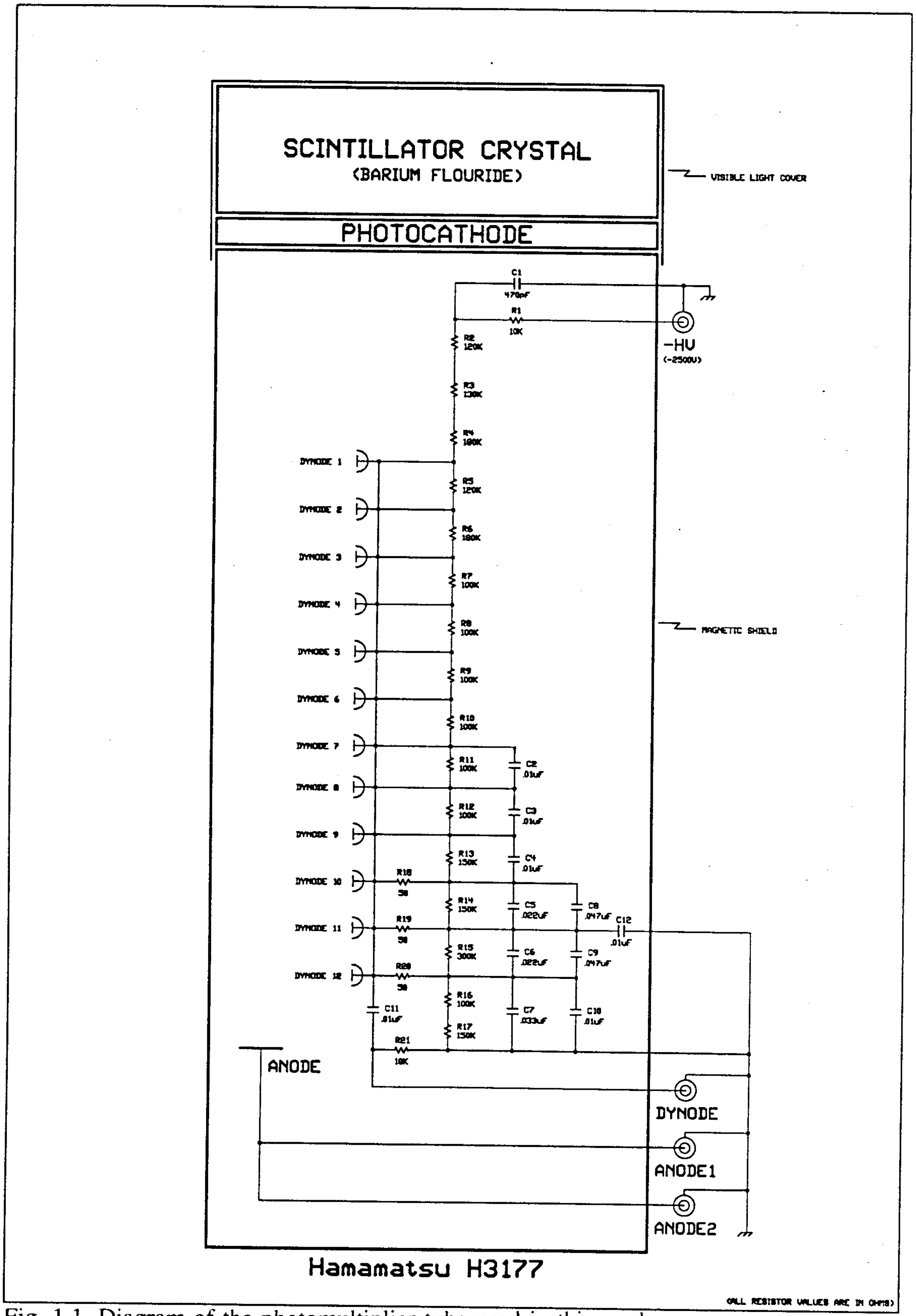


Fig. 1.1. Diagram of the photomultiplier tube used in this work

photons strike the photocathode which causes it to emit electrons. These electrons are accelerated towards a chain of dynodes, each of which is held at a different electric potential. The first electrons (from the photocathode) hit the first dynode, which causes it to emit secondary electrons. This process is repeated throughout the length of the dynode chain, with each link multiplying the number of electrons that are emitted. At the end of the chain there is anode which "collects" the electrons. This burst of electrons produces a voltage pulse across a resistor, the amplitude of which is proportional to the number of electrons and therefore to the energy of the incident photon. There is also a secondary, slower output of the pm tube which comes directly from one of the last dynodes in the chain. It is different in that it is shaped by an R-C network to have a form that is better representative of the energy of the incident photon and is easier to use in conventional amplifiers. The primary, fast pulse (called the anode pulse) is usually used for timing purposes and the slower, secondary pulse (called the dynode pulse) is usually used for energy analysis.

The essential sections of the system after the detectors are a section which analyzes the energy of the emitted gamma rays and a high resolution timer that records the time between the reception of these emissions. The capture of an "event" of interest can be simply stated as a sequence that includes the detection of the first gamma ray of the cascade which starts a timer and the detection of the second gamma ray of the cascade which stops the timer. It is for this reason that the first gamma ray of the cascade is called the start gamma ray and the second gamma ray of the cascade is called the stop gamma ray. By recording thousands of these events, it is possible to establish the angular correlation of the emission of these gamma rays. With these parts of the system in place, the last necessary part of the hardware of the system is the section which collects the data and stores it in a usable form. There is also the need for a data analysis system to process the data after it has been gathered. For flexibility and efficiency reasons, a personal computer was chosen to perform those tasks in the new system instead of the conventional multi-channel analyzer.

This work was undertaken to improve upon existing data collection systems. The goal was to design and build an integrated system that was easy to use, efficient, and could gather a more complete set of data than was currently being accomplished. Care was taken at every step to insure that the design was flexible enough to make future hardware modifications unnecessary by allowing the software to dictate what data is gathered from the system. By making the information available to the computer through software, there is an inherent flexibility in the types of experiments that can be performed with this new system.

For the record, it should be noted that there are actually two different strategies that can be employed in order to gather the data necessary for analysis (see figure 1.2). The first strategy is to use a pair (or pairs) of detectors, keeping one fixed and moving the second detector on the circumference of a circle with the source at the center of the circle (labelled MOVABLE DETECTORS in figure 1.2). By moving the second detector to a sufficient number of different locations and collecting data for a known period of time at each position, the angular correlation between gamma ray emissions can be established. The second method (called time differential perturbed angular correlation or TDPAC) involves placing two pairs of detectors at fixed positions (usually 90° and 180° apart) on the circumference of a circle with the source at the center (labelled FIXED DETECTORS in figure 1.2). It is this experimental set-up that was used for this work and all discussions from this point on will refer to this scheme (references to this method from this point on will be shortened to PAC).

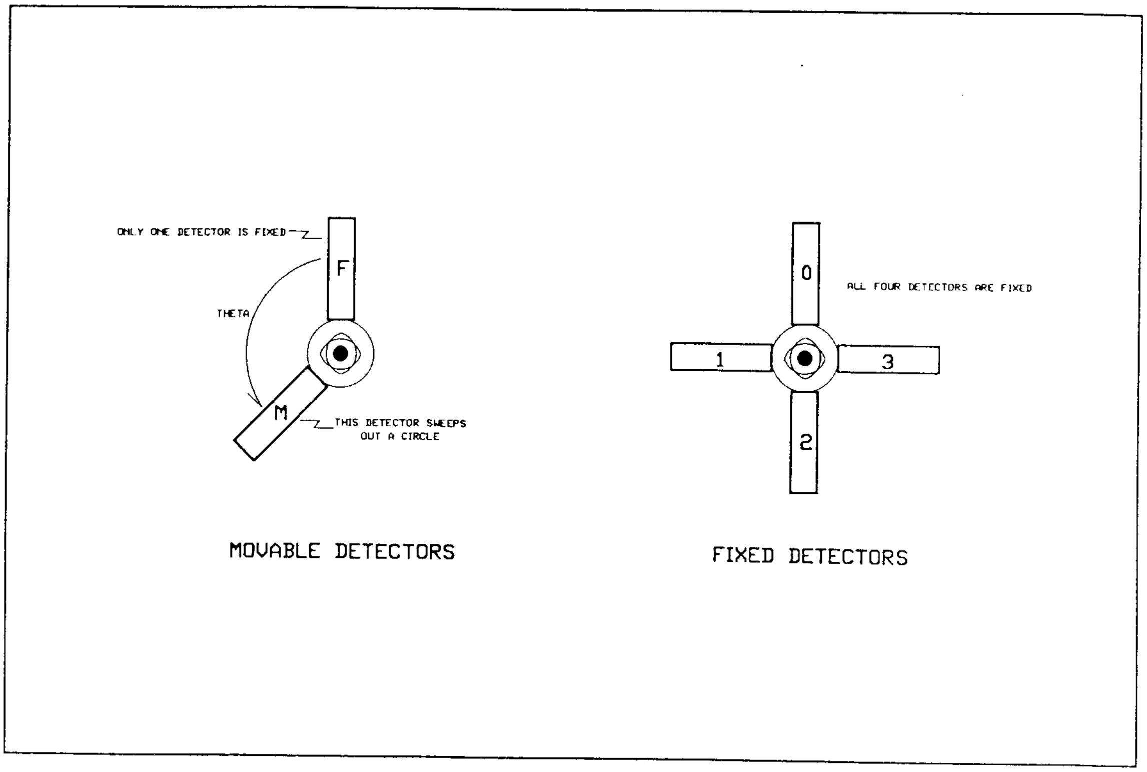

ONLY ONE DETECTOR IS FIXED
F
THETA
M
THIS DETECTOR SWEEPS OUT A CIRCLE
MOVABLE DETECTORS
0
ALL FOUR DETECTORS ARE FIXED
1
3
2
FIXED DETECTORS

Fig. 1.2. Diagram of the different detector configurations used in PAC

2. The design of the old spectrometer

In preparation for a detailed treatment of the new spectrometer design, it is helpful to first examine the architecture of the system from which it has evolved. The "old" spectrometer which is referred to here is, itself, an evolution from an earlier system[22]. This earlier system has been modified and improved upon to the point that inclusion of a discussion on it would not prove beneficial and therefore will be excluded. The evolution that has taken place in the past has been to improve upon the existing design, with the central "core" of the system left relatively unchanged. The new system has been significantly changed and should be considered a new generation. Throughout this discussion, figure 2.1. should be consulted. The discussion will be limited to what is relevant to understanding the overall hardware functions and will not include supporting hardware such as the furnace or furnace controller. Detailed discussion of the calibration and other necessary procedures is mentioned, but the purpose here is not to explain the full operation of the old system.

2.1. Overview

The old spectrometer can best be discussed by tracing the different sequences of events that would occur when the system is running. Consider the fixed detector system in figure 1.2. The four detectors (labelled 0, 1, 2, and 3) are placed on the circumference of a circle with a radius of approximately 5 cm, and the sample is placed in the center of the circle. The radioactive nuclei that are present in the sample are constantly decaying and emitting their gamma rays in all directions. On occasion, a gamma ray will be incident on one of the four detectors and the detector will react to this occurrence (in the manner described in section 1.2).

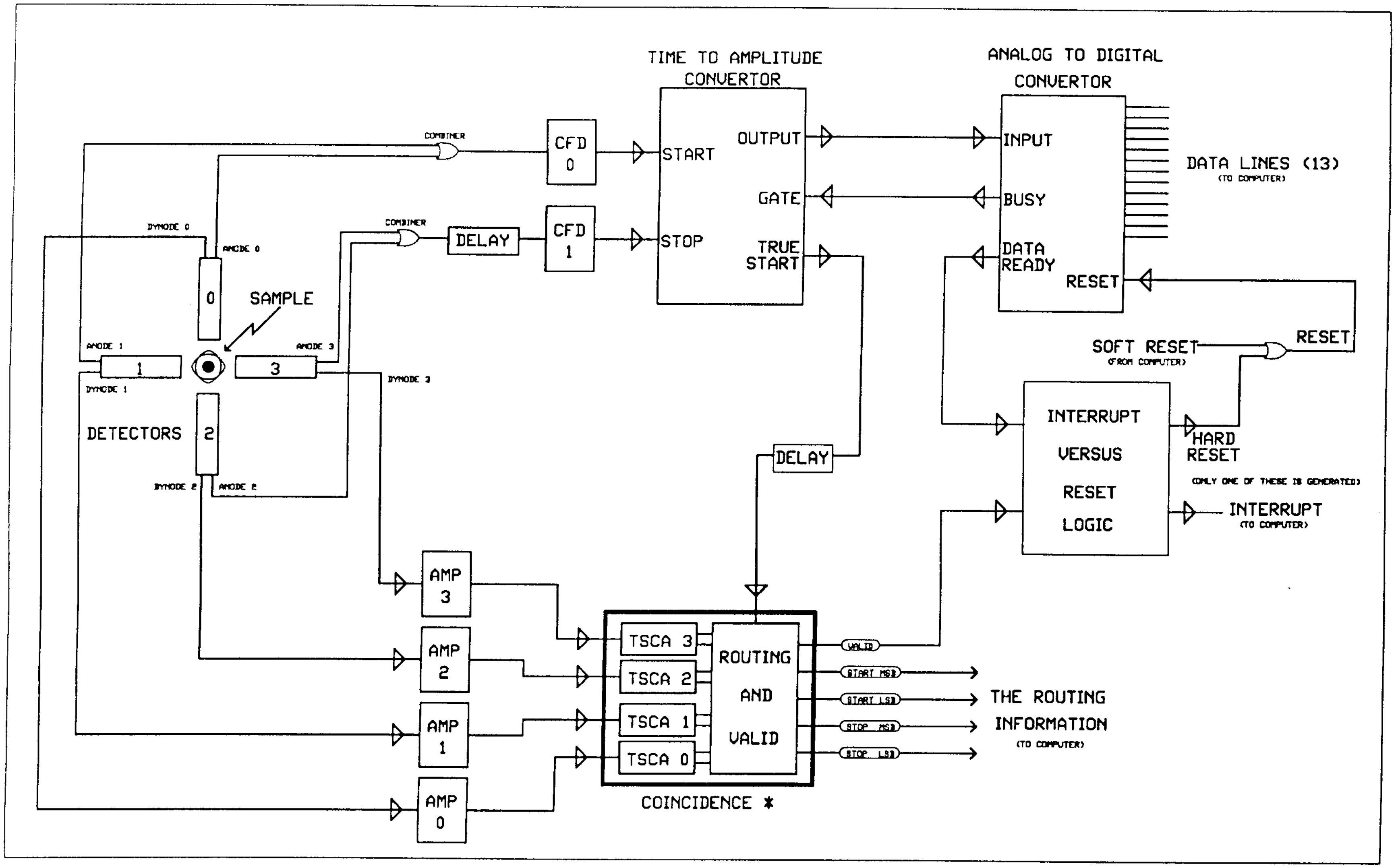


Fig. 2.1. Block diagram of the old spectrometer

This occurrence starts the ball rolling. The detector that received the gamma ray will output an anode pulse which travels down a length of coax cable to a constant fraction discriminator (CFD). The function of the CFD is to discriminate electrical noise from the photon signal in the pm-tube and to produce a timing pulse which alerts the rest of the system that a gamma ray has been detected. The noise level threshold is adjustable by means of a potentiometer accessible on the front panel of the CFD. The CFD internally splits the signal from the anode and delays one branch. The delayed portion of the signal is subtracted from a fraction of the undelayed signal. The result is a bipolar signal which has a baseline crossover that is independent of the amplitude of the input signal. The bottom line is that the CFD outputs a signal at a time that is independent of signal amplitude and is therefore independent of the energy of the incident photon. Therefore, the CFD is exactly what is needed for the timing section of the spectrometer.

An important feature of the old spectrometer is that it was designed with only two CFD's. The output of detectors 0 and 1 are first fed into a radio frequency combiner which performs an "or" function on their signals. The output of the combiner is then routed into the start of a Time-to-Amplitude Convertor (TAC). In a similar manner, the output of detectors 2 and 3 are sent into another combiner. The output of this combiner is sent through a delay module that delays the signal for an amount of time equal to about half of the full-scale of the TAC. The TAC functions in a manner similar in principle to a stopwatch, in that it measures the time between the reception of a start signal and the reception of a stop signal. The difference is that it outputs a voltage level that is proportional to the difference in time, not a conventional representation of the time.

The output of the TAC is sent into an Analog-to-Digital Convertor (ADC), which converts the analog representation of the time into a corresponding digital (binary) value. The delay placed before the CFD that generates the stop of the TAC serves a very important role; It makes the TAC output a value corresponding to half of its maximum for a simultaneous (or as close to simultaneous as electronically possible) event. Thus an event that has a "stop" pulse arriving at the stop combiner **before** the "start" pulse will be recorded as a positive time, provided the time between them is less than half of full scale on the TAC. The end result is that the TAC now records two different time scales, one that is forward in time and one that is backward in time. In other words, since mid-range on the TAC corresponds to a zero time difference between CFD signals, a value between mid-range and the TAC maximum corresponds to positive time and a value between mid-range and the TAC minimum corresponds to negative time. This feature is crucial to the function of the overall system because of the way that the energy branch of the system is designed.

The energy branch starts from the dynode signal of the pm-tubes. Each detector's dynode output signal is connected to a linear amplifier which simultaneously shapes and amplifies the signal. The positive, unipolar output of each amplifier is connected to a laboratory-built Twin Single Channel Analyzer (TSCA). The schematic diagram for these is shown in figure 2.2. The basic function of them is to check to see if the amplitude of the signal from the amplifier is within two different "windows". These two windows are set by manually adjusting potentiometers which adjust a lower level threshold (LLT in the figure) and an upper level threshold (ULT in the figure). Since the amplitude of the voltage pulse from the amplifier depends on the energy of the incident photon, these windows can be set so that one of them gives a "true" output for pulses that correspond to an energy that signifies the detection of a start gamma ray and the other one gives a positive output for pulses that correspond to an energy that signifies the detection of a stop gamma ray. The windows have a finite width that is determined experimentally, since the behavior of all of the associated electronics tends to spread out the range of voltages that correspond to a photon of a specific incident energy. Since there are two "channels" that are possible, the name TSCA is used for this section. One channel is tuned to the start gamma ray and the other to a stop gamma ray.

It should be noted that this design has some inherent problems. One is that the thresholds tend to drift with changes in temperature and with fluctuations in the power supply. Another is that the timing that governs the window-checking employs a peak-sensing operational-amplifier (op-amp) circuit, which is pulse-height sensitive by nature of its design. This means that the timing will be different for a start or stop photon if they are of significantly different energy. The fact that it works at all can be explained by the notion that the energy resolution is not a critical factor in the overall system. If

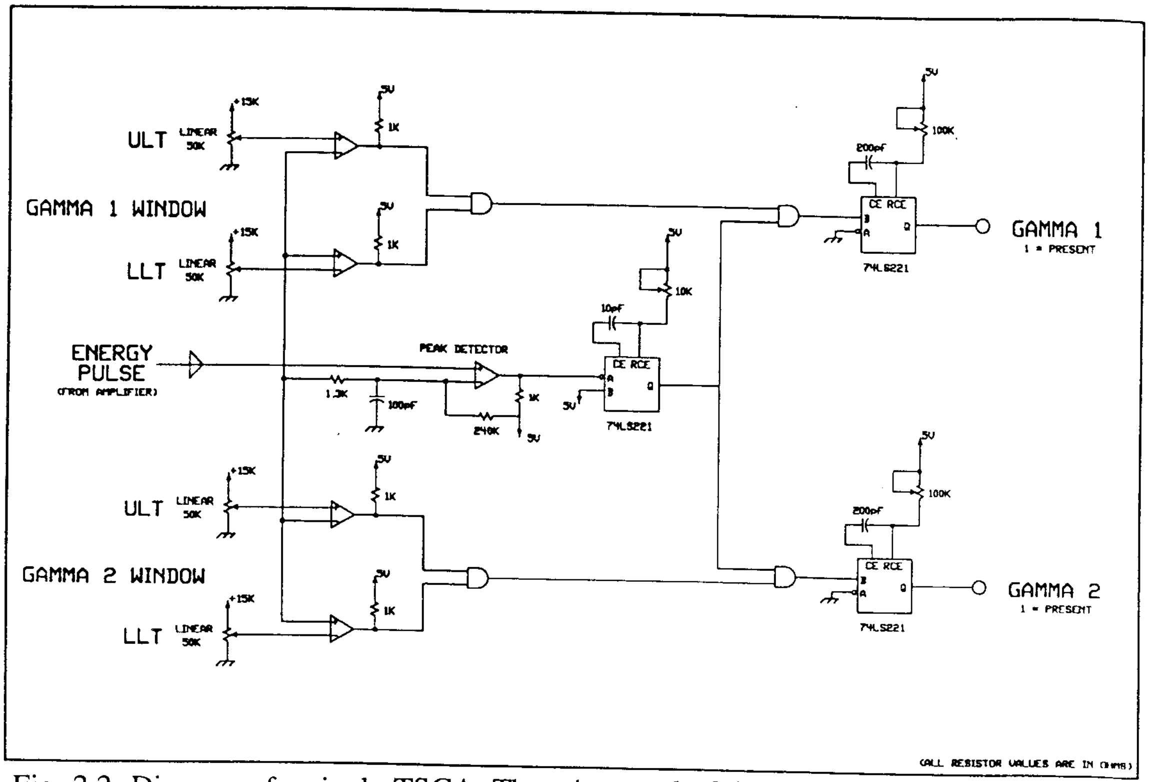


Fig. 2.2. Diagram of a single TSCA. There is a total of 4 of these, one for each detector

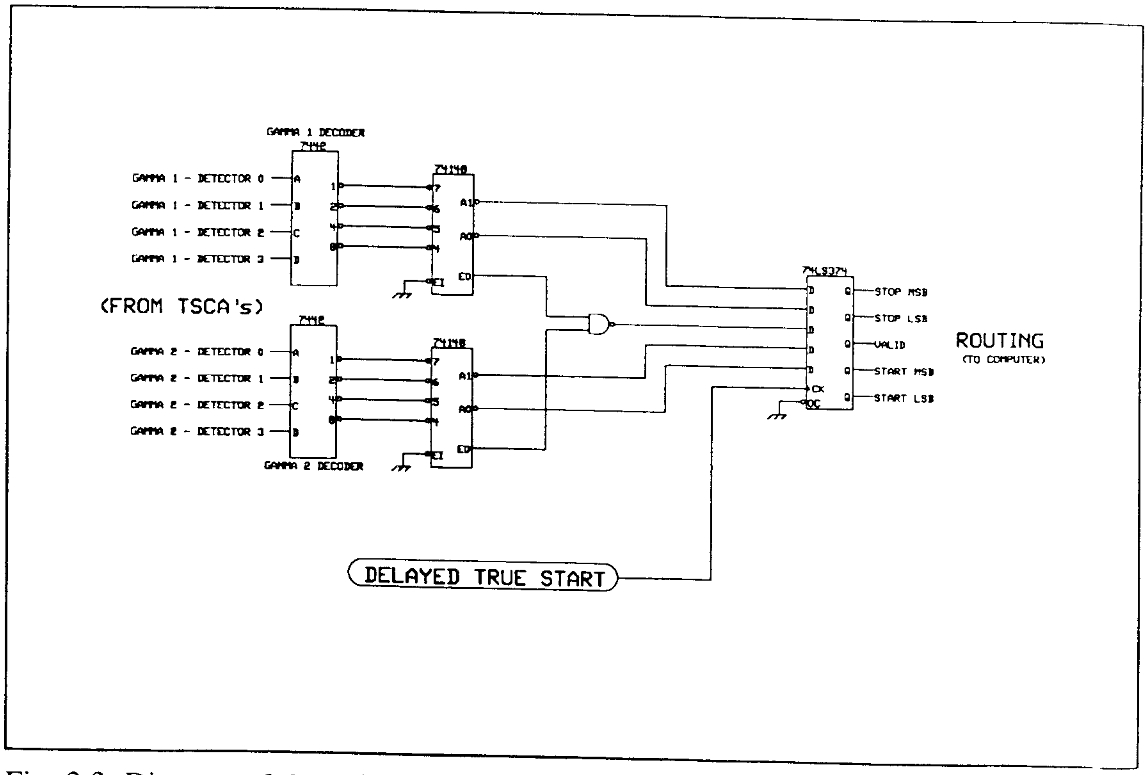


Fig. 2.3. Diagram of the coincidence section of the old spectrometer

the windows are set too wide, then there is an decrease in the signal-to-noise ratio in the data because of the increase in stray (or background) counts. If the windows are set too narrow, then there is a increase in the dead time of the system due to the tight restrictions placed on the acceptable range of energies. If the windows are set somewhere in the middle of this range, than neither of these problems contributes significantly to the overall performance of the system. However, setting windows still remains as one of the biggest problems with the old design and is one area that was addressed in the new design.

In continuing with the design discussion, it is assumed that the windows are set in such a manner that there is a reasonable probability that if a start or stop gamma ray is incident upon a detector than the corresponding TSCA will provide the correct output. The next part of the system is the coincidence section (see figure 2.3). The output of the four TSCA outputs are the inputs to this portion of the system. This section is responsible for providing two crucial pieces of information about the photons that were detected. The first provided information is about which detectors received photons of energies corresponding to a start or stop (the routing information) and the second is whether or not it corresponds to a valid event. A valid "normal" event is one in which detector 0 or 1 absorbed a start gamma ray and detector 2 or 3 detected a stop gamma ray. A valid reverse event is one in which detector 2 or 3 detected a start gamma ray and detector 0 or 1 received a stop gamma ray. The case where more than one detector received either a start or a stop is ambiguous, and is therefore considered invalid. The result of the energy analysis is clocked into a latch by the delayed TRUE START signal from the TAC. This signal only occurs if the TAC received a start and stop signal within its full-scale range. If the TAC receives a start with no stop, than it resets itself and a new sequence is processed. It will now be assumed that the TAC received a start and stop and the current sequence is continued.

At this point in the cycle there are two different cases to be considered. In either case, the timing of the system is such that spectrometer will now be in a state where the event has been flagged as valid or invalid and the system waits for the ADC to assert the DATA READY signal, which lets the rest of the system know that it is done digitizing the signal from the TAC. The VALID line, the DATA READY line, and a line that distinguishes between run or calibrate mode (an output from the computer called RUN/CAL) are now inputs to the interrupt versus reset logic. At the time the DATA READY line is asserted, the decision is made whether or not to interrupt the computer. If the VALID line is not asserted, then the computer will be interrupted only if the RUN/CAL input has the system in the calibrate mode. If the system is in run mode, than a hardware reset (HARD RESET) is asserted and the system is reset. If the system is in the calibrate mode than the computer is interrupted every time the DATA READY line is asserted and calibration data can be obtained from the system by the computer. In the case where the VALID line is asserted, the computer is interrupted at the time the DATA READY line is asserted, regardless of the state of the RUN/CAL line.

Upon interrupt, the computer stops whatever it is doing (after completing the instruction it is executing), saves the state of the machine and executes an interrupt service routine. The interrupt service routine reads the routing information, reads the output of the ADC, and figures out where in memory it will increment a value that corresponds to the count of the occurrences of the event. The computer memory is divided into segments that correspond to the different possible events (see table 2.1). The offset within the segment corresponds to the digitized value of the time between the detection of events. This scheme allows a table look-up method to be implemented in the interrupt service routine and time spent in the service routine to be minimized. This is an important consideration, since this time contributes directly to the dead-time of the system. After the proper memory location has been incremented, the computer issues a software reset (SOFT RESET) signal and the system is reset.

The data is collected in this manner for a long period of time, usually a few days to ensure good statistics. The computer has on-line data analysis capabilities, so that data can be checked from time to time and problems can be detected if they occur. An important point to note is that the interrupt service routine runs in the background and that the computer can be used to do various other tasks while it is collecting data. After a run is completed, the data is transferred by floppy disk to more powerful computers for analysis.

2.2. Data collection and analysis

The data sets that are collected in the old system are limited by the design of the spectrometer. Since the start of the TAC results from an "or" of the signals from CFD 0 and CFD 1 and the stop of the TAC results from an "or" of the signals from CFD 2 and CFD 3, there are four coincidences that are impossible to obtain any information about (see table 2.1). The events that result from a coincidence between the pairs of detectors in the "or" of the start or stop do occasionally occur, but because of the design of the system, they only start (or stop) the TAC twice. This means that these events will do nothing but contribute to the dead time of the system. However, the overall effect is not that significant, because the system processing time is already dominated by events in which a start is received, but there is no corresponding stop (called singles). The fact remains that there is a portion of data that exists but cannot be collected by the spectrometer.

At the end of an experimental run, the data that is collected can be analyzed in a straightforward manner. For each valid detector pair, the data consists of an array in which the index is the time between the reception of a start and stop and the contents in each location is the number of occurrences of that event. If we define D_{ij} to be the raw data for the detector pair i and j, then this raw data can be expressed as:

$$D_{ij} = \frac{1}{\tau_N} e^{-\frac{t}{\tau_N}} e_i \; e_j \; N_0 \; W(\theta, t) \; + B_{ij}$$

and then the background corrected count rate can be expressed as:

$$C_{ij} = D_{ij} - B_{ij} = \frac{1}{\tau_N} e^{-\frac{t}{\tau_N}} e_i \; e_j \; N_0 \; W(\theta, t)$$

in this notation[23] θ is the separation angle between the ith and jth detector and e_i, e_j are the detectors efficiencies. N_0 is the decay rate of the parent isotope and τ_N is the average half-life of the intermediate state of the daughter isotope. W(θ,t) is the angular correlation function which contains all of the information about the probe-environment interaction. B_{ij} is the background rate for the ijth data sector.

Recalling the angular correlation function for polycrystalline samples with static interactions from the first chapter, we can write W(θ,t) as:

$$W(\theta, t) = \sum_{k=0,\, even}^{k_{max}} A_{kk}\, G_{kk}(t)\, P_k(\theta)$$

For most of the isotopes used in this research group, I=5/2, which restricts the value of k in such a way that $k_{max} = 4$ (due to selection rules). Using this, the approximation that $A_{44} \approx 0$, and adopting the notation $A_{22} \equiv A_2$, then the angular correlation can be written:

$$W(\theta, t) \approx 1 + A_2\, G_2(t)\, P_2(\cos\theta)$$

If the spectra-ratio for the normal spectrum is written[24] as:

$$R_n(t) = 2\frac{(C_{02}C_{13})^{\frac{1}{2}} - (C_{03}C_{12})^{\frac{1}{2}}}{(C_{02}C_{13})^{\frac{1}{2}} + 2(C_{03}C_{12})^{\frac{1}{2}}}$$

and for the reverse is written as:

$$R_n(t) = 2\frac{(C_{20}C_{31})^{\frac{1}{2}} - (C_{30}C_{21})^{\frac{1}{2}}}{(C_{20}C_{31})^{\frac{1}{2}} + 2(C_{30}C_{21})^{\frac{1}{2}}}$$

then the spectra-ratio is approximately equal to A_2G_2. Therefore, by very simple calculations, the interesting physics can be extracted from the raw data.

Table 2.1. The raw data sectors for the old system

SECTOR NUMBER	SECTOR LABEL	START GAMMA	STOP GAMMA	COMMENTS
0	01	0	1	INVALID
1	02	0	2	VALID,FORWARD
2	03	0	3	VALID,FORWARD
3	10	1	0	INVALID
4	12	1	2	VALID,FORWARD
5	13	1	3	VALID,FORWARD
6	20	2	0	VALID,REVERSE
7	21	2	1	VALID,REVERSE
8	23	2	3	INVALID
9	30	3	0	VALID,REVERSE
10	31	3	1	VALID,REVERSE
11	32	3	2	INVALID

Table 2.2. The different A_2G_2 sets for the old system and how they are formed

A_2G_2 SET	90° SECTORS	180° SECTORS	FORMULA (* SECTOR FORM)
FORWARD	03,12	02,13	$\frac{(02*13)^{1/2} - (03*12)^{1/2}}{(02*13)^{1/2} + 2*(03*12)^{1/2}}$
REVERSE	30,21	20,31	$\frac{(21*30)^{1/2} - (20*31)^{1/2}}{(21*30)^{1/2} + 2*(20*31)^{1/2}}$

* In sector form the sector label represents the number of background-corrected counts in the sector. There is also an implicit function of time in this equation. In other words:

$$ij = C_{ij}(t) = D_{ij}(t) - B_{ij}(t)$$

in the above table.

3. The architecture of the new spectrometer

Before discussing the internal architecture of the new spectrometer, it would be beneficial to consider an overview of the components. The natural starting place for a discussion on the new spectrometer is at the detectors. The fixed detector method (shown in figure 1.2) is employed in the new system, with the positions of the four detectors being perpendicular to each other. There are two standard "NIM bins" that hold and power devices such as the CFD's and amplifiers. These NIM bins are mounted in a standard 19" rack, which also houses a "VME" card cage and a power supply for the VME card cage. There is also a personal computer (IBM-XT) connected to the rack-mounted components which controls the system.

The VME card cage is usually used for the building of a custom-made computer system, but it is not used for that purpose in the new spectrometer. It was chosen because a standard backplane (or motherboard) was available that provided 14 different mounting positions for "cards", each of which had 96 connections to a common bus. This design allowed custom electronics to be built and plugged into the backplane with the interconnections used for intercommunication. The backplane was also designed in such a way that allowed for easy connection of an external power supply to the power buses which were provided at each card slot. The only restriction placed on the custom electronics was that the boards that the circuits were assembled on had to use the type of connector that the VME standard required. As it turned out, this requirement was a problem in the manufacturing process used for the custom printed boards, because the connectors were of triple row design and the boards could only be manufactured with two layers. This problem was resolved by carefully choosing the placement for the traces on the printed circuit boards. Now that the outline of the system has been described, the internal workings can be described.

3.1. Overview

Although the new spectrometer functions as one integrated unit, it can best be discussed by separating the system into two different sections (see figures 3.1 and 3.2). The natural division between the energy and timing channels will be used in this discussion, since there is only minor overlap between these two segments. The reality that they are both tied together should not be forgotten, however. Without simultaneous processing of events in both of these channels, experimental data could never be recorded. It is also worthwhile to remember that the starting point for the processing of an event in both channels is ultimately at the detectors. From there, the timing signals governing the processing of a given event are tied together until they take separate routes from the CFD's. For this reason, each detector and CFD set can be thought of as a single entity for the discussion of the timing system.

Instead of the two CFD's used in the old system, there is one CFD for each of the four detectors in the new setup. This method allows a more accurate tuning of the CFD threshold to the individual detectors and also provides for the distinction between all timing events. The timing of event processing for a given detector splits up at the three CFD outputs, each of which conforms to the NIM standard. Two of the three CFD outputs are dedicated to recording the time between the reception of gamma rays and the other one is used for the starting the energy analysis channel. It should be noticed that all of the timing signals (both the timing and the energy sections of the system) are generated from the output of the CFDs.

It seems natural to dedicate the first part of this overview to the timing channel, since the control of the overall event processing is governed by this channel. It is assumed that the reader will be consulting the block diagram of the timing section shown in figure 3.1 in conjunction with this discussion. Consider the system from the state in which every component has been reset and is ready to process an event. The first part of the sequence is established following the detection of a gamma ray is when the pm-tube (Hamamatsu H3177) sends out an anode (timing) pulse, followed shortly thereafter

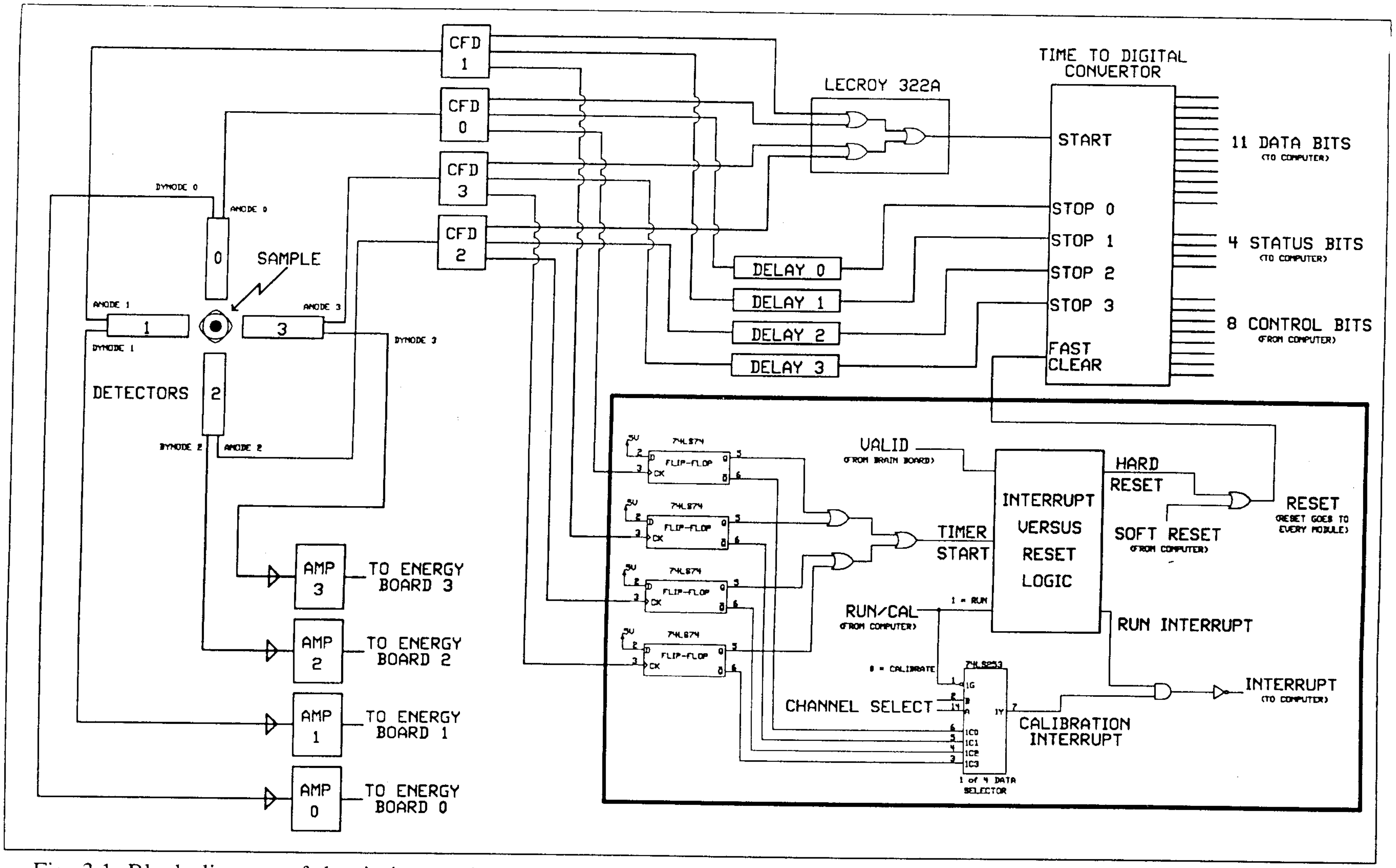


Fig. 3.1. Block diagram of the timing section of the new spectrometer

by a dynode (energy) pulse. The anode pulse travels along a coaxial cable to the CFD (ORTEC model 934-S), where it is level-discriminated. Assuming that the anode pulse resulted from the detection of a photon of interest (above CFD threshold), then the CFD asserts the three output lines. One of these is connected to a Lecroy (model 322A) logic unit that is configured in the "or" mode. This signal propagates through the logic unit and starts the Time-to-Digital Convertor (TDC - Lecroy model 2228A) by asserting the common start input of the TDC. The second output of the CFD traverses a long length of cable that serves as a delay (approximately 80ns). This time delay has been experimentally tuned to stop the channel of the TDC corresponding to the detector channel that started it at a TDC count of about 20. This event is called a self-stop and provides a zero-time reference.

In order to understand this more clearly, a discussion of the TDC is necessary. The TDC is a device that performs the same function as the TAC and ADC from the old system. It is different in that there are eight different channels that keep track of time relative to a common start. Of these eight channels, the four that had the closest calibrations were chosen to be used for this spectrometer. The TDC uses a common start signal to trigger the charging of capacitors in all eight channels (in unison). The charging of the capacitor in each one of these channels can be individually stopped and the resulting total charge on that particular capacitor will then be proportional to the time that elapsed from the reception of the start until it was stopped. If no stop signal for a channel comes within the maximum time allowed, then the unstopped channel will have the maximum possible charge. At the end of this preset maximum time (referred to as full-scale) the TDC starts a 20MHz oscillator and begins discharging the capacitors. Internal counters (each channel) count the number of clock cycles that pass while the capacitors are discharging. In this way, the time that elapsed between reception of start and stop pulses for each individual channel is recorded by the counters. This method, called the Wilkinson run-down method[25], can be made very accurate by choosing high quality circuit components, and therefore a very precise measure of time can be achieved. It should be noted that the time scale resolution of the TDC can be adjusted by an external switch, making it possible to use the device for radioactive probe isotopes with different halflives. For this work, the TDC had a resolution of approximately 0.45ns per channel.

In parallel with the two signals that are associated with the TDC, there is a signal which travels to a circuit board called the STINTSET board which is housed in the VME cage (this board will be discussed in detail in the next section). On that board, a timer is started which keeps a limit on how much time is spent processing each "event". For continuity, assume another gamma ray arrives at one of the other detectors in a time that would allow it to be recorded as a valid event. The second detector sends out an anode pulse which travels to its corresponding CFD. Just like the first CFD sends out three signals, this one does the same. Since the TDC has already been started, the second start signal from the "or" gate is ignored. The delayed stop signal is not ignored, however, and the channel that received this second gamma ray is stopped at a count of about 20, plus the count of whatever amount of time elapsed since the start was received. Using this scheme, the count difference between the second channel that was stopped and the self-stopped channel will be an accurate measure of the time between the detection of the two photons in the detectors (all channels are tuned to self-stop at a count of about 20). At this point, the TDC has recorded an accurate measure of the time between the detection of the two gamma rays and it is time to check the result from the energy analysis to see if a cascade of interest has been timed.

The energy analysis has been taking place in parallel to the timing sequence that was just discussed. The energy analysis section (see figure 3.2) of the new spectrometer is probably the most innovative section of this spectrometer (see the following sections for a more complete discussion of the individual boards). The overall idea is based on the recent technological innovations in the area of flash-type, integrated-circuit Analog-to-Digital Convertors (flash ADCs). The new generation of these components allows analog information to be quickly transformed into its digital representation, which can be processed using fast and accurate digital electronic circuits. The old spectrometer used analog circuitry (the TSCA's) to decide if the voltage pulse from the amplifier was in the correct range to correspond to a gamma ray of interest. The new system uses a fast ADC (Analog Devices AD7821KN - 650ns conversion) to obtain an accurate digital representation of the same information which can than be processed digitally. The binary

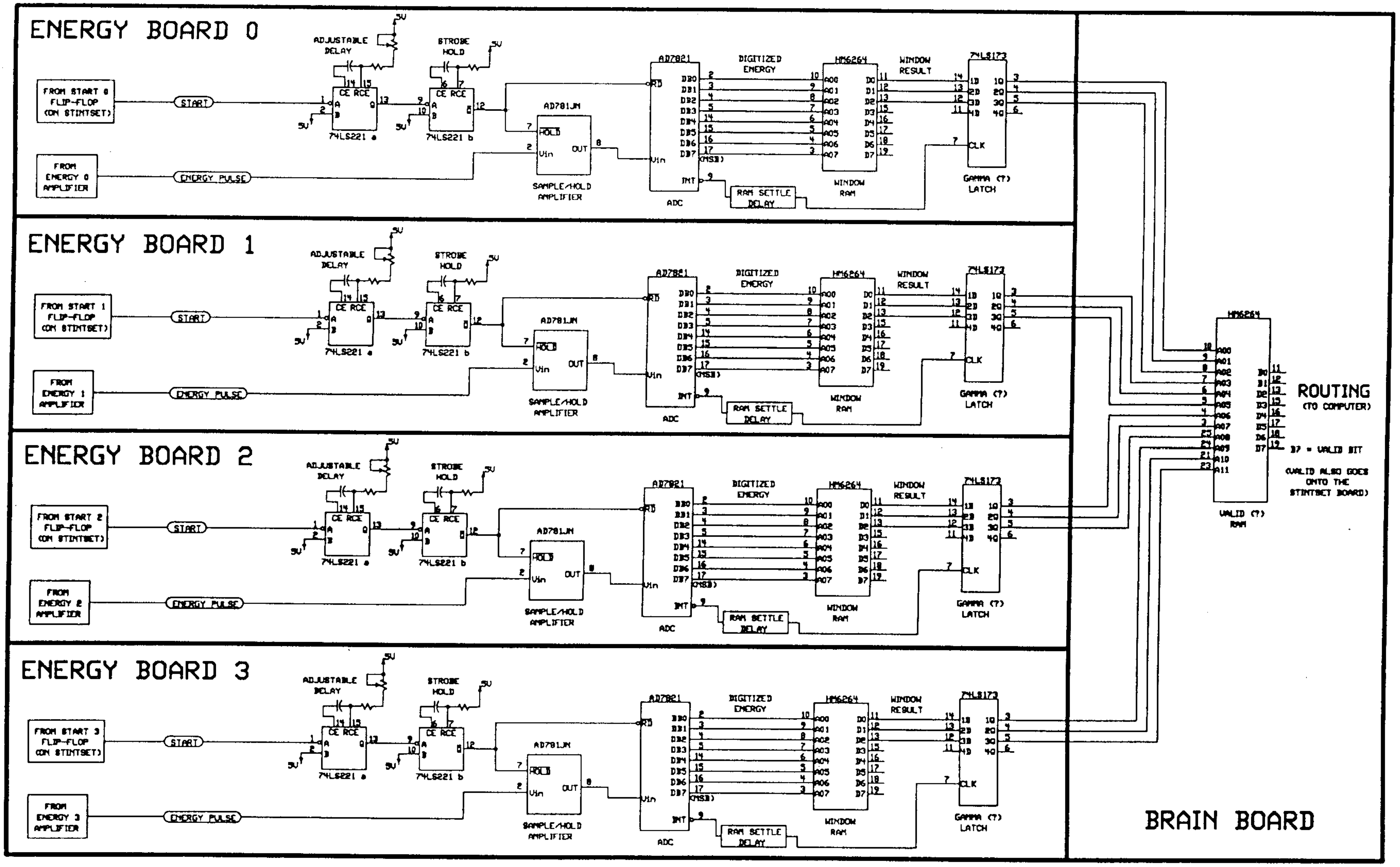


Fig. 3.2. Block diagram of the energy analysis section of the new spectrometer

output of the ADC is used to address a Random Access Memory (RAM) chip (Hitachi HM6264), which has been previously loaded with a "table" of values. In this way, the RAM (referred to as a "WINDOW" RAM) can be used as a hardware look-up table to see if the digital representation of the energy corresponds to a gamma ray of interest. One of these systems (an Energy Board) is attached to the output of each amplifier, which is attached to the dynode signal of each detector.

Further analysis is performed by a printed circuit board in the VME cage called the Brain board. The output of all four of the individual WINDOW RAM outputs are latched and the output of these latches (74LS173) are used as the input to another RAM (HM6264). This RAM is also used as a look-up table (referred to as the "VALID" ram). This RAM has the job of deciphering whether or not the combination of gamma rays that have been detected corresponds to the detection of a gamma ray cascade of interest (a VALID event). This RAM also encodes the combination of individual detector events into the same type of information that was encoded by the coincidence section of the old spectrometer. This information is referred to the "ROUTING" INFORMATION and it identifies which of the detectors received the start and stop gamma rays. For example, if detector 0 detected a start gamma ray and detector 1 detected a stop gamma ray, then this would correspond to a valid 90° event. The valid bit would be asserted and the routing would describe the sources that produced it. In this way, the VALID RAM gives direct information as to whether or not an event is of interest and if it is, what the details of the event were.

The timer on the STINTSET board is preset for an amount of time equal to the full-scale of the TDC (currently $\approx$ 1μs), plus the time of propagation through the energy analysis section ($\approx$ 3μs). The reason for this should be fairly obvious. If the second gamma ray comes in at a time just less than full-scale, then it is valid as far as the timing is concerned, and the final selection criteria must come from the VALID RAM. At the end of the preset time the system looks at the VALID line and decides on one of two choices. If the VALID bit is not asserted, then the event is not interesting, and a hardware reset (HARD RESET) is generated. The total amount of time that the system takes to process an invalid event is approximately 9μs. If the VALID bit is asserted, then the computer is interrupted and executes an interrupt service routine. This service routine is responsible for gathering the timing data after interrogation of the routing information. Just as in the old system, a memory location corresponding to the count of the number of occurrences of the particular type of event is incremented. It then issues a software reset (SOFT RESET) and the system is reset for the processing of another event. The total amount of processing for a valid event is approximately 250μs, due mainly to the processing time of the interrupt service routine.

One section of the new system that was not described above is the CALIBRATION INTERRUPT section of the STINTSET board. The implementation will be left to the next section which describes the STINTSET board in detail. The use of it is very specialized and it is only active while the system is being prepared for a data collection run. The basic need that it fulfills is to provide the computer (and therefore the operator) with the calibration information needed to load the value into the WINDOW RAM. Basically, a known energy gamma ray will give a defined range of ADC outputs, but how is the computer and operator to know what that range is? The answer comes from the use of the CALIBRATION INTERRUPT circuitry. The user sets the system into special mode that interrogates each ADC output individually and thus provides the information necessary for the loading of the WINDOW RAM.

An important detail of the system that should be addressed at this point is the method of overall control within the VME cage. There are six different cards connected to the VME backplane that all have to be **simultaneously** controlled if there is to be any hope of passing information back and forth between them. Imagine a group of people that are all talking at once; No person can hope to hear what another person across the room might be trying to tell them! The scheme that was devised and implemented to solve this problem is one of the most ingenious parts of the new system. On each one of the Energy boards and on the Brain board, there are two Erasable-Programmable Read-Only Memory (EPROM) chips. These integrated circuits are used to store the possible "modes" that each one of the boards needs to be in for a given state of the overall system. After correlating the contents correctly (see Appendix B), the computer can be used to address these EPROMs and all parts of the VME cage can be configured to the correct mode. It should be noted that the STINTSET board does not have any EPROMs on it because it does not have any connections that might conflict with the other boards. It does, however, have connections to the VME FUNCTION bus (the EPROM addresses) and thus can be configured to the CALIBRATION mode (mentioned above).

3.2. The STINTSET board

The name STINTSET board (see figure 3.3) was derived from STart-INTerrupt-reSET which describes the functions that the circuitry on the board performs. Its inputs are the four START signals from the CFDs, the VALID signal from the Brain board, the SOFT RESET line (from the computer), and the three most significant FUNCTION bits (EPROM address bits) which act as the mode control for the board. The outputs include four gate signals for the CFDs, two reset lines that are sent to the Brain and Energy boards via the VME backplane, and the INTERRUPT line (to the computer). Also recall that it has the timer associated with the length of time each event is processed.

The STINTSET board is best viewed as the overall timing control for the whole system. It receives the start signals from the CFDs, which are immediately sent into flip-flops on this board. It should be mentioned that the CFD outputs had to be modified to provide the necessary TTL (transistor-transistor logic) signals (see figure 3.4) for these flip-flops. The positive outputs from each of these flip-flops are "ORed" together resulting in a single signal that is asserted if any one of the detectors receives a photon. This signal is then used to start a timer "one-shot" (the timer was mentioned previously; a one shot can be considered as a preset timer throughout this discussion) that is preset for full-scale of the TDC plus the propagation time through the energy analysis section. At that preset point in time, the valid bit is "looked at" to see if the sequence that started the timer resulted in a valid event. If the event was valid, then a pulse from a second one-shot is steered (by the valid bit) to trigger a flip-flop and the computer is interrupted.

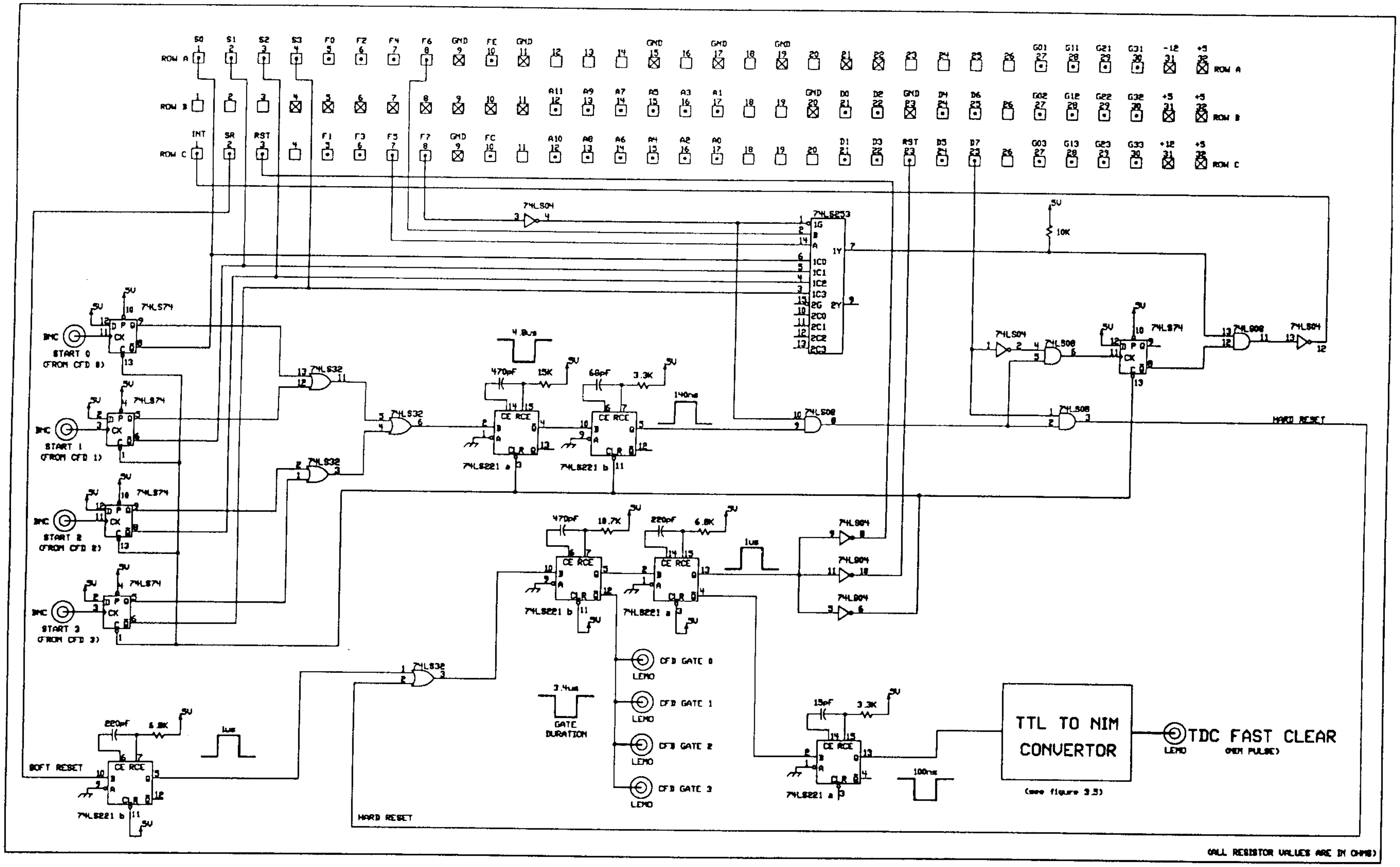


Fig. 3.3. Schematic diagram of the STINTSET board

If the event is not valid, then the pulse is steered (HARD RESET) to a one-shot that controls the resetting of the system. It should be noted that this HARD RESET signal is ORed together with the output of a one-shot that is connected to the computer (SOFT RESET). In the event that the computer was interrupted, after executing an interrupt service routine, the computer will trigger this other one-shot and a reset will be generated.

The one-shot connected to the computer SOFT RESET line was implemented due o the slowness of the computer. It was found that the interrupt service routine holds the SOFT RESET line in the high state for a seven microsecond time period. This caused a problem if an invalid event followed the interrupt, since the computer would hold the input to the OR gate in the high state. The positive edge from the HARD RESET was not able to propagate through the OR gate and the result was that the edge did not reset the system, resulting in a locking-up of the system. By placing the one-shot with a one microsecond pulse width in the path of the computer signal, the problem was avoided.

The reset sequence is something that should also be discussed, since it is not very straightforward. The output of the reset OR gate is a positive edge that triggers the first one-shot. This one-shot pulse width is set to a time of 3.4us, which is connected to the CFD gate inputs and gates them off during this time period. It also triggers a second one-shot which has a time-constant of about 1us, which is the reset signal that is sent onto the VME bus. This second one-shot is always contained within the pulse width of the gate one-shot, thereby insuring that the TTL components in the VME cage will be reset while the CFD is gated off. At the end of the second one-shot pulse (1us), there is a third one-shot triggered which is connected to a TTL-to-NIM convertor (see figure 3.5). The TTL-to-NIM convertor is necessary because the TDC will only accept NIM level signals as its FAST-CLEAR input.

There is one reason that the reset sequence is so complicated in this system; the separation of the timing and energy analysis sections at such an early point in the overall sequence (at the CFD). Recall that in the old system, the delayed TRUE-START of the TAC was use to strobe the result of the energy analysis. In the new design, this signal is absent, which means that the energy and timing sections are no longer tied together in the processing of an event. Therefore, it is necessary to **guarantee** that the entire system

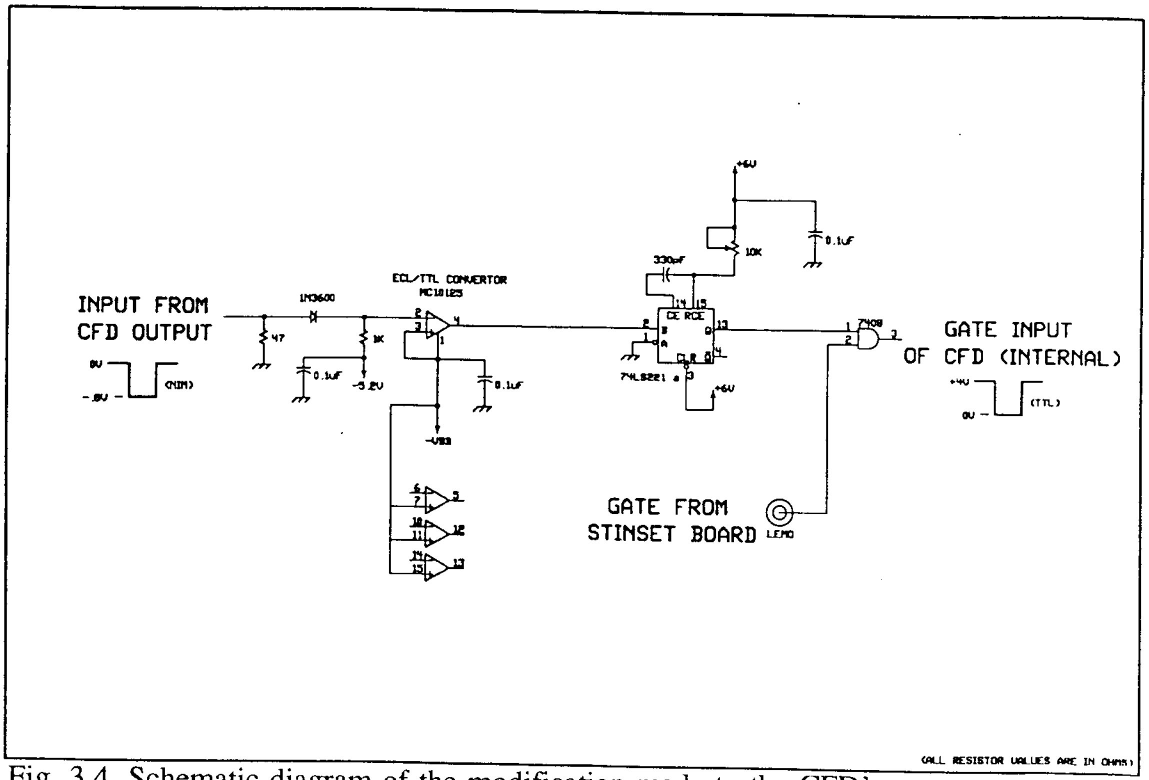


Fig. 3.4. Schematic diagram of the modification made to the CFD's

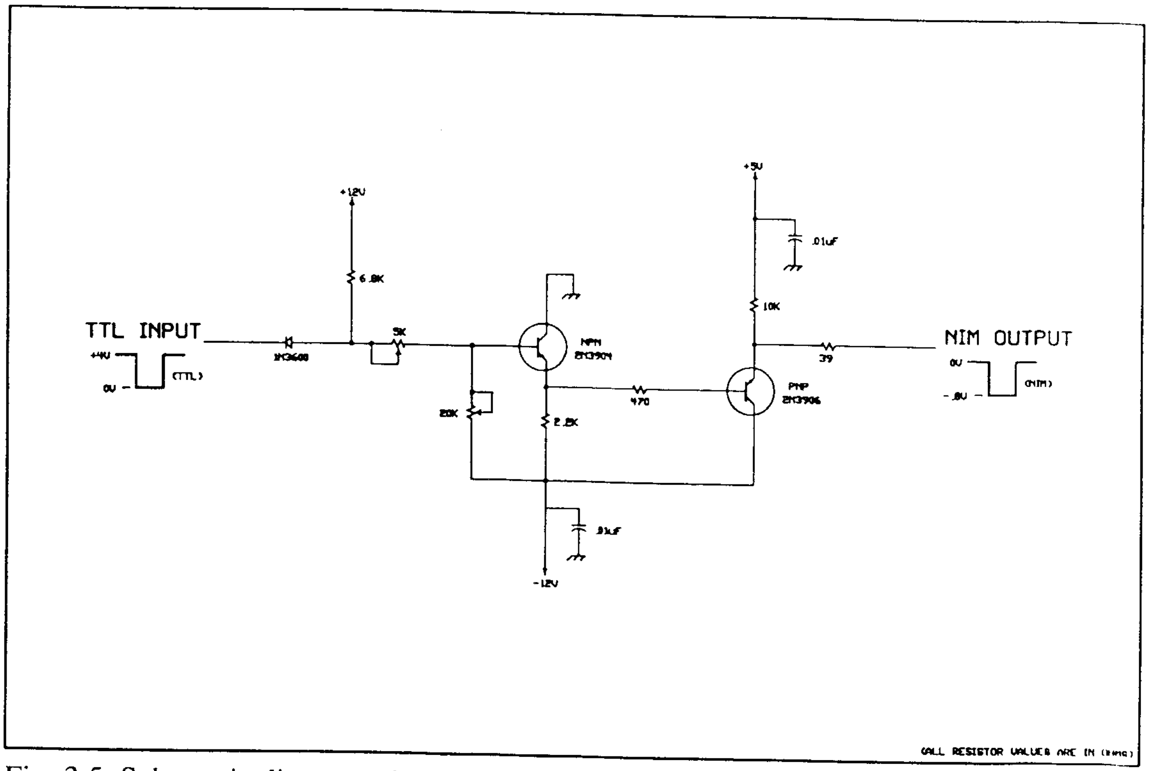


Fig. 3.5. Schematic diagram of the TTL-to-NIM conversion circuit for the TDC fast clear

is reset and ready to go at **exactly** the same time. If the timing section becomes ready before the energy section or vice-versa, then the possibility exists that the two sections will process signals from different photons. In order to prevent this from occurring, theCFDs had to be gated off while everything was reset, and then the CFD gate could be turned off. This was further complicated when it was learned that there was a 700ns delay from the time the CFD gate was asserted until the CFD output was actually gated off. The knowledge that the TDC required a 2us time delay from the assertion of the FAST CLEAR until it was ready to process an event also complicated matters.

The other section of this board that is pertinent to this discussion is the CALIBRATION INTERRUPT section. The computer FUNCTION output to the VME bus is used to control this section. In the calibration mode, the regular interrupt section is disabled and the 1-of-4 decoder is enabled by the most significant bit of the FUNCTION. The computer sets the address to whatever channel the user wants to calibrate via the next two most significant bits of the FUNCTION, and the output of the ADC for the selected channel is available to the computer. In this way, the user can determine what ADC outputs correspond to which gamma rays and can set the WINDOW RAM accordingly. The disabling of the interrupt-delay timer used in regular data gathering mode means that every single incident photon causes the computer to be interrupted, so the count rate is astronomical in this mode. The reader should refer to section 4 for a further discussion of the use of this mode for window setting.

3.3. The Energy boards

The Energy board(s) also obtained their name by the function that they perform in the spectrometer; they are responsible for the energy analysis for each of the detectors. Care was taken in the design of these boards to make sure that all four of the boards could be built as identical units and jumper selected to be a given channel. This feature allowed the boards to be tested and debugged by using the same procedures. During the construction of the first prototype, some flaws in the design were noted and were taken into account before the construction of the final four boards. Several testing methods established while verifying the operation of the first board were carefully chronicled so that the final four boards could be tested in the shortest period of time. Most importantly, computer code written for the basic testing of the single Energy board prototype could be used for the testing the rest of them. Refer to figure 3.6 for a diagram of these boards.

There are only two inputs to the energy boards that should be considered as external, the energy signal from the amplifier and the start signal from the CFDs. The start signal actually comes through the VME bus from the flip-flop on the STINTSET board, a feature which limits the start signal to occurring only once during a processing cycle (the flip-flops on the STINTSET board are set up that way). This prevents the board from losing the information that it is processing due to the occurrence of another start signal. The basic function of the energy boards is the WINDOWing. It is on these boards that the digitization and analysis of the energy pulses is performed. It will be assumed that the WINDOW RAM contains the correct contents in the following discussion.

The first part of the Energy board sequence is the reception of the start pulse (jumper selected) from the STINTSET board. This pulse triggers a one-shot that is adjustable to the rise time of the amplifier pulse. The amplifier output is connected to a sample-and-hold amplifier (SHA), which samples the amplifier pulse at its peak (it is strobed by the variable one-shot). It should be noted that the SHA was necessary because the rise-time of the amplifier pulse is too fast for the ADC to accurately track the pulse. The output of the SHA is the input to the ADC. Notice that the SHA and the ADC are strobed simultaneously (by the same signal). This is possible because the SHA has a propagation delay of only 250ns, whereas the ADC has a 350ns set-up time. The output of the ADC is used to address the WINDOW RAM as was previously mentioned. There is also a signal (INT) from the ADC which is asserted when the ADC is done digitizing the analog signal. That signal is used to trigger another one-shot (labelled LATCH CLOCK in the diagram). The LATCH CLOCK one-shot has a preset pulse width of

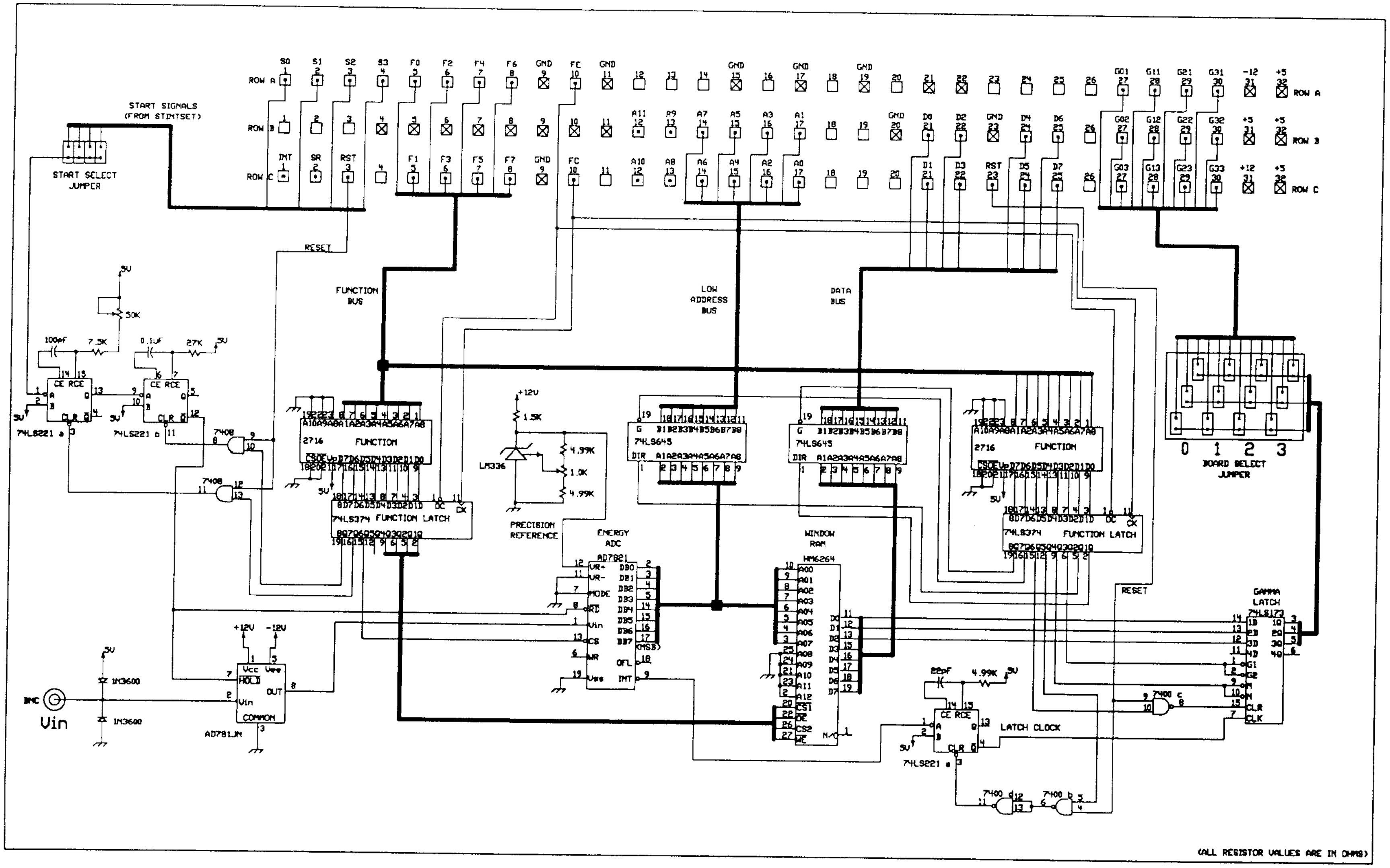


Fig. 3.6. Schematic diagram of the Energy boards

140ns that is just long enough to allow the ADC-addressed result of the RAM to settle. The positive edge of the one-shot is used to trigger the latch which holds the result from the RAM. The three output lines of the latch are then sent (jumper selected) onto the VME bus and is used on the brain board.

It should be noted that there is also the communication link between the VME bus and the WINDOW RAM on this board. The EPROMs that were mentioned previously are employed to make sure there is no bus contention between the RAMs on the other boards and the one that is being loaded (or read). This provision allows the computer to individually to establish the contents of the WINDOW RAM on each Energy board during calibration mode. For further discussion of this functional aspect of the board, refer to chapter 4.

3.4. The Brain board

The Brain board (see figure 3.7) has two different functions in the overall system. Both of these are cerebral in nature, so the name of this board also resulted from its overall function (just like the other boards in the system). The inputs to this board are the outputs from the four Energy board latches. This board is also at the other end of two forty-conductor ribbon cables from the computer some of which can be bi-directional. The FUNCTION bus (and its associated control signals) from the computer is routed via this board, but the overall destination of that bus should be considered as the whole VME cage. The SOFT RESET line from the computer is routed through this board in a similar fashion, but its destination is the STINTSET board (solely). In the opposite direction, the INTERRUPT line from the STINTSET board travels through this board. The outputs from this board (during RUN mode) are the VALID line and the rest of the ROUTING information.

One of the assignments of the Brain board has already been alluded to in earlier

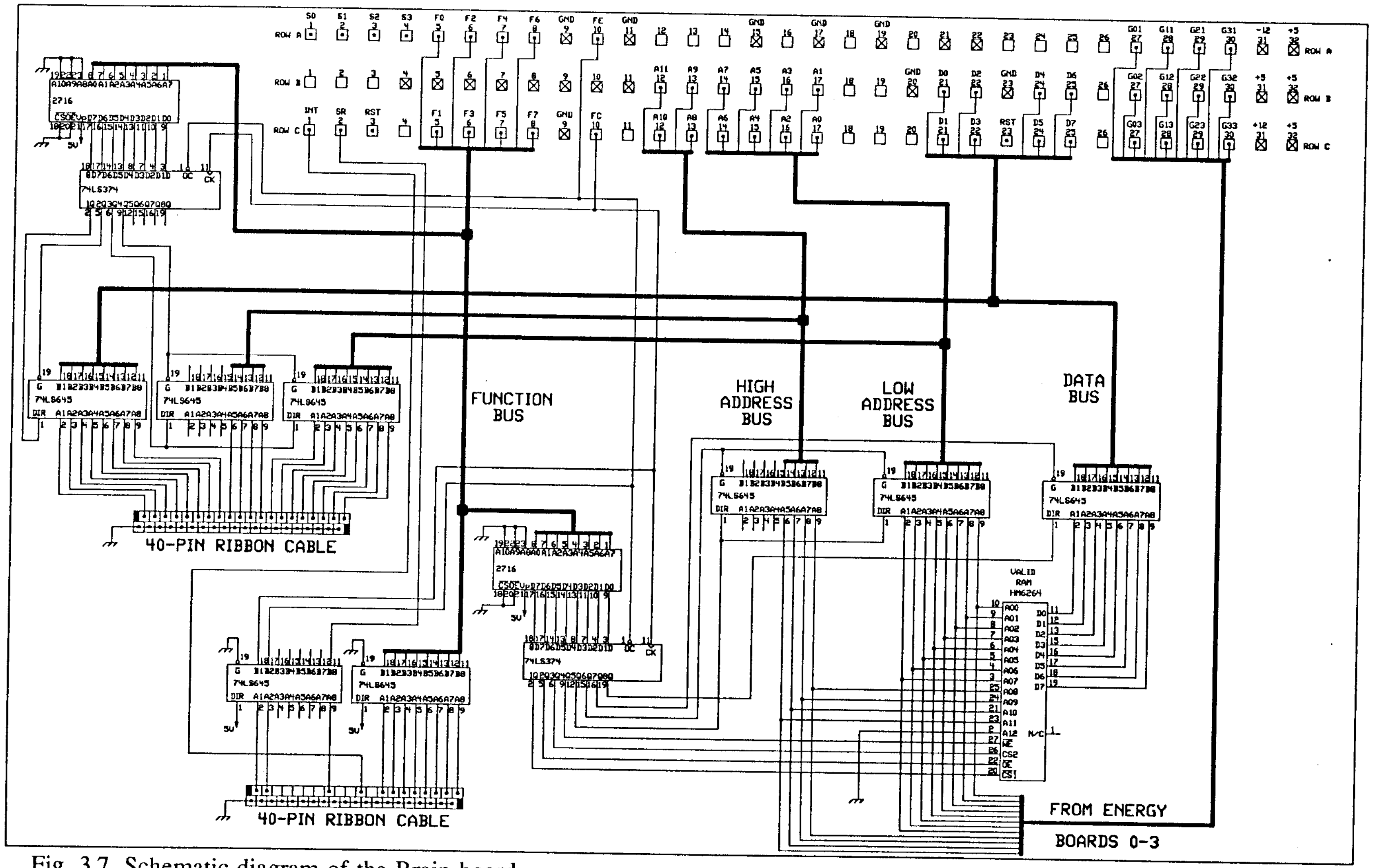


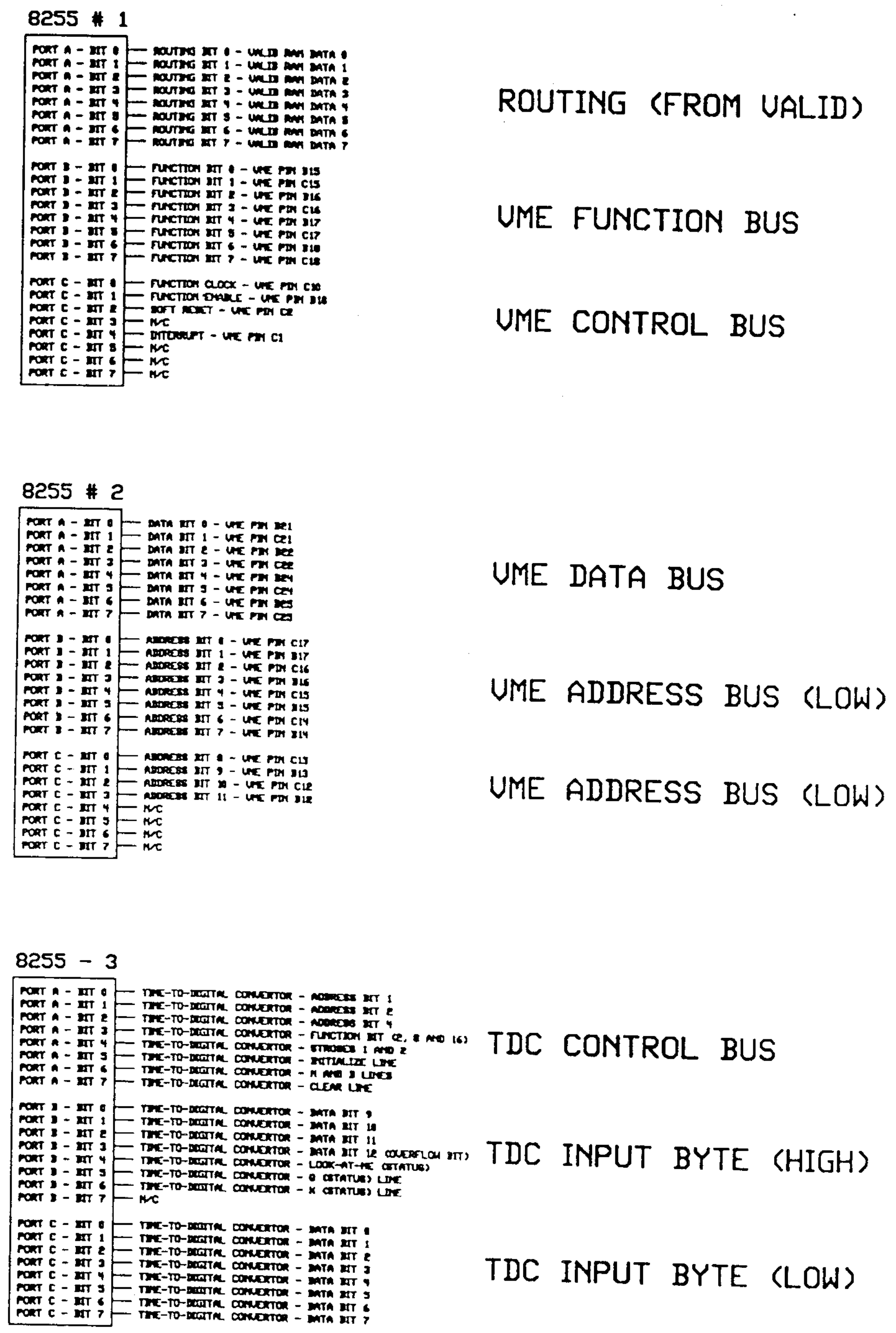
Fig. 3.7. Schematic diagram of the Brain board

discussions. The Brain board is where the VALID RAM is located, and thus it has the job of making the decision of whether or not an event is VALID. This is accomplished by a table-look-up similar to the Energy analysis on the energy boards. It is different inthat the contents of the VALID ram are "predetermined". Unlike the Energy boards, where the ADC output range determines the contents of the WINDOW RAM, the inputs to the VALID RAM are already established by the connections to the Energy boards. For example, if detector 0 receives a start gamma ray and detector 1 receives a stop, then the event is VALID. The memory location that is addressed by this set of outputs from the Energy boards is established as VALID, and it will always be that way. The result of this is that the computer routine that loads the contents of the VALID RAM is unchanging, unless some special circumstances warrant an unusual look-up table.

The other responsibility of the Brain board is the job of communication between the VME bus and the computer. There are two different 40 pin ribbon connectors on this board that go directly to the computer (every other conductor is grounded on these cables). There are also EPROMs on this board that control all of the necessary control bits on the board for the bi-directional bus transceivers at the end of this ribbon cable. The basic idea here is that there must be some way to control the communication between the computer and the card cage while the different modes are being executed. For example, the direction of communication between the computer and VME cage is different when the WINDOW RAM is being loaded then when the ROUTING information is being read.

3.5. The computer interface

The computer interface is centered around a parallel interface board (Quatech PXB-721) that plugs into a single slot in the PC bus. This board is based on three 8255 parallel interface chips. A block diagram is shown in figure 3.8. Access to these 8255's



8255 # 1
PORT A - BIT 0 — ROUTING BIT 0 - VALID RAM DATA 0
PORT A - BIT 1 — ROUTING BIT 1 - VALID RAM DATA 1
PORT A - BIT 2 — ROUTING BIT 2 - VALID RAM DATA 2
PORT A - BIT 3 — ROUTING BIT 3 - VALID RAM DATA 3
PORT A - BIT 4 — ROUTING BIT 4 - VALID RAM DATA 4
PORT A - BIT 5 — ROUTING BIT 5 - VALID RAM DATA 5
PORT A - BIT 6 — ROUTING BIT 6 - VALID RAM DATA 6
PORT A - BIT 7 — ROUTING BIT 7 - VALID RAM DATA 7
PORT B - BIT 0 — FUNCTION BIT 0 - VME PIN B15
PORT B - BIT 1 — FUNCTION BIT 1 - VME PIN C15
PORT B - BIT 2 — FUNCTION BIT 2 - VME PIN B16
PORT B - BIT 3 — FUNCTION BIT 3 - VME PIN C16
PORT B - BIT 4 — FUNCTION BIT 4 - VME PIN B17
PORT B - BIT 5 — FUNCTION BIT 5 - VME PIN C17
PORT B - BIT 6 — FUNCTION BIT 6 - VME PIN B18
PORT B - BIT 7 — FUNCTION BIT 7 - VME PIN C18
PORT C - BIT 0 — FUNCTION CLOCK - VME PIN C10
PORT C - BIT 1 — FUNCTION ENABLE - VME PIN B18
PORT C - BIT 2 — SOFT RESET - VME PIN C2
PORT C - BIT 3 — N/C
PORT C - BIT 4 — INTERRUPT - VME PIN C1
PORT C - BIT 5 — N/C
PORT C - BIT 6 — N/C
PORT C - BIT 7 — N/C
ROUTING (FROM VALID)
VME FUNCTION BUS
VME CONTROL BUS
8255 # 2
PORT A - BIT 0 — DATA BIT 0 - VME PIN B21
PORT A - BIT 1 — DATA BIT 1 - VME PIN C21
PORT A - BIT 2 — DATA BIT 2 - VME PIN B22
PORT A - BIT 3 — DATA BIT 3 - VME PIN C22
PORT A - BIT 4 — DATA BIT 4 - VME PIN B24
PORT A - BIT 5 — DATA BIT 5 - VME PIN C24
PORT A - BIT 6 — DATA BIT 6 - VME PIN B25
PORT A - BIT 7 — DATA BIT 7 - VME PIN C25
PORT B - BIT 0 — ADDRESS BIT 0 - VME PIN C17
PORT B - BIT 1 — ADDRESS BIT 1 - VME PIN B17
PORT B - BIT 2 — ADDRESS BIT 2 - VME PIN C16
PORT B - BIT 3 — ADDRESS BIT 3 - VME PIN B16
PORT B - BIT 4 — ADDRESS BIT 4 - VME PIN C15
PORT B - BIT 5 — ADDRESS BIT 5 - VME PIN B15
PORT B - BIT 6 — ADDRESS BIT 6 - VME PIN C14
PORT B - BIT 7 — ADDRESS BIT 7 - VME PIN B14
PORT C - BIT 0 — ADDRESS BIT 8 - VME PIN C13
PORT C - BIT 1 — ADDRESS BIT 9 - VME PIN B13
PORT C - BIT 2 — ADDRESS BIT 10 - VME PIN C12
PORT C - BIT 3 — ADDRESS BIT 11 - VME PIN B12
PORT C - BIT 4 — N/C
PORT C - BIT 5 — N/C
PORT C - BIT 6 — N/C
PORT C - BIT 7 — N/C
VME DATA BUS
VME ADDRESS BUS (LOW)
VME ADDRESS BUS (LOW)
8255 - 3
PORT A - BIT 0 — TIME-TO-DIGITAL CONVERTOR - ADDRESS BIT 1
PORT A - BIT 1 — TIME-TO-DIGITAL CONVERTOR - ADDRESS BIT 2
PORT A - BIT 2 — TIME-TO-DIGITAL CONVERTOR - ADDRESS BIT 4
PORT A - BIT 3 — TIME-TO-DIGITAL CONVERTOR - FUNCTION BIT (2, 8 AND 16)
PORT A - BIT 4 — TIME-TO-DIGITAL CONVERTOR - STROBES 1 AND 2
PORT A - BIT 5 — TIME-TO-DIGITAL CONVERTOR - INITIALIZE LINE
PORT A - BIT 6 — TIME-TO-DIGITAL CONVERTOR - A AND B LINES
PORT A - BIT 7 — TIME-TO-DIGITAL CONVERTOR - CLEAR LINE
PORT B - BIT 0 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 9
PORT B - BIT 1 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 10
PORT B - BIT 2 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 11
PORT B - BIT 3 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 12 (OVERFLOW BIT)
PORT B - BIT 4 — TIME-TO-DIGITAL CONVERTOR - LOOK-AT-ME (STATUS)
PORT B - BIT 5 — TIME-TO-DIGITAL CONVERTOR - Q (STATUS) LINE
PORT B - BIT 6 — TIME-TO-DIGITAL CONVERTOR - X (STATUS) LINE
PORT B - BIT 7 — N/C
PORT C - BIT 0 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 0
PORT C - BIT 1 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 1
PORT C - BIT 2 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 2
PORT C - BIT 3 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 3
PORT C - BIT 4 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 4
PORT C - BIT 5 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 5
PORT C - BIT 6 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 6
PORT C - BIT 7 — TIME-TO-DIGITAL CONVERTOR - DATA BIT 7
TDC CONTROL BUS
TDC INPUT BYTE (HIGH)
TDC INPUT BYTE (LOW)

Fig. 3.8. Block diagram of the computer interface board

is made through three 34 pin headers, each of which has a prototyping board plugged into it (directly - no cables). These boards mount directly onto the board that plugs directly into the PC bus, and the entire configuration can be closed up into the PC. Two of theseprototyping boards are the "other end" of the 40-conductor ribbon cables from the VME cage. It should be noted that one of these two headers is dedicated to the control of the VME bus system and the other is dedicated to the communication of information between the computer and the VME cage. The third header is used to communicate with the TDC, a complicated process in its own right. Examining the details of how the interface board works is quite complex and will not be discussed.

One detail that will be mentioned is that this board also has the interrupt generating capabilities that communicate directly with the microprocessor of the computer. By using this board, the system can interrupt the computer by asserting the correct lines on the interface board. The detailed communication with the microprocessor is taken care of by the internal workings of this board. It should be noted that the board must be correctly configured by the computer for this interrupt system to work.

4. Operation of the new spectrometer

The operation of the new system is almost completely computer controlled. Eliminating the complexities of the operation of the old system was one of the focus points of this work. The user must still be involved with the same sample preparation (and the associated start-up routines) as they were with the old system, but once the initial hurdles have been cleared the system can almost be operated from the computer keyboard. In the following discussion, a step-by-step procedure for the operation of the spectrometer will be briefly outlined. It will be assumed that there is a prepared sample is available and in place for the data collection run. It will also be assumed that the power is off in all parts of the system and that the computer has the control programs on the hard disk. Another important assumption is that the correct procedure is followed at each step in this process, thereby allowing the next steps in the sequence to be completed.

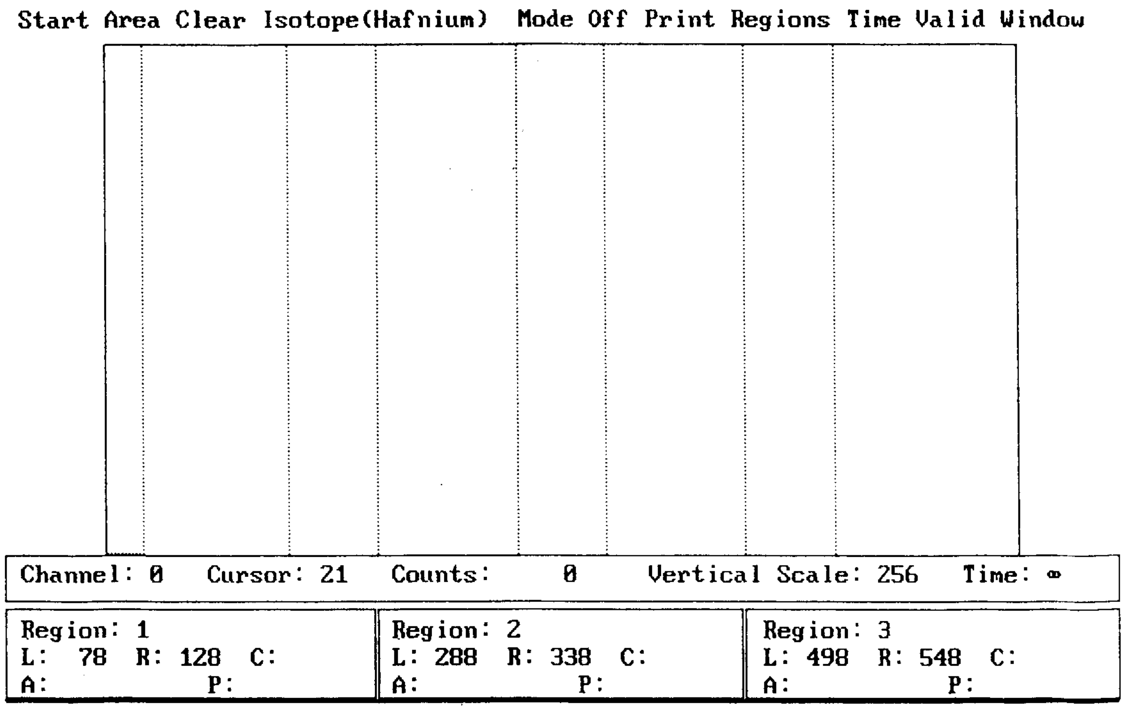
4.1. Procedure

The first step necessary in a data collection run is the initialization of the entire system. With the power off, the entire bus structure outlined in the previous chapter is in an indeterminate state. In order to prevent the system from "burning out" any components in the initial power-up phase, an initialization program was written. The first thing that must be done is turning on the computer. The user would then run a program called INITIALZ (all programs have the extension .EXE and are executable code). A message would appear that instructs the user to turn on the power, and then press a key. At this point, the power for **all** components would be turned on, the key pressed, and the computer would then configure the entire system in what is called the NULL state (nothing happens). The next program that is run is called VMEASS. This program sets up the ASSembly language routines that are used in the data gathering process. This is the program that sets up the interrupt service routine and all of the capabilities associated with collecting data. After the preparatory phase has been completed, the system is ready to be calibrated.

For this phase, there is a program called VMEMCA which must be executed. Upon doing this, the user would a screen that looks like figure 4.1. As the name implies, this is where the spectrometer performs all of the tasks normally associated with a Multi-Channel Analyzer (MCA). This is the phase of operation that was referred to in earlier chapters as the CALIBRATION. It is the time when the computer interrogates the individual Energy boards to find out which memory locations in the WINDOW RAM should be loaded with the values corresponding to the start and stop gamma rays. The computer and spectrometer actually function as a four separate multi-channel analyzers in this stage (each one tuned to a particular detector).

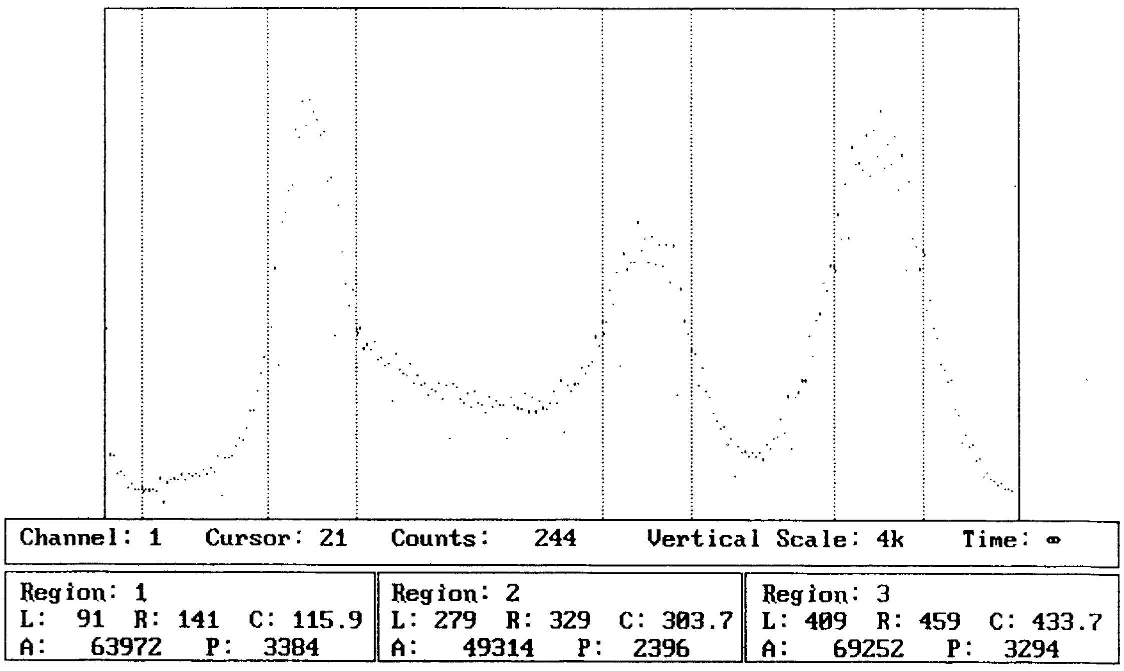
After gathering data for a while (in similar fashion to an MCA), the user would see a screen that looks like figure 4.2. Note the single line on the left hand side of the boxed-in area. That line represents the cursor that is found on commercial MCAs. It is controlled from the keyboard, and with it the user can find out the number of counts in a given channel. Of particular interest in this discussion are the three "paired" lines that are surrounding the peaks in the data (in the figure). These paired lines represent the three windows that have been mentioned previously. If the user were to press "w" while the screen looked like it does in the figure, the "look-up table" in the channel that is being interrogated would be set. The RAM values corresponding to the positions between the cursors would be the windows set within the WINDOW RAM. The user would set all four channels in a similar manner. After that, the user would set up the table in the VALID RAM by pressing a "v". At this point the VMEMCA program could be exited.

The next program that would be run is the actual data gathering program, named VMEPAC. Upon execution of this program, the user would see a screen that looks like figure 4.3. This program allows the user to examine the data and to control the spectrometer. After collecting data for a period of time and then selecting the option enabling the examination of data, the user would see a screen that looks like figure 4.4.


Start Area Clear Isotope(Hafnium) Mode Off Print Regions Time Valid Window

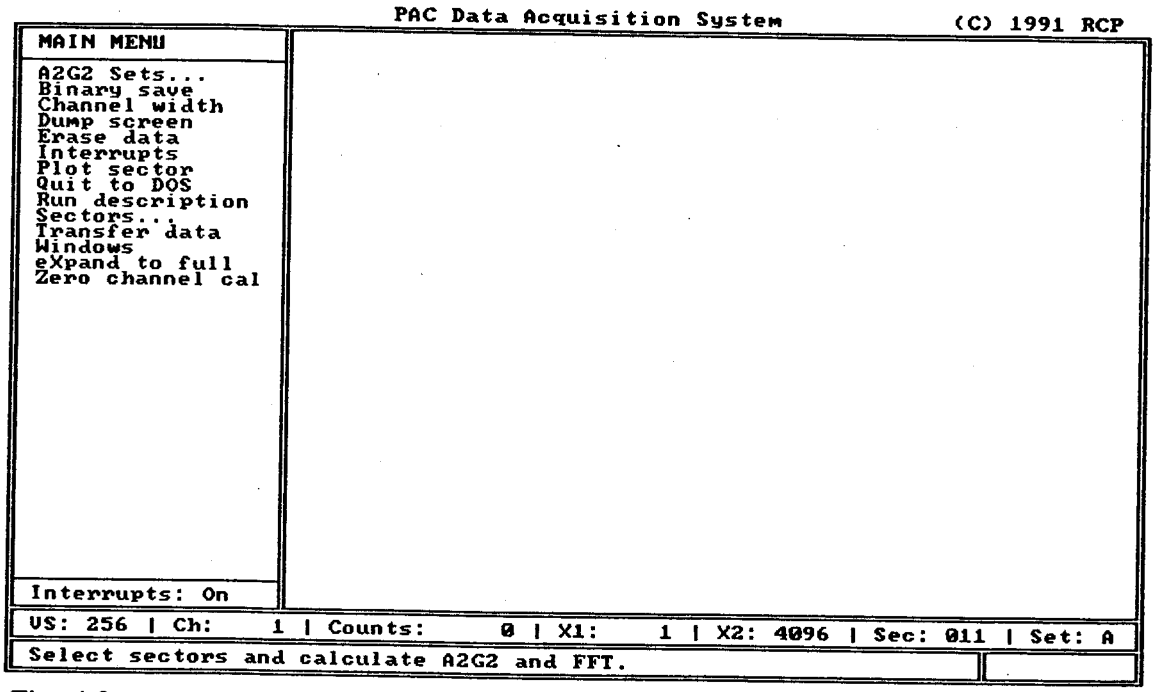
```
Channel: 0   Cursor: 21   Counts:      0     Vertical Scale: 256   Time: ∞

Region: 1                Region: 2                Region: 3
L:  78  R: 128  C:       L: 288  R: 338  C:       L: 498  R: 548  C:
A:          P:           A:          P:           A:          P:
```

Fig. 4.1. The computer screen as seen when starting the VMEMCA program



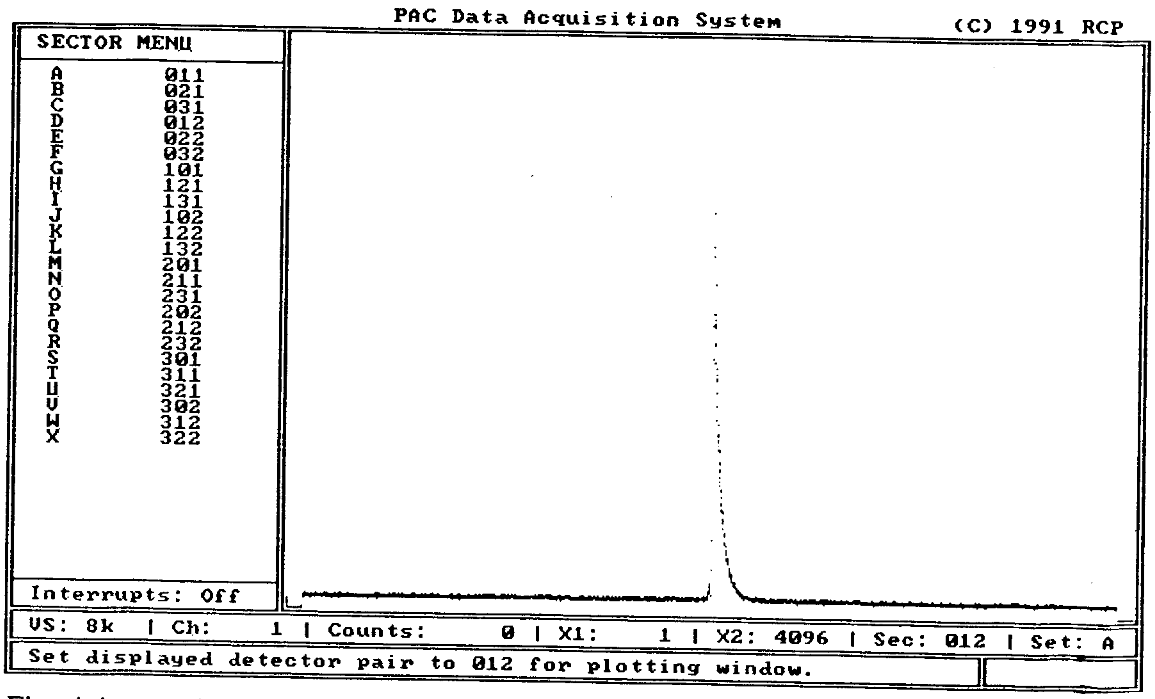
```
Channel: 1   Cursor: 21   Counts:   244    Vertical Scale: 4k    Time: ∞

Region: 1                  Region: 2                  Region: 3
L:  91  R: 141  C: 115.9   L: 279  R: 329  C: 303.7   L: 409  R: 459  C: 433.7
A:   63972   P:  3384      A:   49314   P:  2396      A:   69252   P:  3294
```

Fig. 4.2. The computer screen as seen while using the VMEMCA program to set windows



```
Interrupts: On
VS: 256 | Ch:     1 | Counts:        0 | X1:      1 | X2: 4096 | Sec: 011 | Set: A
Select sectors and calculate A2G2 and FFT.
```

Fig. 4.3. The computer screen as seen when starting the VMEPAC program



```
Interrupts: Off
VS: 8k  | Ch:     1 | Counts:        0 | X1:      1 | X2: 4096 | Sec: 012 | Set: A
Set displayed detector pair to 012 for plotting window.
```

Fig. 4.4. The computer screen as seen while using the VMEPAC program to collect data

4.2. Data analysis

The data analysis for the new spectrometer is not much different than it is for the old spectrometer, except for the fact that there is a great deal more raw data that can be analyzed (see table 4.1). The increase in the amount of raw data sectors also means that there are more possible ways to form A_2G_2 (see tables 4.2 and 4.3). The analysis method is identical to the method used for the old spectrometer, however.

It might be useful to point out that the use of three windows is not restricted to the concept of having a single start and two different stop channels. The other possible scenario that it may be useful for is to split certain gamma ray peaks with two windows. There may be some reason why the data from one part of a peak may be better than another (see chapter 5). There is a downside to having this large amount of data. The storage space is very large (about 450 Kbytes) and it takes considerably longer (a few days) to collect a statistically good set of data than it did with the old system. However, this is presumably outweighed by the increase in the amount of information that is obtained.

Table 4.1. The raw data sectors for the new system

SECTOR NUMBER	SECTOR LABEL	START GAMMA	STOP GAMMA	STOP ENERGY WINDOW
0	011	0	1	LOW
1	021	0	2	LOW
2	031	0	3	LOW
3	012	0	1	HIGH
4	022	0	2	HIGH
5	032	0	3	HIGH
6	101	1	0	LOW
7	121	1	2	LOW
8	131	1	3	LOW
9	102	1	0	HIGH
10	122	1	2	HIGH
11	132	1	3	HIGH
12	201	2	0	LOW
13	211	2	1	LOW
14	231	2	3	LOW
15	202	2	0	HIGH
16	212	2	1	HIGH
17	232	2	3	HIGH
18	301	3	0	LOW
19	311	3	1	LOW
20	321	3	2	LOW
21	302	3	0	HIGH
22	312	3	1	HIGH
23	322	3	2	HIGH

Table 4.2. The different A_2G_2 sets for stop window 1 and how they are formed

A_2G_2 SET	90° SECTORS	180° SECTORS	FORMULA (* SECTOR FORM)
A	031,121	021,131	$\frac{(021*131)^{1/2} - (031*121)^{1/2}}{(021*131)^{1/2} + 2*(031*121)^{1/2}}$
B	011,321	021,311	$\frac{(021*311)^{1/2} - (011*321)^{1/2}}{(021*311)^{1/2} + 2*(011*321)^{1/2}}$
C	101,231	131,201	$\frac{(131*201)^{1/2} - (101*231)^{1/2}}{(131*231)^{1/2} + 2*(131*201)^{1/2}}$
D	211,301	201,311	$\frac{(201*311)^{1/2} - (211*301)^{1/2}}{(201*311)^{1/2} + 2*(211*301)^{1/2}}$
E	011,121,231,301	021,131,201,311	$\frac{(021*131*201*311)^{1/4} - (011*121*231*301)^{1/4}}{(021*131*201*311)^{1/4} + 2*(011*121*231*301)^{1/4}}$
F	011,101,231,321	021,131,201,311	$\frac{(021*131*201*311)^{1/4} - (011*101*231*321)^{1/4}}{(021*131*201*311)^{1/4} + 2*(011*101*231*321)^{1/4}}$
G	031,121,211,301	021,131,201,311	$\frac{(021*131*201*311)^{1/4} - (031*121*211*301)^{1/4}}{(021*131*201*311)^{1/4} + 2*(031*121*211*301)^{1/4}}$
H	031,101,211,321	021,131,201,311	$\frac{(021*131*201*311)^{1/4} - (031*101*211*321)^{1/4}}{(021*131*201*311)^{1/4} + 2*(031*101*211*321)^{1/4}}$
I	011,031,101,121, 211,231,301,321	021,131,201,311	$\frac{\{(021*131*201*311)^{1/4} - (011*031*101*121*211*231*301*321)^{1/8}\}}{\{(021*131*201*311)^{1/4} + 2*(011*031*101*121*211*231*301*321)^{1/8}\}}$

* In sector form the sector label represents the number of background-corrected counts in the sector. There is also an implicit function of time in this equation. In other words:

$$ijk = C_{ijk}(t) = D_{ijk}(t) - B_{ijk}(t)$$

in the above table.

Table 4.3. The different A_2G_2 sets for stop window 2 and how they are formed

A_2G_2 SET	90° SECTORS	180° SECTORS	FORMULA (* SECTOR FORM)
J	032,122	022,132	$\frac{(022*132)^{1/2} - (032*122)^{1/2}}{(022*132)1/2 + 2*(032*122)^{1/2}}$
K	012,322	022,312	$\frac{(022*312)^{1/2} - (012*322)^{1/2}}{(022*312)^{1/2} + 2*(012*322)^{1/2}}$
L	102,232	132,202	$\frac{(132*202)^{1/2} - (102*232)^{1/2}}{(132*232)^{1/2} + 2*(132*202)^{1/2}}$
M	212,302	202,312	$\frac{(202*312)^{1/2} - (212*302)^{1/2}}{(202*312)^{1/2} + 2*(212*302)^{1/2}}$
N	012,122,232,302	022,132,202,312	$\frac{(022*132*202*312)^{1/4} - (012*122*232*302)^{1/4}}{(022*132*202*312)^{1/4} + 2*(012*122*232*302)^{1/4}}$
O	012,102,232,322	022,132,202,312	$\frac{(022*132*202*312)^{1/4} - (012*102*232*322)^{1/4}}{(022*132*202*312)^{1/4} + 2*(012*102*232*322)^{1/4}}$
P	032,122,212,302	022,132,202,312	$\frac{(022*132*202*312)^{1/4} - (032*122*212*302)^{1/4}}{(022*132*202*312)^{1/4} + 2*(032*122*212*302)^{1/4}}$
Q	032,102,212,322	022,132,202,312	$\frac{(022*132*202*312)^{1/4} - (032*102*212*322)^{1/4}}{(022*132*202*312)^{1/4} + 2*(032*102*212*322)^{1/4}}$
R	012,032,102,122, 212,232,302,322	022,132,202,312	$\frac{\{(022*132*202*312)^{1/4} - (012*032*102*122*212*232*302*322)^{1/8}\}}{\{(022*132*202*312)^{1/4} + 2*(012*032*102*122*212*232*302*322)^{1/8}\}}$

* In sector form the sector label represents the number of background-corrected counts in the sector. There is also an implicit function of time in this equation. In other words:

$$ijk = C_{ijk}(t) = D_{ijk}(t) - B_{ijk}(t)$$

in the above table.

5. The experimental verification

In order to test to see if the spectrometer was working correctly, two different experimental runs were conducted. The sample that was used for both was room temperature HfO_2. In the literature[26,27] it is characterized as having four HfO_2 units in a face-centered monoclinic structural arrangement. These Hf atoms are located at each face and corner of the unit cell, and each is surrounded by eight oxygen nearest-neighbors. The sample that was prepared for this experimental testing by irradiation, which produces the radioactive isotope ^{181}Hf that decays by beta decay into the PAC probe ^{181}Ta. The relevant properties of the of the radioactive probe are shown in table 5.1.

The choice of Hf as the radioactive isotope was deliberate, since the decay scheme (see figure 5.1) produces a daughter isotope (^{181}Ta) that has a single start gamma ray (133 keV) and two potential stop gamma rays (346 keV and 482 keV). The normal cascade that is used is 133 keV $\rightarrow$ 482 keV, but it is possible to monitor the other cascade which is 133 keV $\rightarrow$ 346 keV. However, this cannot be done without the problem associated with the overlap of the 136 keV transition peak with the 133 keV start peak. Since the half-life of the $9/2^+$ state is so short (40ps), the result of this overlap is a delta-function at time zero (T_0)in the raw data sector corresponding to the 133 keV $\rightarrow$ 346 keV. The reason for the delta function in those sectors is that the 136 keV gamma ray is erroneously recorded by the electronics as a start gamma ray and the 346 keV gamma ray is interpreted as the stop gamma ray.

The first data collection run was performed using both of the cascades for data collection. The start window was set on the 133 keV gamma ray peak, the low energy stop window was set on the 346 keV gamma ray peak, and the high energy stop window was set on the 482 keV gamma ray peak. In this way, the 12 sectors corresponding to the low energy stop window recorded the 133 keV $\rightarrow$ 346 keV cascade and the 12 sectors corresponding to the high energy stop window recorded the 133 keV $\rightarrow$ 482 keV cascade (see table 4.1 for the raw data sector list). In this way, the concept of more than two windows could be employed in data gathering.

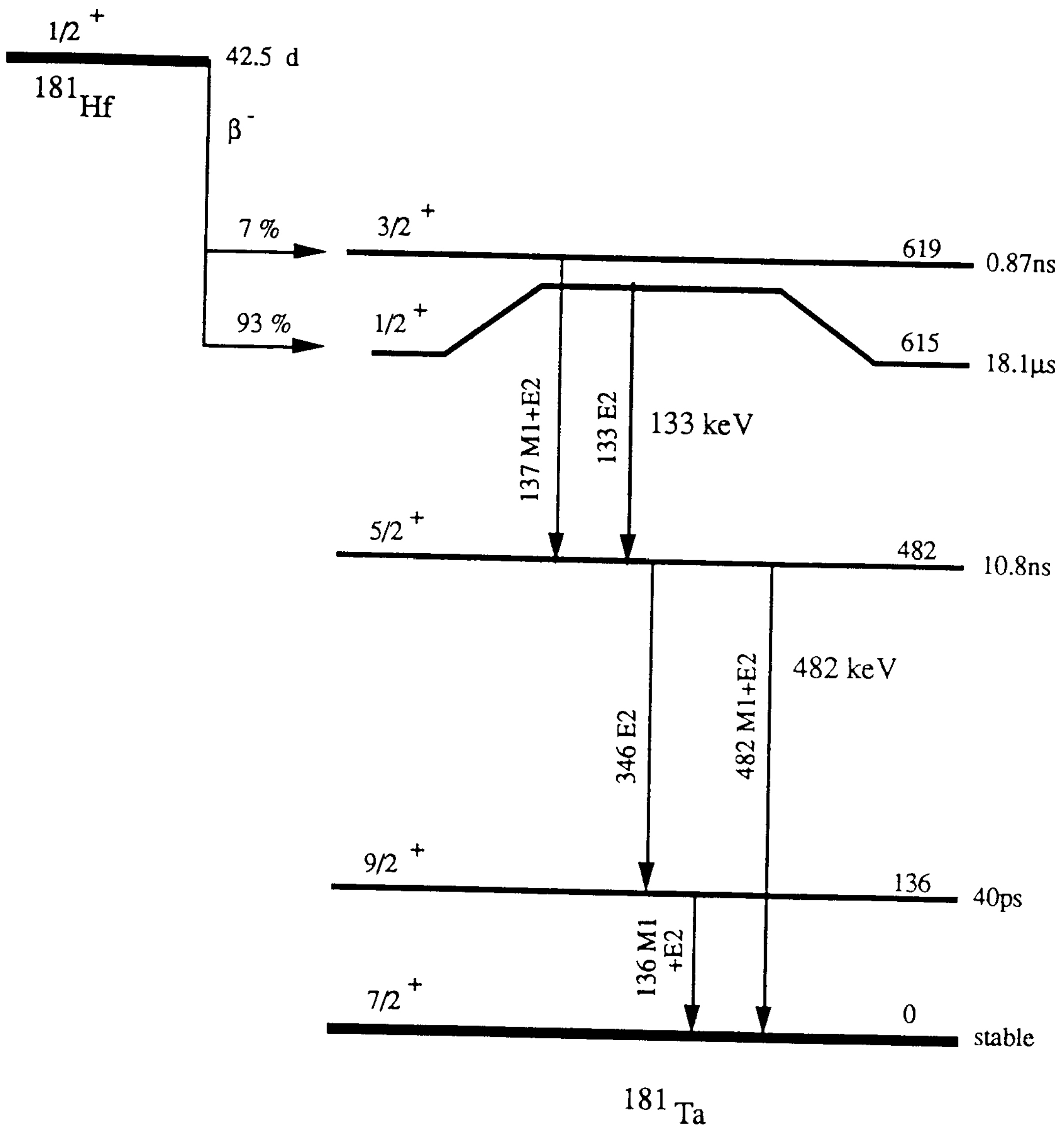


Fig. 5.1. The decay scheme[28] for ^{181}Hf → ^{181}Ta

Table 5.1. Summary[29] of the properties of ^{181}Ta

Parent Half-life (^{181}Hf)	42.5 days
Start gamma ray energy	133 keV
Energy* of stop gamma ray #1	346 keV
Energy of stop gamma ray #2	482 keV
Nuclear spin sequence	$1/2^+ \rightarrow 5/2^+ \rightarrow 7/2^+$
Angular-correlation coefficient A_{22}	-0.295(5)
Angular-correlation coefficient A_{44}	-0.069
Half-life of the intermediate state	10.8 ns
Electric quadrupole moment	+ 2.5 b

* For this work, the cascade 133 keV → 346 keV was measured as well as the cascade 133 keV → 482 keV which is normally used for PAC spectroscopy.

Some typical data resulting from the low energy stop window is shown in figures 5.2 and 5.3. Notice the presence of the delta-function that was described above in both of the raw data sectors shown. The presence of this in the data around T_0 caused the data analysis to be very difficult, since it effects the A_2G_2 calculation in this critical area (see figure 5.4). The fit (solid line), which had to be calculated in the region away from T_0, is not very good for this reason. Some typical data for the high energy stop window is shown in figures 5.5 and 5.6. Notice that there is no delta function in these sectors, which corresponds to the expected result. A typical A_2G_2 plot resulting from this data set is shown in figure 5.7. The fit to this data set could be done much closer to T_0, and the result is that it looks much better. The error bars in both A_2G_2 sets are large due to the restriction on how long a run could be made without an data overflow in the delta-function region.

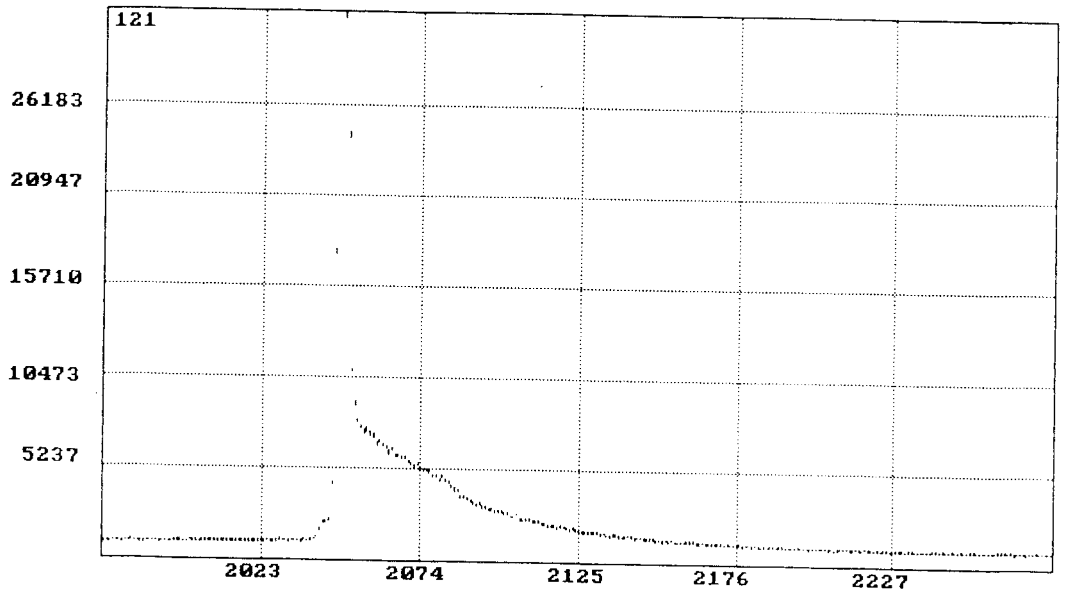


Fig. 5.2. Typical 90° raw data for a sector with the stop window set on the low energy stop gamma ray peak

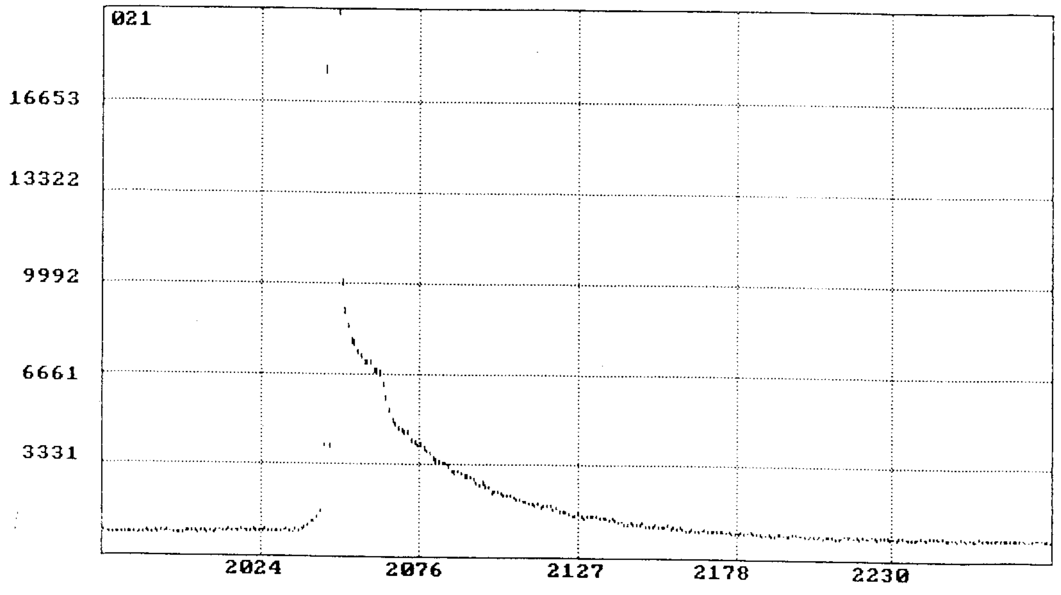


Fig. 5.3. Typical 180° raw data for a sector with the stop window set on the low energy stop gamma ray peak

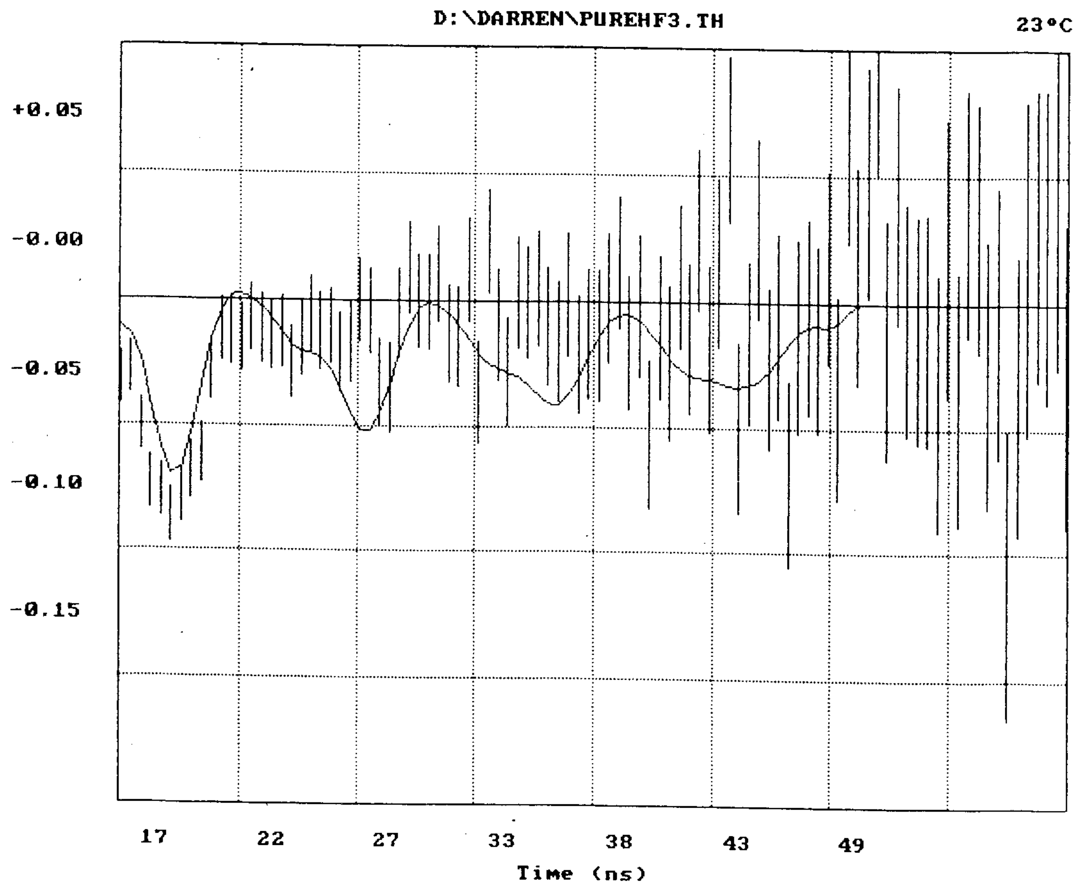


Fig. 5.4. A typical fitted A_2G_2 plot for a data set with the windows set on the low energy stop gamma ray peak

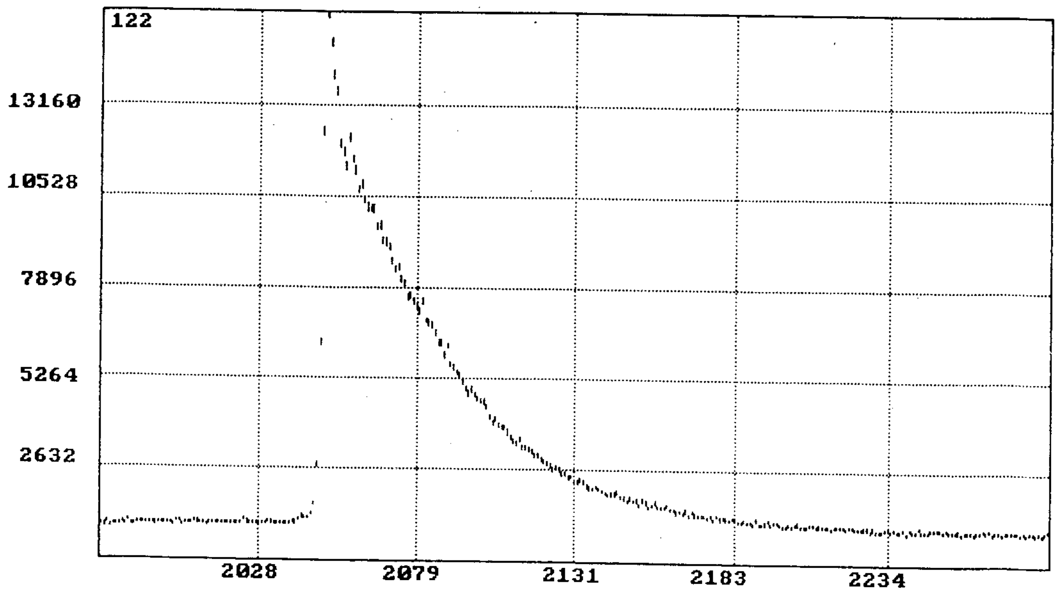


Fig. 5.5. Typical 90° raw data for a sector with the stop window set on the high energy stop gamma ray peak

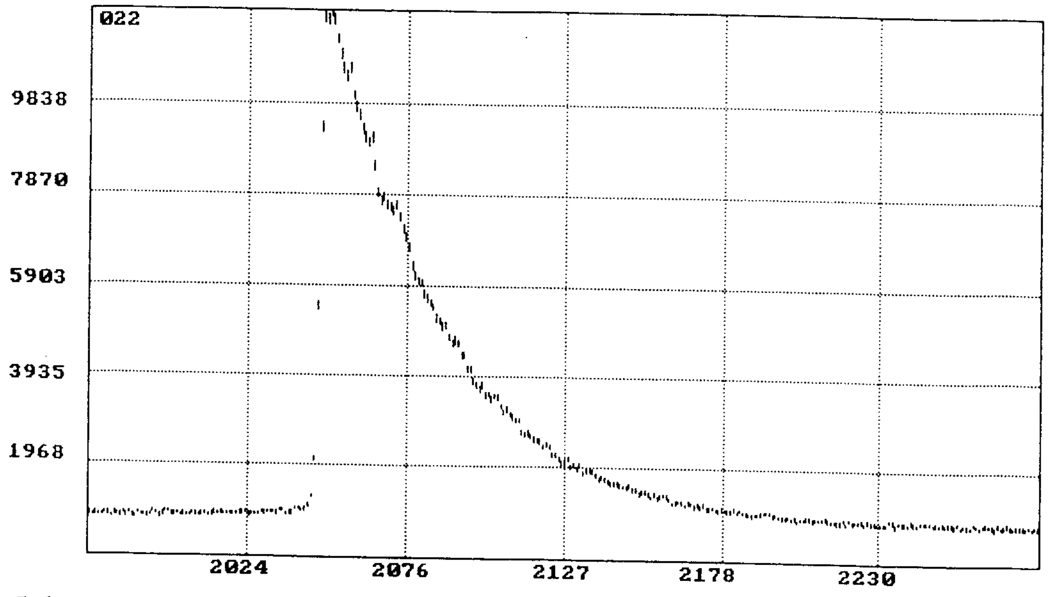


Fig. 5.6. Typical 180° raw data for a sector with the stop window set on the high energy stop gamma ray peak

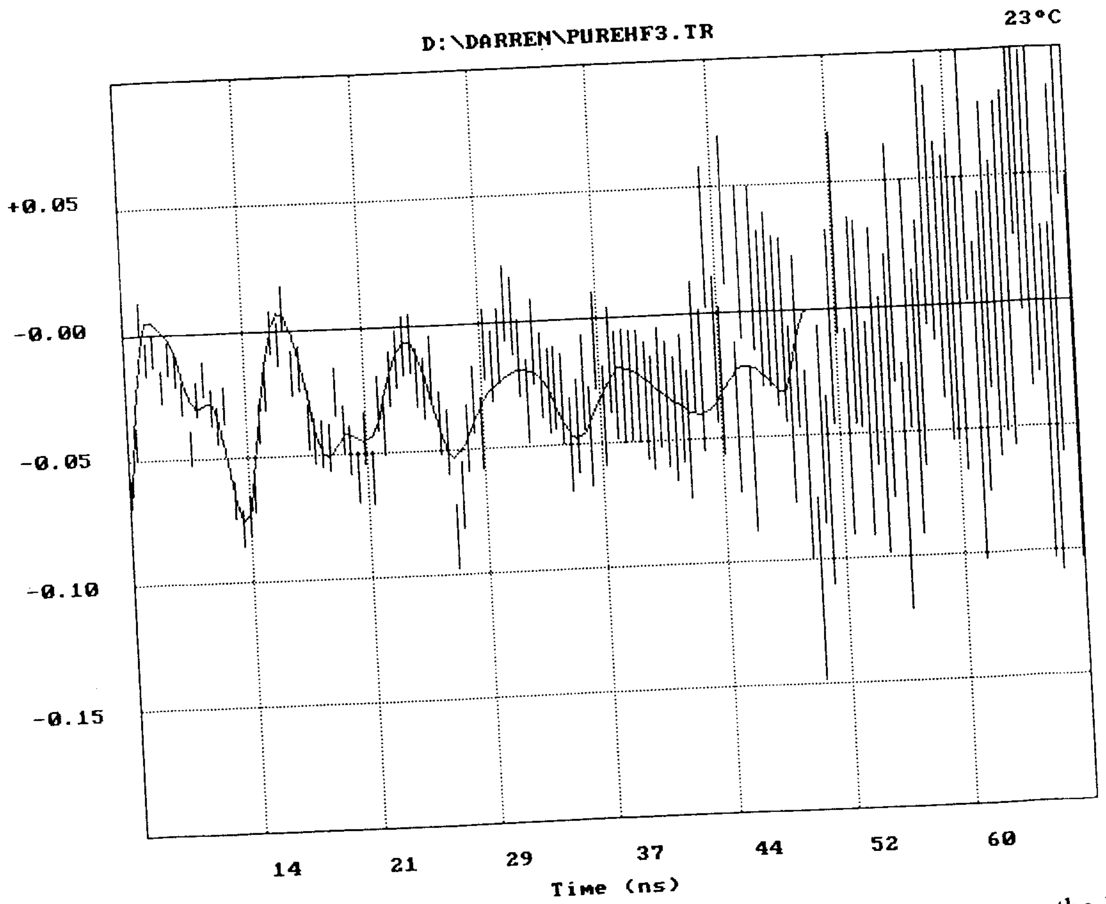


Fig. 5.7. A typical fitted A_2G_2 plot for a data set with the windows set on the high energy stop gamma ray peak

The second run was performed on the same HfO_2 sample, but the data collection was done quite differently. In this data collection run, the 133 keV $\rightarrow$ 482 keV cascade was the only one that was monitored. A problem that had arisen in the previous use of the ^{181}Ta probe was investigated by performing a very interesting experiment. In the past, there were certain situations when the delta function associated with the 346 keV cascade appeared in the raw data. This was finally determined to be the result of the overlap of edges of the Gaussian tails of the two energy peaks. When setting the stop window on the 482 keV peak with the old spectrometer, it was very difficult to see the appropriate place to set the lower level threshold (LLT) for the stop window. If the threshold was set too low, the resulting data set could contain some of the manifestations of this overlap. The idea came to mind to use the new system to split the 482 keV stop gamma ray peak with the two stop windows. The prediction was that the raw data sectors with the stop window set on the lower half of the "split" peak would contain some signs off the delta function, but the data set with the stop window set on the upper half of the peak would not show any signs of the anomaly.

Some typical raw data sets for sectors with the stop windows set on the lower half of the peak are shown in figures 5.8 and 5.9. The surprising result was that there was no significant indication of any delta function at T_0. One probable reason for this was that the window was not set "sloppily" enough. In the old spectrometer, the window setting is not as exact as it is in the new system, and it would therefore be much easier to set the windows in such a way as to overlap the 346 keV peak. A typical A_2G_2 plot for the data set that has the window set on the lower half of the 482 keV peak is shown in figure 5.10. The fit seems to be quite reasonable, which tends to support the idea that the window setting was actually too well done to overlap the edge of the 346 keV peak.

Some typical raw data for sectors with the stop window set on the upper half of the 482 keV peak are shown in figures 5.11 and 5.12. These sectors show the expected result, which is the absence of the delta function near T_0. The raw data in these sectors is very similar in appearance to the raw data from the sectors that had the stop window set on the lower half of the window. This provides further support for the hypothesis that the windows for the other sectors were set too closely to the 482 keV peak to show the

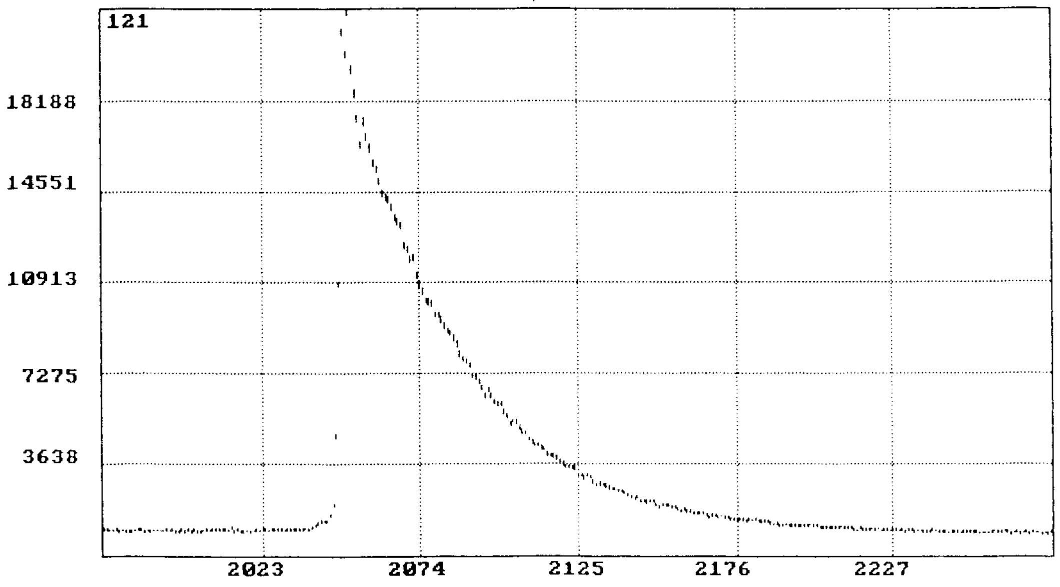


Fig. 5.8. Typical 90° raw data for a sector with the stop window set on the lower half of the high energy stop gamma ray peak

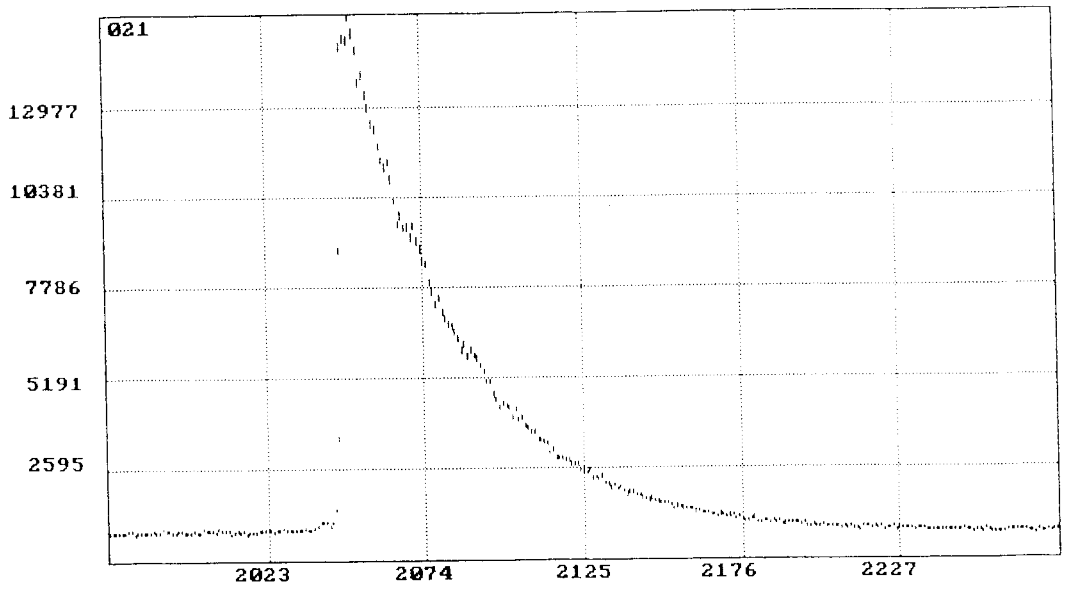


Fig. 5.9. Typical 180° raw data for a sector with the stop window set on the lower half of the high energy stop gamma ray peak

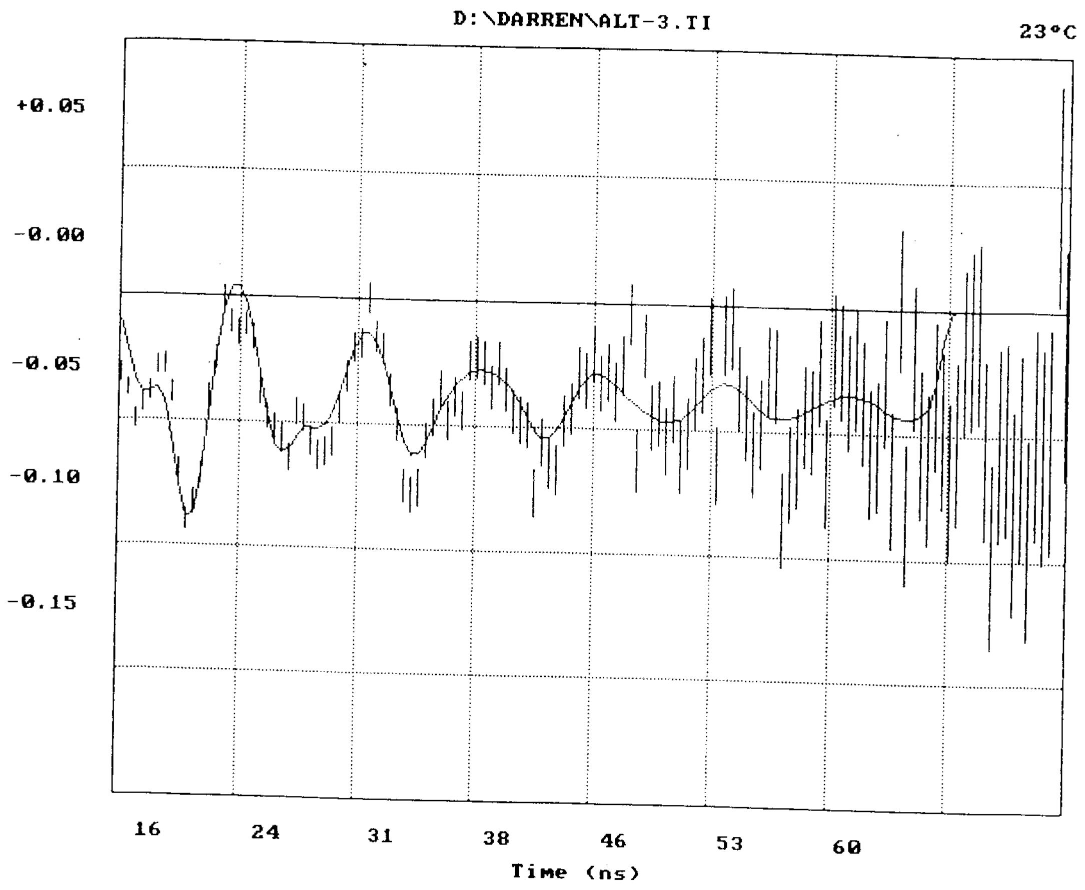


Fig. 5.10. A typical fitted A_2G_2 plot for a data set with the windows set on the lower half of the high energy stop gamma ray peak

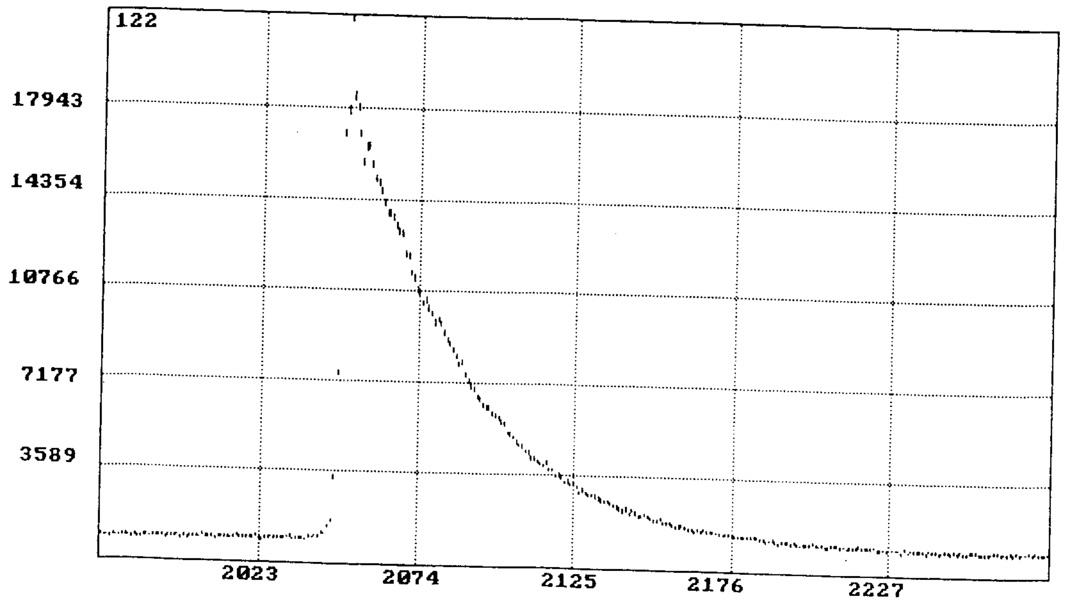


Fig. 5.11. Typical 90° raw data for a sector with the stop window set on the high energy stop gamma ray peak

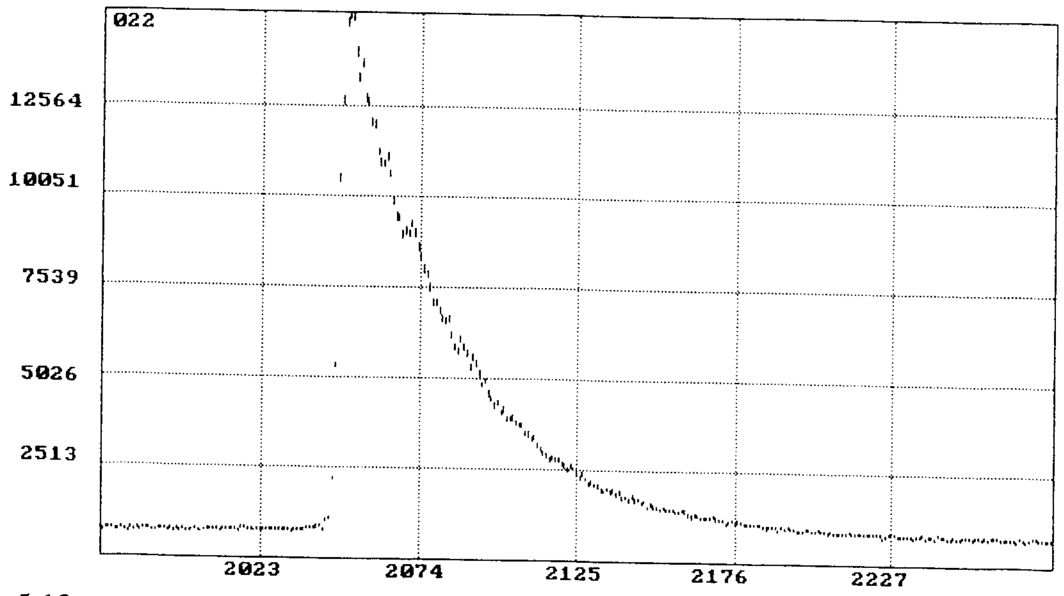


Fig. 5.12. Typical 180° raw data for a sector with the stop window set on the high energy stop gamma ray peak

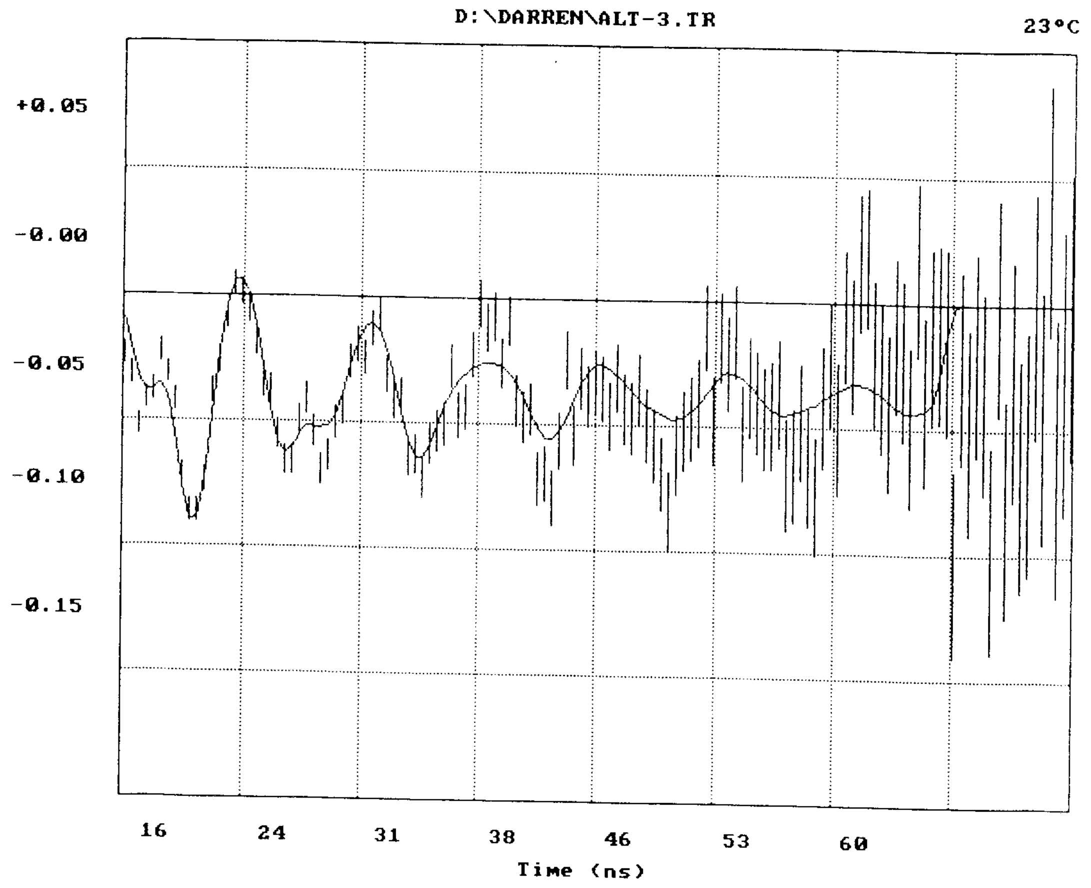


Fig. 5.13. A typical fitted A_2G_2 plot for a data set with the windows set on the upper half of the high energy stop gamma ray peak

influence from the other gamma ray peak. A typical A_2G_2 plot for the data sets with the stop window set on the upper half of the 482 keV peak is shown in figure 5.13. Just as in the set from the other stop window, the fit appears to be quite reasonable.

Since the windows seemed to be aligned too well to show the effect of the overlap of the upper edge of the low energy stop peak and the lower edge of the stop window, the analysis of the data sets was used for a different purpose. The data from both stop windows appeared to be clean enough to allow a comparison between the frequencies from fitting the A_2G_2 from the two data sets. The expected result was that the two sets would give nearly identical results. The fitted A_2G_2 plot for the lower half of the high energy stop window is shown in figure 5.14. The resulting fit can be considered quite good, even though there are some areas in which the data and fit seem to differ a little bit. The fitted A_2G_2 plot for the lower half of the high energy stop window is shown in figure 5.15. Again, the resulting fit is quite good and agrees quite well with the result from the other stop window. The frequencies from the fit of the data resulting lower half of the high energy stop peak were:

$$\omega_1 = 829 \pm 2 \ \mathit{Mrad/s}$$
$$\omega_2 = 1447 \pm 5 \ \mathit{Mrad/s}$$

and for the upper half of the high energy stop peak were:

$$\omega_1 = 827 \pm 3 \ \mathit{Mrad/s}$$
$$\omega_2 = 1448 \pm 5 \ \mathit{Mrad/s}$$

These values were in close agreement with each other, which indicates that the system is functional to the point that it give consistent results. The real verification is established when these results are compared to those of other groups. By consulting the literature, two sets of results were found and are shown on the following page. Upon inspection, the results obtained by this new system can be seen to agree quite well with work done on the same material by other groups.

The results obtained by Yeshurun and Arad[30] using a least squares fit method were:

$$\omega_1 = 830 \pm 14 \quad Mrad/s$$
$$\omega_2 = 1443 \pm 24 \quad Mrad/s$$

and using a Fourier transform method they obtained:

$$\omega_1 = 834 \pm 34 \quad Mrad/s$$
$$\omega_2 = 1463 \pm 48 \quad Mrad/s$$

The results obtained by Gardner and Prestwich[31] were:

$$\omega_1 = 819 \pm 10 \quad Mrad/s$$
$$\omega_2 = 1483 \pm 25 \quad Mrad/s$$

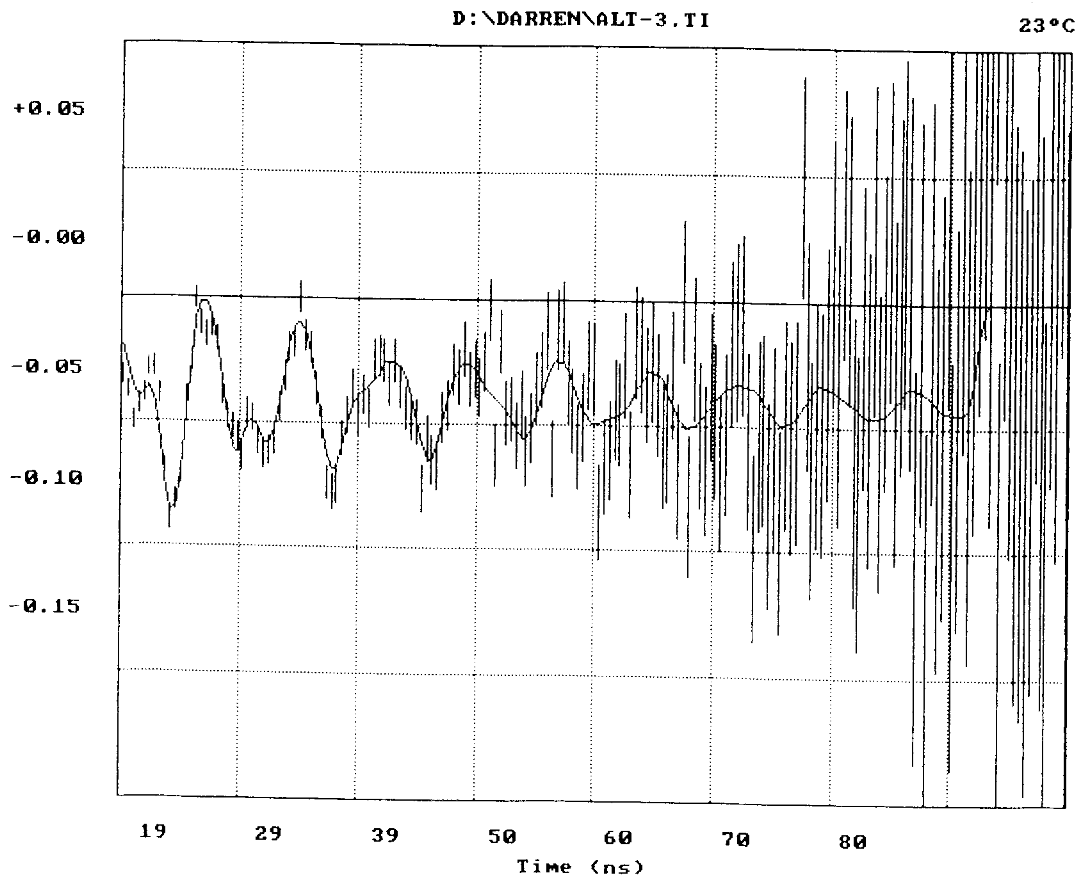


Fig. 5.14. Plot of the experimental A_2G_2 and fit used for frequency comparisons (first set)

The frequencies for this data set were:

$$\omega_1 = 829 \pm 2 \ \mathit{Mrad/s}$$
$$\omega_2 = 1447 \pm 5 \ \mathit{Mrad/s}$$

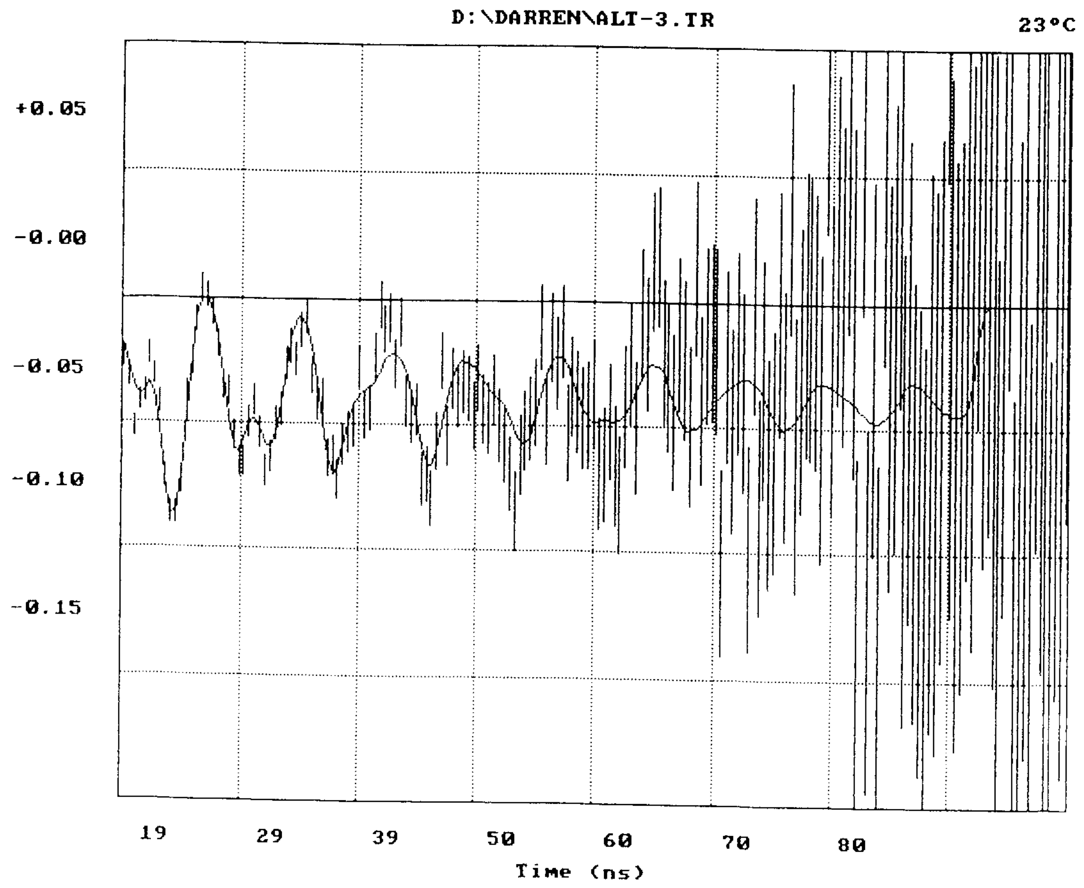


Fig. 5.15. Plot of the experimental A_2G_2 and fit used for frequency comparisons (second set)

The frequencies for this data set were:

$$\omega_1 = 827 \pm 3 \ Mrad/s$$
$$\omega_2 = 1448 \pm 5 \ Mrad/s$$

6. Summary and conclusions

There are several features of the new spectrometer that make it a revolutionary design. The significant new features are (i) the digitization of the energy pulses, (ii) the flexibility of the design, and (iii) the use of a four-channel TDC. All three of these features combine to make the new design a significant improvement over the existing systems. Any single one of these would be an improvement, thus the implementation of all three of them puts the new system in a class by itself.

The first feature that sets it apart from the older designs is the digitization of the energy pulses that takes place in the energy analysis section of the spectrometer. The old system uses analog circuitry to see if the pulse falls within a range of voltages that correspond one of two preset "windows", whereas the new system uses a flash-type ADC to digitize the voltage pulse. The associated digital circuitry for this process was chosen and implemented in such a way that there should be no problems with "drift" associated with fluctuations in temperature and power-supply variations. The old system had potentiometers that had to be checked periodically to see if the "windows" had moved, a process that was quite bothersome. There is also the advantage associated with this "windowing" being performed using digital circuitry in that the computer can process this digital information. When using the new system the user can set up and check this information while sitting at the computer, whereas the old system's analog information cannot be monitored by a computer. There are other advantages associated with this digital processing in that more windows can be used in the new system, while the old system is restricted in the hardware to having only two windows (a single start and a single stop). If the software used to run the new spectrometer were modified, it would be possible to have seven total windows. This opens up the possibility of performing new experiments (such as monitoring more than one cascade) that the old spectrometer cannot currently perform.

The second feature in the new design that make its a better system is the way that it is layed out and conceived. The design was done with flexibility in mind, which allows the user to have better control of the system. For example, the system is designed in such a way that allows the computer to have access to the data that is used in every part of the process. If there seems to be a problem in some section, or there is a need to monitor something that is not currently observed, then the software can be modified in order to allow just that feature. The convenience of using a computer to control the system also makes the system much nicer to use, since the software can be modified at any time to incorporate new ideas, or to fix any problems as they are encountered. Also, the control program can be written in such a way that it is easy to use and user-friendly.

The third major improvement in the new system is the use of the four-channel TDC instead of a single-channel TAC-ADC combination. This difference means that there are no invalid data sectors associated with "ORing" the starts and stops. This also means that the user has the ability to change the time scale by the flick of a switch on the TDC (the older system has to be significantly modified to change probes). Therefore, when a variety of experiments needs to be performed, there is no significant down-time associated with converting the spectrometer. Another thing to note is that with the advances in such technologies as GaAs, the future for improved TDCs looks promising. If one becomes available that performs better than the one employed in the present system, then it can be installed with only minor alterations.

The overall goal of this work was to design and build a new spectrometer that was an improvement over the old systems. The overall complexity of the system made its final design much more complicated than was ever anticipated. However, the results of the analysis done on data that was collected using the new system clearly demonstrates that the system is fully operational. The ease of use and the differences from the operation of the old spectrometer demonstrate that it is in fact a revolutionary design when compared to the older systems. Based on this, it is clear that the goal of this work was realized.

6.1. Comparison of design features

There are many design differences between the old spectrometer and the new improved version that was completed in this work. The first is the difference in the way that the windows are set in the two different systems. The old system uses imprecise analog threshold comparisons and the new design employs accurate digital comparisons. The second major difference is the restriction on the usable detector pairs in the old design that does not exist in the new system. The restriction in the old system results from the fact that there is only one timing mechanism, whereas the new system employs a single unit that can time four channels (out of the eight total TDC channels). The next major architectural difference is the limit of two windows that exists in the old system, which is not present in the new system. The new system is theoretically restricted to seven windows (based on the three bits of information obtained from the WINDOW RAM - one bit pattern is reserved for encoding a "no window" result), but is realistically limited to the present three windows due to the enormous amount of data that results from each "extra" window.

There is also a comparison to be made in the amount of data that is collected by each system. There is a curse and blessing in the fact that the new system gathers much more information than the old system. On one hand, more data means more information, but on the other hand it takes more time to gather it and more room is necessary to store it. The extra window is also a win-lose situation for the same reason. The flexibility associated with gathering more data suggests that even though there are some downfalls, the more data gathered the better. It remains to be seen if the flexibility of the new system can be used to perform some experiment that no other system in the world is capable of, but all indications are that it will be able to do just that.

6.2. Future considerations

The future looks very bright for the new spectrometer, since it has been proven to be such an enormous improvement over the old system. There is one area that needs to be improved upon if the system is to perform up to the capabilities that were designed into the overall system. That area has clearly shown itself to be the main component of the timing channel, the TDC. It was known at the time of incorporation that the TDC was not guaranteed to perform up to what the manufacturer claimed, based on some earlier functional testing that was performed on the device. The specifications also showed it to be a slow device (the conversion time is 100us), but the fast clear function seemed to be a possible savior. Knowing that most of the events that the spectrometer would be analyzing would be singles (starts with no stops), the hope was that the dead-time of the system would not receive a significant contribution from the slow conversion time of the TDC.

The overall result was that the assumptions seemed to be correct. The incredible speed (3μs) at which the energy analysis is performed allows the fast clear input to be utilized in the TDC. If the event is invalid at the end of the maximum time that a valid event could be detected, then the system generates a clear and the conversion time of the TDC does not have to be realized. If the event is valid, then the slowness of the computer overshadows the long conversion time of the TDC, since it takes longer for the computer to request the data than it does for the TDC to provide its conversion result.

Obviously, the best future plan would be to replace the TDC with a timing device that is faster, but functionally equivalent. In order to get the most benefit out of that improvement, a faster computer would have to accompany this change, since a faster TDC would tend to elucidate the slowness of the computer. A faster TDC would require the placement of a wait loop in the interrupt service routine, since the TDC in the current system is ready to provide data at the same time the computer is ready to receive it. Replacement of the computer alone may actually be a good idea for the time being, since there is a need for speed in the interrupt service routine after the TDC data has been read.

Any increase in the execution speed of the service routine would provide an increase in the system performance as a whole. There is also a need for bigger capacity floppy and hard disk drives, since a data file for the new spectrometer averages about 450 kbytes. This need could be taken care of by the purchase of the faster computer, so the benefits of this change would be two-fold.

As far as the true test is concerned, the spectrometer needs to be used over a period of time in order to uncover any bugs that may exist. At this point in time there are no known bugs in the design, but in all likelihood there are some hidden problems. The pure complexity of the system makes it improbable that it is glitch-free. One mystery that has arisen is a problem in the data in very close proximity to T_0. It is not known at this time what is the cause, but when an attempt to fit the experimental data in this region the quality of the fit is diminished considerably. There is reason to believe that there is some cross-talk in the TDC that causes it to "hiccup" when two stops occur at the same time in the region of T_0. One fix for this may be to have the self-stop channels "staggered" in time in the TDC instead of having them all tuned to a count of about 20. This would alleviate the possibility that there is some confusion instilled in the TDC as a result of the close proximity of the self-stop channels. The T_0 for each of the sectors could be calibrated and stored in the same manner as it is currently, so the sector-to-sector (relative) shift would not be a problem.

In Appendix A there is a list of most of the known PAC probes. By close inspection, it was noticed that there is one probe that has a start gamma ray and two different stop gamma rays ($^{131}Te^{m}$). If this probe were used in the new system, it would be one of the first times that two different cascades were recorded at the same time (successfully). There is also a probe that has two start gamma rays and one stop gamma ray (^{99}Rh). It is improbable that anyone has ever performed an experiment using that sort of probe, thus it would be an excellent demonstration of the capabilities of the new system. It is the opinion of the author that there are other candidates for this type of probe and that they have not been discovered, because they have not been pursued.

References

1. D. Kurt Gaskill, Ph. D. Thesis, Oregon State University (1984), unpublished.

2. Herbert Jaeger, Ph. D. Thesis, Oregon State University (1987), unpublished.

3. Heinz D. Fuchs, M. S. Thesis, Oregon State University (1990), unpublished.

4. H. T. Su, Ph. D. Thesis, Oregon State University (1989), unpublished.

5. Georg Weidlich, M. S. Thesis, Oregon State University (1989), unpublished.

6. Rainer Schwenker, M. S. Thesis, Oregon State University (1990), unpublished.

7. Ruiping Wang, Ph. D. Thesis, Oregon State University (1991), unpublished.

8. A. Lerf and T. Buttz, Hyperfine Interactions **36**, 275 (1987).

9. R. Vianden, Hyperfine Interactions **15/16**, 1081 (1983).

10. H. H. Rinneberg, Atomic Energy Review, **17**, 477 (1979).

11. J. D. Jackson, *Classical Electrodynamics*, 2nd edition, John Wiley & Sons, New York (1975), Chapter 16.

12. H. Frauenfelder and R. M. Steffen, *Alpha- Beta- and Gamma-Ray Spectroscopy*, Vol. 2, edited by K. Siegbahn, North-Holland, Amsterdam (1965), Chapter XIXA, pp 997-1198.

13. D. Kurt Gaskill, Ph. D. Thesis, Oregon State University (1984), unpublished.

14. Herbert Jaeger, Ph. D. Thesis, Oregon State University (1987), unpublished.

15. H. T. Su, Ph. D. Thesis, Oregon State University (1989), unpublished.

16. Georg Weidlich, M. S. Thesis, Oregon State University (1989), unpublished.

17. Heinz D. Fuchs, M. S. Thesis, Oregon State University (1990), unpublished.

18. Rainer Schwenker, M. S. Thesis, Oregon State University (1990), unpublished.

19. Ruiping Wang, Ph. D. Thesis, Oregon State University (1991), unpublished.

20. Herbert Jaeger, Ph. D. Thesis, Oregon State University (1987), unpublished.

21. H. T. Su, Ph. D. Thesis, Oregon State University (1989), unpublished.

22. H Jaeger, J. A. Gardner, H. T. Su and R.L. Rasera, Rev. Sci. Instrum. **58**, 1694 (1987).

23. H. T. Su, Ph. D. Thesis, Oregon State University (1989), unpublished, p. 75.

24. A. R. Arends, C. Hohennemser, F. Pleiter, H. De Waard, L. Chow, and R. M. Sutter, Hyperfine Interactions **8**, 191 (1980).

25. Horowitz and Hill, *The Art of Electronics*, 2nd edition, Cambridge University Press, New York (1989), p. 1035.

26. J. W. Ball and M. Kaplan, J. Chem. Phys. **69**, 117 (1978)

27. W. B. Blumenthal, *The Chemical Behavior of Zirconium*, Van Nostrand, Princeton (1958).

28. *Table of Isotopes*, 7th edition, edited by C. M. Lederer and V. S. Shirley, Wiley-Interscience, New York (1978), p. 521.

29. Y. A. Ellis, Nuclear Data Sheets **9**, 319 (1972).

30. Y. Yeshurun and B. Arad, J. Phys. C., **7**, 430 (1973).

31. P. R. Gardner and W. V. Prestwich, Canadian Journal of Physics, **48**, 1430 (1970).

32. H. H. Rinneberg, Atomic Energy Review, **17**, 477 (1979).

Appendices

Appendix A. Some of the known PAC probes[32]

PARENT/ HALF-LIFE	DECAY MODE	DAUGHTER/ HALF-LIFE(ns)	START ENERGY(keV)	STOP ENERGY(keV)
^{44}Ti/47a	EC	^{44}Sc/155.8	78.4	67.8
^{99}Mo/66h	β^-	^{99}Tc/3.6	739.4	181.1
^{99}Rh/15d	EC,β^+	^{99}Ru/20.7	528.2 353.0	89.8 89.8
^{100}Pd/3.6d	EC	^{100}Rh/214.5	84.0	74.8
^{111}Ag/7.5d	β^-	^{111}Cd/84.1	96.3	245.4
^{111}Cd^m/48.6m	IT	^{111}Cd/84.1	150.6	245.4
^{111}In/2.8d	EC	^{111}Cd/84.1	171.3	245.4
^{115}Cd/53.4h	β^-	^{115}In/6.0	35.6	492.2
^{116}Sb^m/60.4m	EC,β^+	^{116}Sn/370	542.9	1072.4
^{117}Cd/2.4h	β^-	^{117}In/58.7	89.7	344.5
^{118}Sb^m/5.0h	EC,β^+	^{118}Sn/21.7	253.7	1091.5
^{131}Te^m/30h	β^-	^{131}I/5.9	102.1 102.1	200.6 240.9
^{133}Ba/10.7a	EC	^{133}Cs/6.4	356.0	81.0
^{133}Ce/5.4h	EC,β^+	^{133}La/64	510.4	58.4
^{140}La/40.3h	β^-	^{140}Ce/3.4	328.8	487.0
^{155}Tb/5.3d	EC	^{155}Gd/6.4	180.1	86.5
^{172}Lu/6.7d	EC	^{172}Yb/7.8	90.6	1093.6
^{177}Yb/1.9h	β^-	^{177}Lu/120	138.5 1080.1	150.4 150.4
^{181}Hf/42.4d	β^-	^{181}Ta/10.8	133.0	482.0
^{187}W/23.9h	β^-	^{187}Re/563	479.5	72.0
^{199}Hg^m/42.6m	IT	^{199}Hg/2.45	374	158.4
^{204}Pb^m/66.9m	IT	^{204}Pb/270	911.7	374.7
^{204}Bi/11.2h	EC	^{204}Pb/270	984.0	374.7

Appendix B. List of the EPROM-based functions used to control the VME-based boards

The following pages contain the actual contents of the EPROMs on the Brain and Energy boards. They are programmed in such a way that there is no bus contention when a function is selected. The procedure for communicating with the VME card cage is to send the NULL FUNCTION, followed by the new FUNCTION. In this way, when a new FUNCTION is selected, the NULL FUNCTION is always the last FUNCTION that the system has seen. Therefore there is **absolutely** no chance that there could be bus contention when changing functions. It should be noted that the EPROM used for storing these FUNCTIONs has 8 address lines, which means that there is a total of 256 different FUNCTIONs that can be programmed into the system. Since there are only 28 FUNCTIONS currently programmed, there are 228 FUNCTIONs that can still be programmed. This flexibility allows for any future changes in the use of the system.

B.1. Brain board EPROM #1 (top left-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	0	0	0	0	1	0	1	0	0A
WRITE SET-UP (VALID RAM)	01	0	0	0	0	0	1	0	1	05
WRITE (VALID RAM)	02	0	0	0	0	0	1	0	1	05
READ (VALID RAM)	03	0	0	0	0	0	1	0	0	04
WRITE SET-UP (WINDOW RAM 0)	04	0	0	0	0	0	1	0	1	05
WRITE (WINDOW RAM 0)	05	0	0	0	0	0	1	0	1	05
READ (WINDOW RAM 0)	06	0	0	0	0	0	1	0	0	04
WRITE SET-UP (WINDOW RAM 1)	07	0	0	0	0	0	1	0	1	05
WRITE (WINDOW RAM 1)	08	0	0	0	0	0	1	0	1	05
READ (WINDOW RAM 1)	09	0	0	0	0	0	1	0	0	04
WRITE SET-UP (WINDOW RAM 2)	0A	0	0	0	0	0	1	0	1	05
WRITE (WINDOW RAM 2)	0B	0	0	0	0	0	1	0	1	05
READ (WINDOW RAM 2)	0C	0	0	0	0	0	1	0	0	04
WRITE SET-UP (WINDOW RAM 3)	0D	0	0	0	0	0	1	0	1	05
WRITE (WINDOW RAM 3)	0E	0	0	0	0	0	1	0	1	05
READ (WINDOW RAM 3)	0F	0	0	0	0	0	1	0	0	04
RUN MODE	10	0	0	0	0	1	0	0	0	08
ADC-RAM CHECK 0	20	0	0	0	0	0	0	0	0	00
ADC-RAM CHECK 1	21	0	0	0	0	0	0	0	0	00
ADC-RAM CHECK 2	22	0	0	0	0	0	0	0	0	00
ADC-RAM CHECK 3	23	0	0	0	0	0	0	0	0	00
CALIBRATE 0	80	0	0	0	0	0	0	1	0	02
CALIBRATE 1	A0	0	0	0	0	0	0	1	0	02
CALIBRATE 2	C0	0	0	0	0	0	0	1	0	02
CALIBRATE 3	E0	0	0	0	0	0	0	1	0	02

D7 = NO CONNECTION

D6 = NO CONNECTION

D5 = NO CONNECTION

D4 = NO CONNECTION

D3 = G of 74LS645 CONTROLLING ADDRESS-IN FROM COMPUTER (0 ≡ ENABLE)

D2 = DIR of 74LS645 CONTROLLING ADDRESS-IN FROM COMPUTER (0 ≡ TO COMPUTER)

D1 = G of 74LS645 CONTROLLING DATA-IN FROM COMPUTER (0 ≡ ENABLE)

D0 = DIR of 74LS645 CONTROLLING DATA-IN FROM COMPUTER (0 ≡ TO COMPUTER)

B.2. Brain board EPROM #2 (center of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	0	1	1	0	1	0	1	1	6B
WRITE SET-UP (VALID RAM)	01	0	0	0	0	1	1	1	0	0E
WRITE (VALID RAM)	02	0	0	0	0	0	1	1	0	06
READ (VALID RAM)	03	1	0	0	0	1	1	0	0	8C
WRITE SET-UP (WINDOW RAM 0)	04	0	1	1	0	1	0	1	1	6B
WRITE (WINDOW RAM 0)	05	0	1	1	0	1	0	1	1	6B
READ (WINDOW RAM 0)	06	0	1	1	0	1	0	1	1	6B
WRITE SET-UP (WINDOW RAM 1)	07	0	1	1	0	1	0	1	1	6B
WRITE (WINDOW RAM 1)	08	0	1	1	0	1	0	1	1	6B
READ (WINDOW RAM 1)	09	0	1	1	0	1	0	1	1	6B
WRITE SET-UP (WINDOW RAM 2)	0A	0	1	1	0	1	0	1	1	6B
WRITE (WINDOW RAM 2)	0B	0	1	1	0	1	0	1	1	6B
READ (WINDOW RAM 2)	0C	0	1	1	0	1	0	1	1	6B
WRITE SET-UP (WINDOW RAM 3)	0D	0	1	1	0	1	0	1	1	6B
WRITE (WINDOW RAM 3)	0E	0	1	1	0	1	0	1	1	6B
READ (WINDOW RAM 3)	0F	0	1	1	0	1	0	1	1	6B
RUN MODE	10	1	0	0	0	1	1	0	0	8C
ADC-RAM CHECK 0	20	0	1	1	0	1	0	1	1	6B
ADC-RAM CHECK 1	21	0	1	1	0	1	0	1	1	6B
ADC-RAM CHECK 2	22	0	1	1	0	1	0	1	1	6B
ADC-RAM CHECK 3	23	0	1	1	0	1	0	1	1	6B
CALIBRATE 0	80	0	1	1	0	1	0	1	1	6B
CALIBRATE 1	A0	0	1	1	0	1	0	1	1	6B
CALIBRATE 2	C0	0	1	1	0	1	0	1	1	6B
CALIBRATE 3	E0	0	1	1	0	1	0	1	1	6B

D7 = DIR of 74LS645 CONTROLLING DATA of VALID RAM (0 ≡ INTO RAM)

D6 = G of 74LS645 CONTROLLING DATA of VALID RAM (0 ≡ ENABLE)

D5 = G of 74LS645 CONTROLLING ADDRESS of VALID RAM (0 ≡ ENABLE)

D4 = DIR of 74LS645 CONTROLLING ADDRESS of VALID RAM (0 ≡ INTO RAM)

D3 = WE of VALID RAM (0 ≡ WRITE ENABLED)

D2 = CS2 of VALID RAM (1 ≡ SELECTED)

D1 = OE of VALID RAM (0 ≡ OUTPUT ENABLED)

D0 = CS1 of VALID RAM (0 ≡ SELECTED)

B.3. Energy board 0 - EPROM #1 (left-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (VALID RAM)	01	1	1	1	0	1	0	1	1	EB
WRITE (VALID RAM)	02	1	1	1	0	1	0	1	1	EB
READ (VALID RAM)	03	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	0	1	1	1	0	EE
WRITE (WINDOW RAM 0)	05	1	1	1	0	0	1	1	0	E6
READ (WINDOW RAM 0)	06	1	1	1	0	1	1	0	0	EC
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 1)	08	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 1)	09	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 2)	0B	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 2)	0C	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 3)	0E	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 3)	0F	1	1	1	0	1	0	1	1	EB
RUN MODE	10	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 0	20	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 1	21	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 2	22	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 3	23	1	1	1	0	1	0	1	1	EB
CALIBRATE 0	80	1	1	0	0	1	0	1	1	CB
CALIBRATE 1	A0	1	1	1	0	1	0	1	1	EB
CALIBRATE 2	C0	1	1	1	0	1	0	1	1	EB
CALIBRATE 3	E0	1	1	1	0	1	0	1	1	EB

D7 = INPUT "AND" GATE for RESET OF SECOND START ONE-SHOT (0 ≡ RESET)

D6 = INPUT "AND" GATE for RESET OF FIRST START ONE-SHOT (0 ≡ RESET)

D5 = CS OF ADC (0 ≡ ENABLED)

D4 = NO CONNECTION

D3 = WE OF WINDOW RAM (0 ≡ WRITE ENABLED)

D2 = CS2 OF WINDOW RAM (1 ≡ SELECTED)

D1 = OE OF WINDOW RAM (0 ≡ OUTPUT ENABLED)

D0 = CS1 OF WINDOW RAM (0 ≡ SELECTED)

B.4. Energy board 0 - EPROM #2 (right-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (VALID RAM)	01	1	1	1	1	1	1	0	0	FC
WRITE (VALID RAM)	02	1	1	1	1	1	1	0	0	FC
READ (VALID RAM)	03	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 0)	04	0	0	1	1	1	1	0	0	3C
WRITE (WINDOW RAM 0)	05	0	0	1	1	1	1	0	0	3C
READ (WINDOW RAM 0)	06	0	0	1	1	1	1	0	1	3D
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 1)	08	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 1)	09	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 2)	0B	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 2)	0C	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 3)	0E	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 3)	0F	1	1	1	1	1	1	0	0	FC
RUN MODE	10	1	1	1	1	0	0	0	0	F0
ADC-RAM CHECK 0	20	0	1	1	1	0	0	0	1	71
ADC-RAM CHECK 1	21	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 2	22	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 3	23	1	1	1	1	1	1	0	0	FC
CALIBRATE 0	80	1	0	1	1	1	1	1	0	BE
CALIBRATE 1	A0	1	1	1	1	1	1	0	0	FC
CALIBRATE 2	C0	1	1	1	1	1	1	0	0	FC
CALIBRATE 3	E0	1	1	1	1	1	1	0	0	FC

D7 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D6 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D5 = INPUT "NAND" GATE for LATCH CLEAR (0 ≡ CLEAR)

D4 = INPUT "NAND" GATE for RESET of LATCH CLOCK ONE-SHOT (0 ≡ RESET)

D3 = G1 & G2 of 74LS173 GAMMA LATCH (0 ≡ DATA ENABLED)

D2 = M & N of 74LS173 GAMMA LATCH (0 ≡ OUTPUT ENABLED)

D1 = DIR of 74LS645 CONTROLLING ADDRESS of WINDOW RAM (0 ≡ INTO RAM)

D0 = DIR of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ INTO RAM)

B.5. Energy board 1 - EPROM #1 (left-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (VALID RAM)	01	1	1	1	0	1	0	1	1	EB
WRITE (VALID RAM)	02	1	1	1	0	1	0	1	1	EB
READ (VALID RAM)	03	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 0)	05	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 0)	06	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	0	1	1	1	0	EE
WRITE (WINDOW RAM 1)	08	1	1	1	0	0	1	1	0	E6
READ (WINDOW RAM 1)	09	1	1	1	0	1	1	0	0	EC
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 2)	0B	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 2)	0C	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 3)	0E	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 3)	0F	1	1	1	0	1	0	1	1	EB
RUN MODE	10	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 0	20	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 1	21	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 2	22	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 3	23	1	1	1	0	1	0	1	1	EB
CALIBRATE 0	80	1	1	1	0	1	0	1	1	EB
CALIBRATE 1	A0	1	1	0	0	1	0	1	1	CB
CALIBRATE 2	C0	1	1	1	0	1	0	1	1	EB
CALIBRATE 3	E0	1	1	1	0	1	0	1	1	EB

D7 = INPUT "AND" GATE for RESET OF SECOND START ONE-SHOT (0 ≡ RESET)

D6 = INPUT "AND" GATE for RESET OF FIRST START ONE-SHOT (0 ≡ RESET)

D5 = CS OF ADC (0 ≡ ENABLED)

D4 = NO CONNECTION

D3 = WE OF WINDOW RAM (0 ≡ WRITE ENABLED)

D2 = CS2 OF WINDOW RAM (1 ≡ SELECTED)

D1 = OE OF WINDOW RAM (0 ≡ OUTPUT ENABLED)

D0 = CS1 OF WINDOW RAM (0 ≡ SELECTED)

B.6. Energy board 1 - EPROM #2 (right-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (VALID RAM)	01	1	1	1	1	1	1	0	0	FC
WRITE (VALID RAM)	02	1	1	1	1	1	1	0	0	FC
READ (VALID RAM)	03	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 0)	05	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 0)	06	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 1)	07	0	0	1	1	1	1	0	0	3C
WRITE (WINDOW RAM 1)	08	0	0	1	1	1	1	0	0	3C
READ (WINDOW RAM 1)	09	0	0	1	1	1	1	0	1	3D
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 2)	0B	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 2)	0C	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 3)	0E	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 3)	0F	1	1	1	1	1	1	0	0	FC
RUN MODE	10	1	1	1	1	0	0	0	0	F0
ADC-RAM CHECK 0	20	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 1	21	0	1	1	1	0	0	0	1	71
ADC-RAM CHECK 2	22	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 3	23	1	1	1	1	1	1	0	0	FC
CALIBRATE 0	80	1	1	1	1	1	1	0	0	FC
CALIBRATE 1	A0	1	0	1	1	1	1	1	0	BE
CALIBRATE 2	C0	1	1	1	1	1	1	0	0	FC
CALIBRATE 3	E0	1	1	1	1	1	1	0	0	FC

D7 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D6 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D5 = INPUT "NAND" GATE for LATCH CLEAR (0 ≡ CLEAR)

D4 = INPUT "NAND" GATE for RESET of LATCH CLOCK ONE-SHOT (0 ≡ RESET)

D3 = G1 & G2 of 74LS173 GAMMA LATCH (0 ≡ DATA ENABLED)

D2 = M & N of 74LS173 GAMMA LATCH (0 ≡ OUTPUT ENABLED)

D1 = DIR of 74LS645 CONTROLLING ADDRESS of WINDOW RAM (0 ≡ INTO RAM)

D0 = DIR of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ INTO RAM)

B.7. Energy board 2 - EPROM #1 (left-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (VALID RAM)	01	1	1	1	0	1	0	1	1	EB
WRITE (VALID RAM)	02	1	1	1	0	1	0	1	1	EB
READ (VALID RAM)	03	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 0)	05	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 0)	06	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 1)	08	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 1)	09	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	0	1	1	1	0	EE
WRITE (WINDOW RAM 2)	0B	1	1	1	0	0	1	1	0	E6
READ (WINDOW RAM 2)	0C	1	1	1	0	1	1	0	0	EC
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 3)	0E	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 3)	0F	1	1	1	0	1	0	1	1	EB
RUN MODE	10	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 0	20	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 1	21	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 2	22	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 3	23	1	1	1	0	1	0	1	1	EB
CALIBRATE 0	80	1	1	1	0	1	0	1	1	EB
CALIBRATE 1	A0	1	1	1	0	1	0	1	1	EB
CALIBRATE 2	C0	1	1	0	0	1	0	1	1	CB
CALIBRATE 3	E0	1	1	1	0	1	0	1	0	EB

D7 = INPUT "AND" GATE for RESET OF SECOND START ONE-SHOT (0 ≡ RESET)

D6 = INPUT "AND" GATE for RESET OF FIRST START ONE-SHOT (0 ≡ RESET)

D5 = CS OF ADC (0 ≡ ENABLED)

D4 = NO CONNECTION

D3 = WE OF WINDOW RAM (0 ≡ WRITE ENABLED)

D2 = CS2 OF WINDOW RAM (1 ≡ SELECTED)

D1 = OE OF WINDOW RAM (0 ≡ OUTPUT ENABLED)

D0 = CS1 OF WINDOW RAM (0 ≡ SELECTED)

B.8. Energy board 2 - EPROM #2 (right-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (VALID RAM)	01	1	1	1	1	1	1	0	0	FC
WRITE (VALID RAM)	02	1	1	1	1	1	1	0	0	FC
READ (VALID RAM)	03	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 0)	05	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 0)	06	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 1)	08	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 1)	09	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 2)	0A	0	0	1	1	1	1	0	0	3C
WRITE (WINDOW RAM 2)	0B	0	0	1	1	1	1	0	0	3C
READ (WINDOW RAM 2)	0C	0	0	1	1	1	1	0	1	3D
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 3)	0E	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 3)	0F	1	1	1	1	1	1	0	0	FC
RUN MODE	10	1	1	1	1	0	0	0	0	F0
ADC-RAM CHECK 0	20	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 1	21	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 2	22	0	1	1	1	0	0	0	1	71
ADC-RAM CHECK 3	23	1	1	1	1	1	1	0	0	FC
CALIBRATE 0	80	1	1	1	1	1	1	0	0	FC
CALIBRATE 1	A0	1	1	1	1	1	1	0	0	FC
CALIBRATE 2	C0	1	0	1	1	1	1	1	0	BE
CALIBRATE 3	E0	1	1	1	1	1	1	0	0	FC

D7 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D6 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D5 = INPUT "NAND" GATE for LATCH CLEAR (0 ≡ CLEAR)

D4 = INPUT "NAND" GATE for RESET of LATCH CLOCK ONE-SHOT (0 ≡ RESET)

D3 = G1 & G2 of 74LS173 GAMMA LATCH (0 ≡ DATA ENABLED)

D2 = M & N of 74LS173 GAMMA LATCH (0 ≡ OUTPUT ENABLED)

D1 = DIR of 74LS645 CONTROLLING ADDRESS of WINDOW RAM (0 ≡ INTO RAM)

D0 = DIR of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ INTO RAM)

B.9. Energy board 3 - EPROM #1 (left-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (VALID RAM)	01	1	1	1	0	1	0	1	1	EB
WRITE (VALID RAM)	02	1	1	1	0	1	0	1	1	EB
READ (VALID RAM)	03	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 0)	05	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 0)	06	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 1)	08	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 1)	09	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	0	1	0	1	1	EB
WRITE (WINDOW RAM 2)	0B	1	1	1	0	1	0	1	1	EB
READ (WINDOW RAM 2)	0C	1	1	1	0	1	0	1	1	EB
WRITE SET-UP (WINDOW RAM 3)	0D	1	1	1	0	1	1	1	0	EE
WRITE (WINDOW RAM 3)	0E	1	1	1	0	0	1	1	0	E6
READ (WINDOW RAM 3)	0F	1	1	1	0	1	1	0	0	EC
RUN MODE	10	1	1	0	0	1	1	0	0	CC
ADC-RAM CHECK 0	20	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 1	21	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 2	22	1	1	1	0	1	0	1	1	EB
ADC-RAM CHECK 3	23	1	1	0	0	1	1	0	0	CC
CALIBRATE 0	80	1	1	1	0	1	0	1	1	EB
CALIBRATE 1	A0	1	1	1	0	1	0	1	1	EB
CALIBRATE 2	C0	1	1	1	0	1	0	1	1	EB
CALIBRATE 3	E0	1	1	0	0	1	0	1	1	CB

D7 = INPUT "AND" GATE for RESET OF SECOND START ONE-SHOT (0 ≡ RESET)

D6 = INPUT "AND" GATE for RESET OF FIRST START ONE-SHOT (0 ≡ RESET)

D5 = CS OF ADC (0 ≡ ENABLED)

D4 = NO CONNECTION

D3 = WE OF WINDOW RAM (0 ≡ WRITE ENABLED)

D2 = CS2 OF WINDOW RAM (1 ≡ SELECTED)

D1 = OE OF WINDOW RAM (0 ≡ OUTPUT ENABLED)

D0 = CS1 OF WINDOW RAM (0 ≡ SELECTED)

B.10. Energy board 3 - EPROM #2 (right-hand side of board)

FUNCTION	ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0	HEX
NULL FUNCTION (ISOLATION)	00	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (VALID RAM)	01	1	1	1	1	1	1	0	0	FC
WRITE (VALID RAM)	02	1	1	1	1	1	1	0	0	FC
READ (VALID RAM)	03	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 0)	04	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 0)	05	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 0)	06	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 1)	07	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 1)	08	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 1)	09	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 2)	0A	1	1	1	1	1	1	0	0	FC
WRITE (WINDOW RAM 2)	0B	1	1	1	1	1	1	0	0	FC
READ (WINDOW RAM 2)	0C	1	1	1	1	1	1	0	0	FC
WRITE SET-UP (WINDOW RAM 3)	0D	0	0	1	1	1	1	0	0	3C
WRITE (WINDOW RAM 3)	0E	0	0	1	1	1	1	0	0	3C
READ (WINDOW RAM 3)	0F	0	0	1	1	1	1	0	1	3D
RUN MODE	10	1	1	1	1	0	0	0	0	F0
ADC-RAM CHECK 0	20	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 1	21	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 2	22	1	1	1	1	1	1	0	0	FC
ADC-RAM CHECK 3	23	0	1	1	1	0	0	0	1	71
CALIBRATE 0	80	1	1	1	1	1	1	0	0	FC
CALIBRATE 1	A0	1	1	1	1	1	1	0	0	FC
CALIBRATE 2	C0	1	1	1	1	1	1	0	0	FC
CALIBRATE 3	E0	1	0	1	1	1	1	1	0	BE

D7 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D6 = G of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ ENABLE)

D5 = INPUT "NAND" GATE for LATCH CLEAR (0 ≡ CLEAR)

D4 = INPUT "NAND" GATE for RESET of LATCH CLOCK ONE-SHOT (0 ≡ RESET)

D3 = G1 & G2 of 74LS173 GAMMA LATCH (0 ≡ DATA ENABLED)

D2 = M & N of 74LS173 GAMMA LATCH (0 ≡ OUTPUT ENABLED)

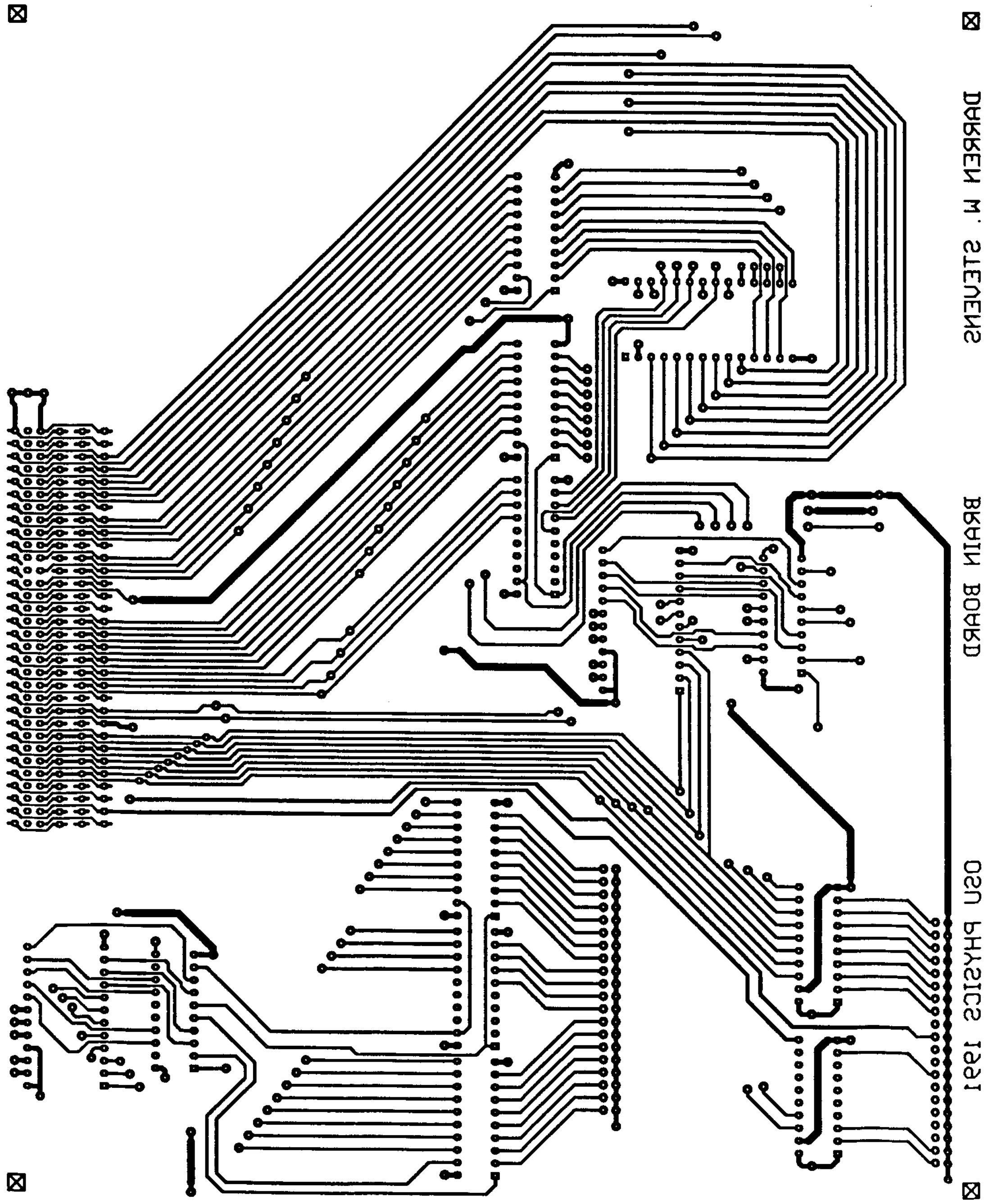
D1 = DIR of 74LS645 CONTROLLING ADDRESS of WINDOW RAM (0 ≡ INTO RAM)

D0 = DIR of 74LS645 CONTROLLING DATA of WINDOW RAM (0 ≡ INTO RAM)

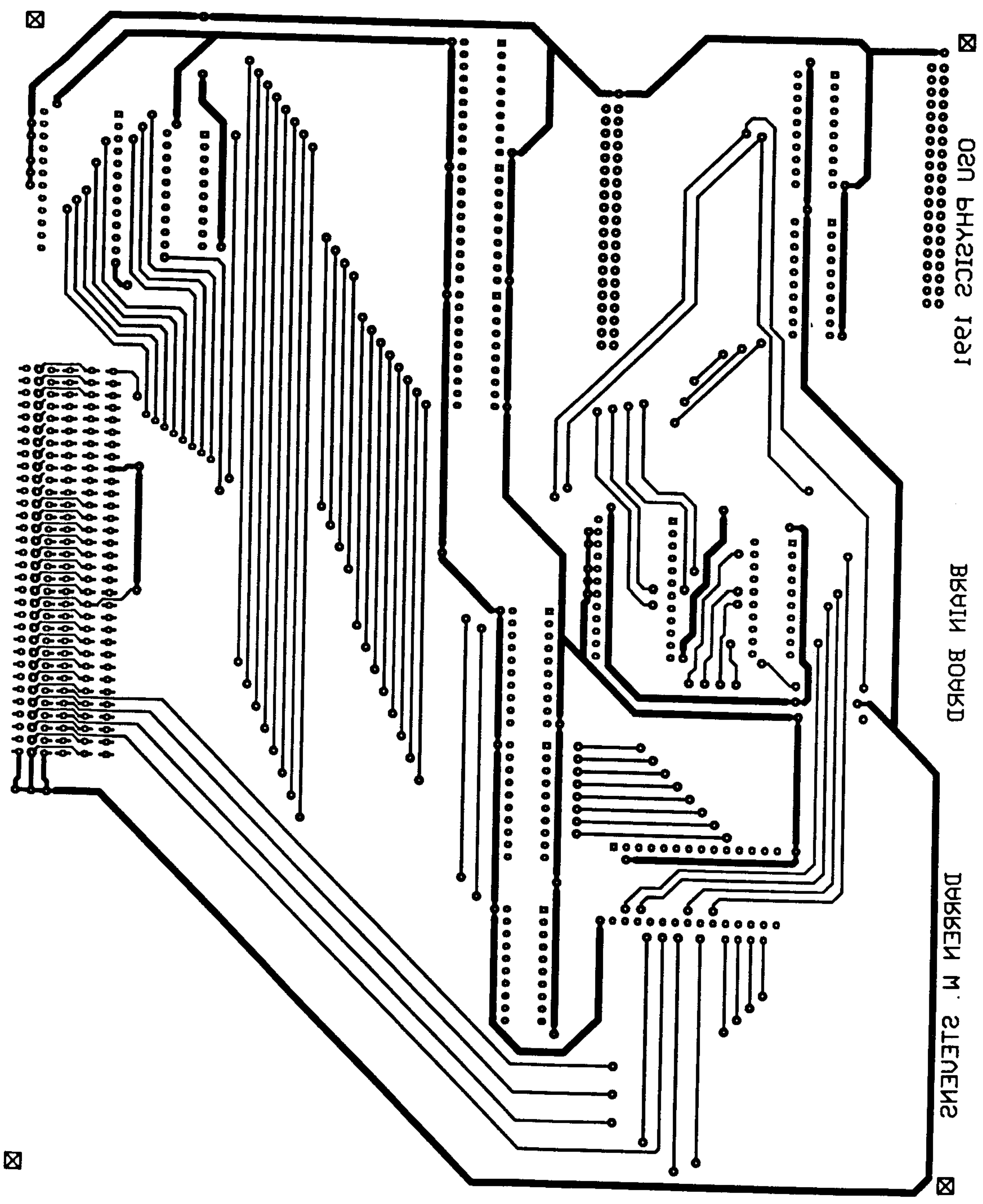
C. The PCB Artwork used in the etching process (reduced from full-size)

On the following pages is a copy of the actual artwork used in the photo etching process for the Brain and Energy boards. They are shown for the purpose of being complete and are being shown without any embellishment.

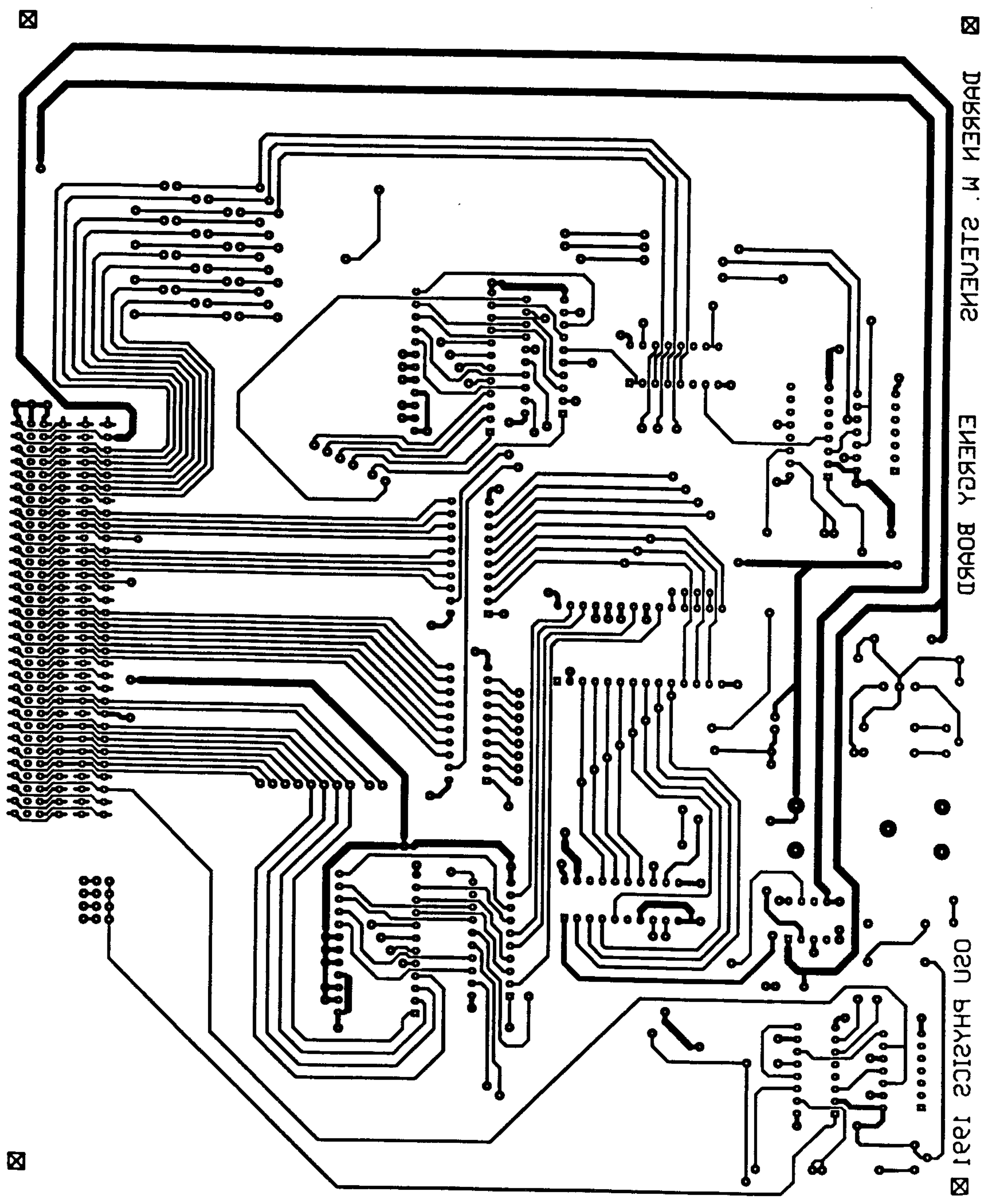
C.1. The solder side of the Brain board



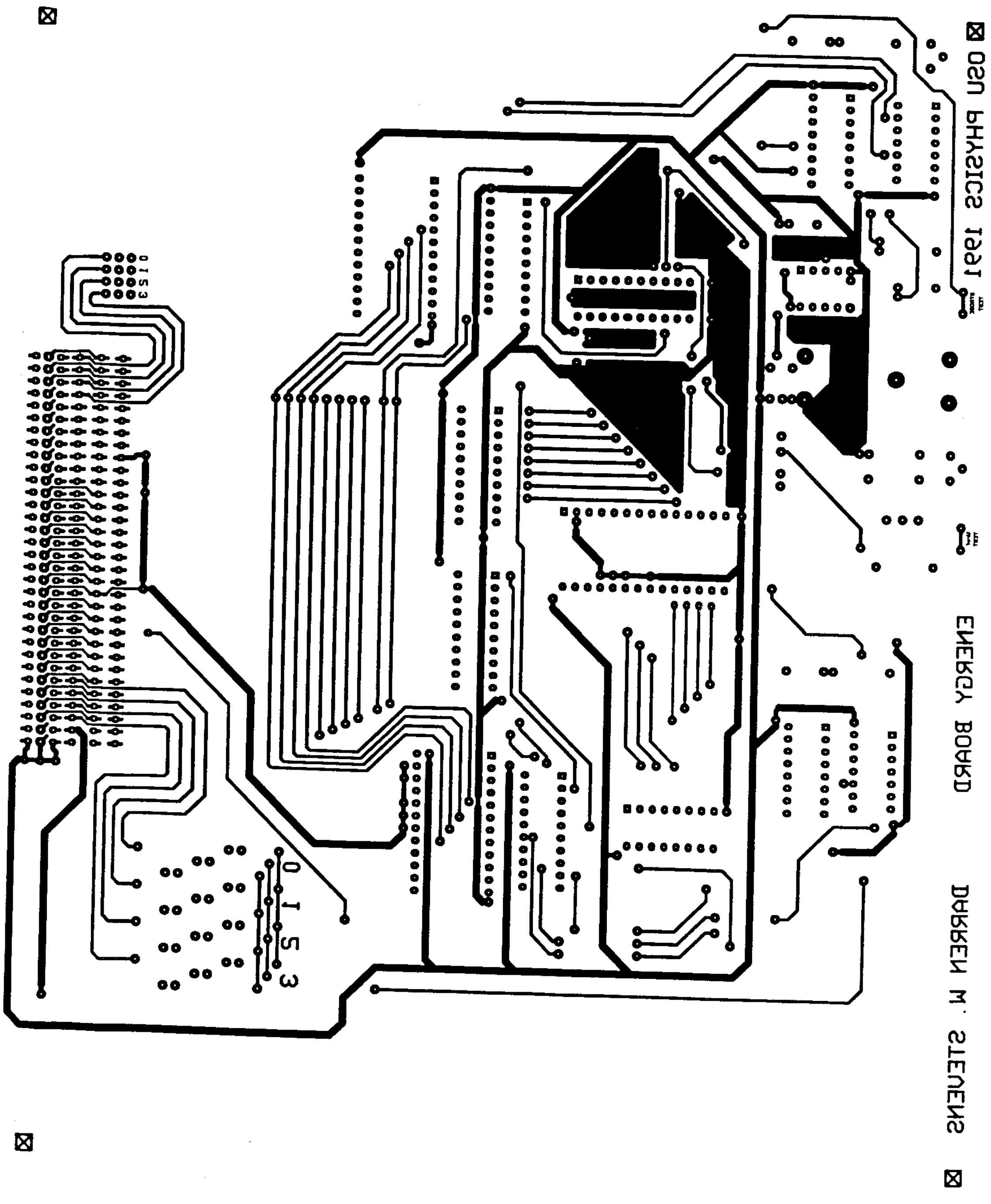
C.2. The component side of the Brain board



C.3. The solder side of the Energy boards



C.4. The component side of the Energy boards



D. Assembly language listing of the program VMEASS.ASM

In this appendix, the listing of the Assembly language program VMEASS.ASM can be found. This program performs the task of setting up the interrupt service routines and their associated data space. The listing of the actual interrupt service routines are contained within the structure of the main program, as the routines are implemented using a Terminate-and-Stay-Resident (TSR) method. For this reason, this program **must** be executed before any of the other programs. The listing are presented here without detailed explanation, but they can be understood the experienced programmer by reading the commented source code.

```
PAGE 60,80
;**********************************************************************
;*                                                                    *
;*  PAC SPECTROMETER INTERRUPT PROGRAM - VMEASS.ASM    2/9/92         *
;*        written by  DARREN STEVENS AND RANDY LUNDQUIST              *
;*                                                                    *
;**********************************************************************

;------------------------------------------------------------
;------------------------------------------------------------
.MODEL COMPACT
;------------------------------------------------------------
;------------------------------------------------------------
STACK SEGMENT PARA STACK 'STACK'

DB 200H DUP (?)                        ;200 (HEX) BYTE STACK DEFINITION

STACK ENDS

;------------------------------------------------------------
;------------------------------------------------------------
DATA SEGMENT BYTE PUBLIC 'DATA'

;--------------------------------------

H1_PORTA          EQU   0300H         ;
INTDATA           EQU   0300H         ;

H1_PORTB          EQU   0301H         ;HEADER 1 ADDRESSES FOR
FUNKTION          EQU   0301H         ;

H1_PORTC          EQU   0302H         ;CONTROL REGISTER AND PORTS
FUNINTSTUFF       EQU   0302H         ;

H1_CONTROL        EQU   0303H         ;
FUNCONTROL        EQU   0303H         ;

;--------------------------------------

H2_PORTA          EQU   0304H         ;
VMEDATA           EQU   0304H         ;

H2_PORTB          EQU   0305H         ;HEADER 2 ADDRESSES FOR
VMEADDLOW         EQU   0305H         ;

H2_PORTC          EQU   0306H         ;CONTROL REGISTER AND PORTS
VMEADDHIGH        EQU   0306H         ;

H2_CONTROL        EQU   0307H         ;
ADDATACONTROL     EQU   0307H         ;
```

```
;-------------------------------------

H3_PORTA          EQU   0308H     ;
TDCOUT            EQU   0308H     ;

H3_PORTB          EQU   0309H     ;HEADER 3 ADDRESSES FOR
TDCHIGHIN         EQU   0309H     ;

H3_PORTC          EQU   030AH     ;CONTROL REGISTER AND PORTS
TDCLOWIN          EQU   030AH     ;

H3_CONTROL        EQU   030BH     ;
TDCCONTROL        EQU   030BH     ;

;**************************************

PC0_RESET         EQU   00H       ;
FUNCLOCK_LOW      EQU   00H       ;PORT C - BIT #0 SET/RESET FUNCTIONS
PC0_SET           EQU   01H       ;SENT TO THE CONTROL REGISTERS OF 8255
FUNCLOCK_HIGH     EQU   01H       ;

PC1_RESET         EQU   02H       ;
FUNCTION_ENABLE   EQU   02H       ;PORT C - BIT #1 SET/RESET FUNCTIONS
PC1_SET           EQU   03H       ;SENT TO THE CONTROL REGISTERS OF 8255
FUNCTION_DISABLE  EQU   03H       ;

PC2_RESET         EQU   04H       ;
SOFTRESET_LOW     EQU   04H       ;PORT C - BIT #2 SET/RESET FUNCTIONS
PC2_SET           EQU   05H       ;SENT TO THE CONTROL REGISTERS OF 8255
SOFTRESET_HIGH    EQU   05H       ;

PC3_RESET         EQU   06H       ;PORT C - BIT #3 SET/RESET FUNCTIONS
PC3_SET           EQU   07H       ;SENT TO THE CONTROL REGISTERS OF 8255

PC4_RESET         EQU   08H       ;
INTERRUPT_OFF     EQU   08H       ;PORT C - BIT #4 SET/RESET FUNCTIONS
PC4_SET           EQU   09H       ;SENT TO THE CONTROL REGISTERS OF 8255
INTERRUPT_ON      EQU   09H       ;

;**************************************

FUN_MODE          EQU   0B8H      ;MODE CONTROL WORD - FUNCTION HEADER

TDC_MODE          EQU   8BH       ;MODE CONTROL WORD - TDC HEADER

READ_RAM          EQU   50H       ;MODE CONTROL WORDS
WRITE_RAM         EQU   80H       ;FOR DATA/ADDRESS HEADER
CALIBRATION       EQU   9BH       ;THESE SETTINGS DEPEND
RUN_MODE          EQU   9BH       ;ON VME BUS STATUS

;**************************************
```

```
TDC_OFF          EQU   0FFH           ;FUNCTION SENT TO TDC - TURNS BUS OFF

TDCREAD_0        EQU   0BFH           ;FUNCTION SENT TO TDC - READ CHANNEL 0
TDCREAD_1        EQU   0BEH           ;FUNCTION SENT TO TDC - READ CHANNEL 1
TDCREAD_2        EQU   0BDH           ;FUNCTION SENT TO TDC - READ CHANNEL 2
TDCREAD_3        EQU   0BCH           ;FUNCTION SENT TO TDC - READ CHANNEL 3
TDCREAD_4        EQU   0BBH           ;FUNCTION SENT TO TDC - READ CHANNEL 4
TDCREAD_5        EQU   0BAH           ;FUNCTION SENT TO TDC - READ CHANNEL 5
TDCREAD_6        EQU   0B9H           ;FUNCTION SENT TO TDC - READ CHANNEL 6
TDCREAD_7        EQU   0B8H           ;FUNCTION SENT TO TDC - READ CHANNEL 7

;*************************************

I8259_CONTROL    EQU   20H            ;PORT ADDRESS-8259 CONTROL REGISTER
I8259_IMASKR     EQU   21H            ;PORT ADDRESS-8259 IRQ MASK REGISTER

END_OF_INT       EQU   20H            ;END OF INTERRUPT SIGNAL

LOWINT_ENABLE    EQU   0F8H           ;BYTE ENABLES IRQ 0, 1 AND 2 (ON PC BUS)

IRQ3_INT         EQU   11             ;INT# = IRQ#+8 (WE USE IRQ3 FOR XT'S)

QB_INT           EQU   7H             ;INTERRUPT FOR QuickBASIC PROGRAMS

MCA_ARRAY        DB    400H DUP (0)   ;MCA DATA ARRAY
PLOT_SCALE       DW    1              ;PLOTTING SCALE
BIT              DW    0              ;PLOTTING BIT
BITE             DW    0              ;PLOTTING BYTE
BYTEBIT          DB    0              ;PLOTTING BIT IN BYTE
COUNTS           DW    0              ;COUNT VALUE
SCOUNT           DB    0              ;SCALED COUNT VALUE
ENERGY           DW    0              ;BOTH BYTES OF ENERGY

SEGTABLE         DW    18H DUP (0)    ;LOOK-UP TABLE FOR SECTOR SEGMENTS

START_TABLE      DB    18H DUP (0)    ;LOOK UP TABLE FOR START CHANNELS

STOP_TABLE       DB    18H DUP (0)    ;LOOK UP TABLE FOR STOP CHANNELS

TABLE_OFFSET     DW    (0)            ;WORD HOLDS OFFSET WITHIN THE TABLES

DATA ENDS                             ;END OF DATA SEGMENT

;-----------------------------------------------------------
;-----------------------------------------------------------
S011 SEGMENT PARA PUBLIC 'DATA'       ;DATA SEGMENT FOR SECTOR #0 - 011
  SECTOR011   DW 1000H DUP (0)        ;
S011 ENDS                             ;START 0 - STOP 1 (LOW ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
S021 SEGMENT PARA PUBLIC 'DATA'       ;DATA SEGMENT FOR SECTOR #1 - 021
```

```
  SECTOR021   DW 1000H DUP (0)           ;
S021 ENDS                                ;START 0 - STOP 2 (LOW ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S031 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #2 - 031
  SECTOR031   DW 1000H DUP (0)           ;
S031 ENDS                                ;START 0 - STOP 3 (LOW ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S012 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #3 - 012
  SECTOR012   DW 1000H DUP (0)           ;
S012 ENDS                                ;START 0 - STOP 1 (HIGH ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S022 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #4 - 022
  SECTOR022   DW 1000H DUP (0)           ;
S022 ENDS                                ;START 0 - STOP 2 (HIGH ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S032 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #5 - 032
  SECTOR032   DW 1000H DUP (0)           ;
S032 ENDS                                ;START 0 - STOP 3 (HIGH ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S101 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #6 - 101
  SECTOR101   DW 1000H DUP (0)           ;
S101 ENDS                                ;START 1 - STOP 0 (LOW ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S121 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #7 - 121
  SECTOR121   DW 1000H DUP (0)           ;
S121 ENDS                                ;START 1 - STOP 2 (LOW ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S131 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #8 - 131
  SECTOR131   DW 1000H DUP (0)           ;
S131 ENDS                                ;START 1 - STOP 3 (LOW ENERGY)

;------------------------------------------------------------
;------------------------------------------------------------
S102 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #9 - 102
  SECTOR102   DW 1000H DUP (0)           ;
S102 ENDS                                ;START 1 - STOP 0 (HIGH ENERGY)

;------------------------------------------------------------
```

```
;-------------------------------------------------------------
S122 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #10 - 122
  SECTOR122   DW 1000H DUP (0)           ;
S122 ENDS                                ;START 1 - STOP 2 (HIGH ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S132 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #11 - 132
  SECTOR132   DW 1000H DUP (0)           ;
S132 ENDS                                ;START 1 - STOP 3 (HIGH ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S201 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #12 - 201
  SECTOR201   DW 1000H DUP (0)           ;
S201 ENDS                                ;START 2 - STOP 0 (LOW ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S211 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #13 - 211
  SECTOR211   DW 1000H DUP (0)           ;
S211 ENDS                                ;START 2 - STOP 1 (LOW ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S231 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #14 - 231
  SECTOR231   DW 1000H DUP (0)           ;
S231 ENDS                                ;START 2 - STOP 3 (LOW ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S202 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #15 - 202
  SECTOR202   DW 1000H DUP (0)           ;
S202 ENDS                                ;START 2 - STOP 0 (HIGH ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S212 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #16 - 212
  SECTOR212   DW 1000H DUP (0)           ;
S212 ENDS                                ;START 2 - STOP 1 (HIGH ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S232 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #17 - 232
  SECTOR232   DW 1000H DUP (0)           ;
S232 ENDS                                ;START 2 - STOP 3 (HIGH ENERGY)

;-------------------------------------------------------------
;-------------------------------------------------------------
S301 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #18 - 301
  SECTOR301   DW 1000H DUP (0)           ;
S301 ENDS                                ;START 3 - STOP 0 (LOW ENERGY)
```

```
;-----------------------------------------------------------
;-----------------------------------------------------------
S311 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #19 - 311
  SECTOR311   DW 1000H DUP (0)           ;
S311 ENDS                                ;START 3 - STOP 1 (LOW ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
S321 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #20 - 321
  SECTOR321   DW 1000H DUP (0)           ;
S321 ENDS                                ;START 3 - STOP 2 (LOW ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
S302 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #21 - 302
  SECTOR302   DW 1000H DUP (0)           ;
S302 ENDS                                ;START 3 - STOP 0 (HIGH ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
S312 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #22 - 312
  SECTOR312   DW 1000H DUP (0)           ;
S312 ENDS                                ;START 3 - STOP 1 (HIGH ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
S322 SEGMENT PARA PUBLIC 'DATA'          ;DATA SEGMENT FOR SECTOR #23 - 322
  SECTOR322   DW 1000H DUP (0)           ;
S322 ENDS                                ;START 3 - STOP 2 (HIGH ENERGY)

;-----------------------------------------------------------
;-----------------------------------------------------------
CODE SEGMENT PARA PUBLIC 'CODE'          ;THE CODE SEGMENT (EXECUTABLE STUFF)

;-------------------------------------

ASSUME CS:CODE,DS:DATA,ES:DATA,SS:STACK        ;ASSEMBLER DIRECTIVES

;**************************************************************
;-----------------------------------------------------------
PAC_INT  PROC  NEAR                      ;THE HARDWARE (RUN) INTERRUPT ROUTINE

   PUSH  AX                              ;SAVE AX REGISTER
   PUSH  BX                              ;SAVE BX REGISTER
   PUSH  CX                              ;SAVE CX REGISTER
   PUSH  DX                              ;SAVE DX REGISTER
   PUSH  DS                              ;SAVE DS REGISTER
   PUSH  ES                              ;SAVE ES REGISTER
   PUSH  DI                              ;SAVE DI REGISTER
   PUSH  SI                              ;SAVE SI REGISTER

   MOV   AX,DATA                         ;SET AX TO POINT TO DATA SEGMENT
```

```
    MOV  DS,AX                          ;MOVE THIS TO THE DS REGISTER

    MOV  DX,INTDATA                     ;READ THE VME DATA WHICH
    IN   AL,DX                          ;CONTAINS THE ROUTING BYTE
    MOV  AH,0H                          ;ZERO THE HIGH BYTE OF AX

    CMP  AX,17H                         ;COMPARE THE ROUTING TO 23 TO SEE IF IT
    JNG  OK                             ;IS ALLOWED - THERE ARE ONLY 23 SECTORS

    JMP  FAR PTR BAD                    ;JUMP IF ROUTING IS INCORRECT

OK: NOP                                 ;ARRIVAL HERE IF ROUTING IS OK

    MOV  TABLE_OFFSET,AX                ;STORE THE LOOK-UP-TABLE OFFSET

    MOV  DX,AX                          ;MULTIPLY AX BY TWO
    ADD  DX,AX                          ;(TWO BYTES PER ENTRY)

    MOV  BX,OFFSET SEGTABLE             ;GET SEGMENT-TABLE OFFSET
    ADD  BX,DX                          ;ADD OFFSET (CALCULATED FROM ROUTING)
    MOV  CX,WORD PTR [BX]               ;GET THE ACTUAL SEGMENT ADDRESS
    MOV  ES,CX                          ;ES NOW HOLDS SEGMENT OF SECTOR ARRAY

    MOV  CX,5H                          ;LOOP IS EXECUTED THIS AMOUNT OF TIMES
WATE: NOP                               ;LOOPING POINT FOR THE WAIT LOOP
    DEC  CX                             ;DECREMENT THE "TIMER" VARIABLE
    JZ   BAD                            ;THE POINT IS BAD IF WE WAIT THIS LONG

    MOV  DX,TDCHIGHIN                   ;SET UP DX FOR TDC READ
    IN   AL,DX                          ;READ THE HIGH TDC INPUT BITS
    AND  AX,10H                         ;ISOLATE THE LOOK-AT-ME BIT (BIT #4)
    JNZ  WATE                           ;LOOP IF LOOK-AT-ME IS STILL HIGH

    MOV  BX,OFFSET START_TABLE          ;GET THE START-TABLE OFFSET
    ADD  BX,TABLE_OFFSET                ;ADD THE OFFSET (THE ROUTING)
    MOV  AL,BYTE PTR [BX]               ;GET THE CORRECT READ INSTRUCTION
    MOV  DX,TDCOUT                      ;SET UP DX FOR TDC READ
    OUT  DX,AL                          ;OUTPUT THE READ INSTRUCTION

    AND  AL,0EFH                        ;ASSERT THE STROBE BIT WITH THIS MASKING
    OUT  DX,AL                          ;OUTPUT READ INSTRUCTION AND STROBES

    MOV  DX,TDCHIGHIN                   ;SET UP DX FOR TDC READ
    IN   AL,DX                          ;READ THE HIGH DATA BITS OF THE START

    MOV  DX,AX                          ;MOVE THE READ RESULT TO DX FOR TESTING
    AND  DX,8H                          ;CHECK FOR OVERFLOW BIT (DATA BIT #12)
    JZ   BAD                            ;JUMP OUT IF IT IS SET - BAD DATA

    XOR  AL,0FFH                        ;INVERT THE HIGH TIMING (INVERSE LOGIC)
    AND  AL,7H                          ;MASK OUT THE TDC STATUS BITS
    MOV  AH,AL                          ;MOVE THE HIGH BITS INTO AH
MOV  DX,TDCLOWIN                ;SET UP DX FOR TDC READ
IN   AL,DX                      ;READ THE LOW DATA BITS OF THE START
XOR  AL,0FFH                    ;INVERT THE HIGH TIMING (INVERSE LOGIC)

CMP  AX,2000                    ;COMPARE DATA TO 2000 (ABOUT FULL SCALE)
JG   BAD                        ;JUMP OUT IF IT IS GREATER - BAD DATA

CMP  AX,15                      ;COMPARE DATA TO 42 (ALMOST ZERO TIME)
JL   BAD                        ;JUMP OUT IF IT IS GREATER - BAD DATA

MOV  CX,AX                      ;SAVE THE TIME FROM TDC START CHANNEL

MOV  DX,TDCOUT                  ;SET UP DX FOR TDC "BUS OFF"
MOV  AL,TDC_OFF                 ;SET UP AL FOR TDC "BUS OFF"
OUT  DX,AL                      ;TURN OFF THE TDC BUS

MOV  BX,OFFSET STOP_TABLE       ;GET THE STOP-TABLE OFFSET
ADD  BX,TABLE_OFFSET            ;ADD THE OFFSET (THE ROUTING)
MOV  AL,BYTE PTR [BX]           ;GET THE CORRECT READ INSTRUCTION
MOV  DX,TDCOUT                  ;SET UP DX FOR TDC READ
OUT  DX,AL                      ;OUTPUT THE READ INSTRUCTION

AND  AL,0EFH                    ;ASSERT THE STROBE BIT WITH THIS MASKING
OUT  DX,AL                      ;OUTPUT READ INSTRUCTION AND STROBES

MOV  DX,TDCHIGHIN               ;SET UP DX FOR TDC READ
IN   AL,DX                      ;READ THE HIGH DATA BITS OF THE STOP

MOV  DX,AX                      ;MOVE THE READ RESULT TO DX FOR TESTING
AND  DX,8H                      ;CHECK FOR OVERFLOW BIT (DATA BIT #12)
JZ   BAD                        ;JUMP OUT IF IT IS SET - BAD DATA

XOR  AL,0FFH                    ;INVERT THE HIGH TIMING (INVERSE LOGIC)
AND  AL,7H                      ;MASK OUT THE TDC STATUS BITS
MOV  AH,AL                      ;MOVE THE HIGH BITS INTO AH

MOV  DX,TDCLOWIN                ;SET UP DX FOR TDC READ
IN   AL,DX                      ;READ THE LOW DATA BITS OF THE STOP
XOR  AL,0FFH                    ;INVERT THE HIGH TIMING (INVERSE LOGIC)

CMP  AX,2000                    ;COMPARE DATA TO 2000 (ABOUT FULL SCALE)
JG   BAD                        ;JUMP OUT IF IT IS GREATER - BAD DATA

CMP  AX,15                      ;COMPARE DATA TO 42 (ALMOST TIME ZERO)
JL   BAD                        ;JUMP OUT IF IT IS GREATER - BAD DATA

ADD  AX,2048                    ;MOVE TIME=0 TO THE MIDDLE OF THE ARRAY

SUB  AX,CX                      ;ACTUAL TIME = STOP - START (CHANNELS)

MOV  BX,AX                      ;MOVE THE RESULT TO BX AND THEN
ADD  BX,AX                      ;MULTIPLY BY 2 (2 BYTES PER COUNT)
```

```
    INC  WORD PTR ES:[BX]               ;INC ELEMENT OF PROPER SECTOR

 BAD:NOP                                ;ARRIVAL HERE IF BAD DATA (JUMPS)

    MOV  DX,TDCOUT                      ;SET UP DX FOR TDC "BUS OFF"
    MOV  AL,TDC_OFF                     ;SET UP AL FOR TDC "BUS OFF"
    OUT  DX,AL                          ;TURN OFF THE TDC BUS

    MOV  AL,SOFTRESET_HIGH              ;SOFTRESET_HIGH=05H=00001001
    MOV  DX,FUNCONTROL                  ;FUNCONTROL=0303H=771 (D)
    OUT  DX,AL                          ;BRING SOFT RESET HIGH (RESET)

    MOV  AL,SOFTRESET_LOW               ;SOFTRESET_LOW=04H=00000100
    MOV  DX,FUNCONTROL                  ;FUNCONTROL=0303H=771 (D)
    OUT  DX,AL                          ;BRING SOFT RESET BACK LOW

    MOV  AL,END_OF_INT                  ;IRQINT_ENABLE=20H=00100000
    OUT  I8259_CONTROL,AL               ;SIGNALS END OF INTERRUPT

    POP  SI                             ;RESTORE SI REGISTER
    POP  DI                             ;RESTORE DI REGISTER
    POP  ES                             ;RESTORE ES REGISTER
    POP  DS                             ;RESTORE DS REGISTER
    POP  DX                             ;RESTORE DX REGISTER
    POP  CX                             ;RESTORE CX REGISTER
    POP  BX                             ;RESTORE BX REGISTER
    POP  AX                             ;RESTORE AX REGISTER

    IRET                                ;RETURN FROM INTERRUPT

PAC_INT  ENDP                           ;END - HARDWARE (RUN) INTERRUPT ROUTINE

;***********************************************************
;-----------------------------------------------------------
PACINT7  PROC  NEAR                     ;PAC INTERRUPTS (SOFTWARE) FUNCTIONS

    MOV  DX,AX                          ;SAVE AX REGISTER

    MOV  AX,DATA                        ;SET AX TO POINT TO DATA SEGMENT
    MOV  DS,AX                          ;MOVE THIS TO THE DS REGISTER

    MOV  AX,DX                          ;RESTORE AX REGISTER

;-----------------------------------------------------------

    CMP  AH,00H                         ;CHECK FOR INTERRUPT FUNCTION 0
    JNE  F1                             ;JUMP IF NOT FUNCTION 0

;*****  NOT A CURRENT FUNCTION  - FUTURE EXPANSION  ***********************

    JMP  F8                             ;FUNCTION DONE - GO RESET AND RETURN
```

```
;................................

 F1: CMP  AH,01H                                ;CLEAR ALL DATA SECTORS
     JNE  F2                                    ;JUMP IF NOT FUNCTION 1

     MOV  AX,0H                                 ;CLEAR AX
 L1O: NOP                                       ;OUTER LOOPING POINT
     MOV  BX,OFFSET SEGTABLE                    ;GET SEGMENT-TABLE OFFSET
     ADD  BX,AX                                 ;ADD THE TABLE INDEX (FIRST TIME)
     ADD  BX,AX                                 ;ADD THE TABLE INDEX (SECOND TIME)
     MOV  CX,WORD PTR[BX]                       ;GET THE ACTUAL SEGMENT ADDRESS (WORD)
     MOV  ES,CX                                 ;MOVE THE SEGMENT ADDRESS TO ES

     MOV  BX,0H                                 ;CLEAR BX
 L1I: NOP                                       ;INNER LOOPING POINT
     MOV  WORD PTR ES:[BX],0H                   ;CLEAR THE ARRAY LOCATION (WORD)
     INC  BX                                    ;INCREMENT ONE BYTE (FIRST TIME)
     INC  BX                                    ;INCREMENT ONE BYTE (SECOND TIME)
     CMP  BX,8192                               ;CHECK IF AT THE END OF THE SECTOR ARRAY
     JNE  L1I                                   ;IF NOT - JUMP TO INNER LOOPING POINT

     INC  AX                                    ;INCREMENT TABLE INDEX
     CMP  AX,24                                 ;CHECK IF BEYOND TOTAL # OF SECTORS
     JNE  L1O                                   ;JUMP TO OUTER LOOPING POINT

     JMP  F8                                    ;FUNCTION DONE - GO RESET AND RETURN

;................................

 F2: CMP  AH,02H                                ;CLEAR A SINGLE DATA SECTOR
     JNE  F3                                    ;JUMP IF NOT FUNCTION 2

     MOV  AH,0H                                 ;ZERO AH, AL CONTAINS THE SECTOR NUMBER
     MOV  BX,OFFSET SEGTABLE                    ;GET SEGMENT-TABLE OFFSET
     ADD  BX,AX                                 ;ADD THE TABLE INDEX (FIRST TIME)
     ADD  BX,AX                                 ;ADD THE TABLE INDEX (SECOND TIME)
     MOV  CX,WORD PTR [BX]                      ;GET THE ACTUAL SEGMENT ADDRESS (WORD)
     MOV  ES,CX                                 ;MOVE THE SEGMENT ADDRESS TO ES

     MOV  BX,0H                                 ;CLEAR BX
 L2: NOP                                        ;LOOPING POINT
     MOV  WORD PTR ES:[BX],0H                   ;CLEAR THE ARRAY LOCATION (WORD)
     INC  BX                                    ;INCREMENT ONE BYTE (FIRST TIME)
     INC  BX                                    ;INCREMENT ONE BYTE (SECOND TIME)
     CMP  BX,8192                               ;CHECK IF AT THE END OF THE SECTOR ARRAY
     JNE  L2                                    ;IF NOT - JUMP TO LOOPING POINT

     JMP  F8                                    ;FUNCTION DONE - GO RESET AND RETURN

;................................

 F3: CMP  AH,03H                                ;TRANSFER DATA INTO QUICKBASIC ARRAY
```

```
    JNE  F4                                ;JUMP IF NOT FUNCTION 3

    MOV  AH,00H                            ;ZERO AH, AL CONTAINS THE SECTOR NUMBER
    MOV  BX,OFFSET SEGTABLE                ;GET SEGMENT-TABLE OFFSET
    ADD  BX,AX                             ;ADD TWICE THE SECTOR NUMBER TO GET THE
    ADD  BX,AX                             ;PROPER SEGMENT ADDRESS IN THE TABLE
    MOV  CX,WORD PTR [BX]                  ;GET THE ACTUAL SEGMENT ADDRESS (WORD)

    PUSH  DS                               ;SAVE THE DS REGISTER

    MOV  DS,CX                             ;MOVE THE SEGMENT ADDRESS TO DS

    MOV  BX,0H                             ;CLEAR BX
 L3: NOP                                   ;LOOPING POINT
    MOV  CX,WORD PTR DS:[BX]               ;GET THE NUMBER OF COUNTS FROM MEMORY
    MOV  ES:[SI],CX                        ;STORE THIS IN QuickBASIC ARRAY
    INC  BX                                ;INCREMENT SOURCE POINTER (FIRST TIME)
    INC  BX                                ;INCREMENT SOURCE POINTER (SECOND TIME)
    INC  SI                                ;INCREMENT DESTINATION POINTER (FIRST)
    INC  SI                                ;INCREMENT DESTINATION POINTER (SECOND)
    CMP  BX,8192                           ;CHECK IF AT THE END OF THE SECTOR ARRAY
    JNE  L3                                ;IF NOT - JUMP TO LOOPING POINT

    POP  DX                                ;
    MOV  DS,DX                             ;RESTORE THE DATA SEGMENT REGISTER

    JMP  F8                                ;FUNCTION DONE - GO RESET AND RETURN

 ;...............................

 F4: CMP  AH,04H                           ;TRANSFER DATA FROM QUICKBASIC ARRAY
    JNE  F5                                ;JUMP IF NOT FUNCTION 4

 ;*****  NOT A CURRENT FUNCTION  - FUTURE EXPANSION  ***********************

    JMP  F8                                ;FUNCTION DONE - GO RESET AND RETURN

 ;...............................

 F5: CMP  AH,05H                           ;SWITCH TO MCA MODE
    JNE  F8                                ;JUMP IF NOT FUNCTION 5

    PUSH  DS                               ;SAVE THE DATA SEGMENT REGISTER

    MOV  AX,SEG MCA_INT                    ;GET SEGMENT OF INTERRUPT ROUTINE
    MOV  DS,AX                             ;MOVE INTO DS FOR INT 21H (25H)
    MOV  DX,OFFSET MCA_INT                 ;
    MOV  AL,IRQ3_INT                       ;CHANGE THE OLD VECTOR TO
    MOV  AH,25H                            ;POINT TO THE NEW ROUTINE
    INT  21H                               ;

    MOV  AX,SEG MCAINT7                    ;GET SEGMENT OF INTERRUPT ROUTINE
    MOV  DS,AX                              ;MOVE INTO DS FOR INT 21H (25H)
    MOV  DX,OFFSET MCAINT7                  ;
    MOV  AL,QB_INT                          ;CHANGE THE OLD VECTOR TO
    MOV  AH,25H                             ;POINT TO THE NEW ROUTINE
    INT  21H                                ;

    POP  DX                                 ;
    MOV  DS,DX                              ;RESTORE THE DATA SEGMENT REGISTER

;...............................

F8: IRET                                    ;RETURN FROM INTERRUPT

PACINT7  ENDP                               ;END OF INTERRUPT ROUTINE

;***********************************************************************
;-------------------------------------------------------------------
MCA_INT  PROC  NEAR                         ;NEW INTERRUPT ROUTINE

    PUSH  AX                                ;SAVE AX REGISTER
    PUSH  BX                                ;SAVE BX REGISTER
    PUSH  CX                                ;SAVE CX REGISTER
    PUSH  DX                                ;SAVE DX REGISTER
    PUSH  BP                                ;SAVE BP REGISTER
    PUSH  DS                                ;SAVE DS REGISTER
    PUSH  ES                                ;SAVE ES REGISTER
    PUSH  DI                                ;SAVE DI REGISTER
    PUSH  SI                                ;SAVE SI REGISTER

    MOV  AX,DATA                            ;SET AX TO POINT TO DATA SEGMENT
    MOV  DS,AX                              ;MOVE THIS TO THE DS REGISTER

;-------------------------------------------------------------------

    MOV  DX,VMEADDLOW                       ;VMEADDLOW = 0305H = 773D
    IN   AL,DX                              ;READ THE ENERGY BYTE
    MOV  AH,0                               ;ZERO THE HIGH BYTE OF AX

;-------------------------------------------------------------------

    MOV  ENERGY,AX                          ;STORE THE ADC READ RESULT
    ADD  AX,ENERGY                          ;MULTIPLY BY TWO (SCREEN SCALE FACTOR)
    MOV  BX,OFFSET MCA_ARRAY                ;GET THE MCA DATA ARRAY OFFSET
    MOV  CX,02H                             ;SET UP CX FOR MULTIPLICATION
    MUL  CX                                 ;MULTIPLY IT (TWO BYTES PER ELEMENT)
    ADD  BX,AX                              ;BX NOW HOLDS OFFSET TO ARRAY ELEMENT

    MOV  AX,DATA                            ;SET AX TO POINT TO DATA SEGMENT
    MOV  ES,AX                              ;MOVE THIS TO THE ES REGISTER
    MOV  AX,ES:[BX]                         ;GET THE NUMBER OF COUNTS FROM ARRAY

    CALL  PLOT                              ;CALL THE PLOT SUBROUTINE
    INC  AX                              ;INC ELEMENT OF DATARRAY

    MOV  ES:[BX],AX                      ;STORE THE # OF COUNTS (IN THE ARRAY)

;-----------------------------------------------------------------

    MOV  AL,SOFTRESET_HIGH               ;SOFTRESET_HIGH=05H=00000101B
    MOV  DX,FUNCONTROL                   ;FUNCONTROL=0303H=771 (D)
    OUT  DX,AL                           ;BRING SOFT RESET HIGH (RESET)

    MOV  AL,SOFTRESET_LOW                ;SOFTRESET_LOW=04H=00001000
    MOV  DX,FUNCONTROL                   ;FUNCONTROL=0303H=771 (D)
    OUT  DX,AL                           ;BRING SOFT RESET BACK LOW

    MOV  AL,END_OF_INT                   ;IRQINT_ENABLE=20H=00100000
    OUT  I8259_CONTROL,AL                ;SIGNALS END OF INTERRUPT

    POP  SI                              ;RESTORE SI REGISTER
    POP  DI                              ;RESTORE DI REGISTER
    POP  ES                              ;RESTORE ES REGISTER
    POP  DS                              ;RESTORE DS REGISTER
    POP  BP                              ;RESTORE BP REGISTER
    POP  DX                              ;RESTORE DX REGISTER
    POP  CX                              ;RESTORE CX REGISTER
    POP  BX                              ;RESTORE BX REGISTER
    POP  AX                              ;RESTORE AX REGISTER

    IRET                                 ;RETURN FROM INTERRUPT

MCA_INT  ENDP                            ;END OF MCA_INT

;****************************************************************
;-----------------------------------------------------------------
PLOT  PROC  NEAR                         ;UNPLOTTING ROUTINE

    PUSH  ES                             ;HELD MCA_ARRAY SEGMENT
    PUSH  AX                             ;AX HOLDS COUNTS
    PUSH  BX                             ;HELD MCA_ARRAY OFFSET
    PUSH  DX                             ;DL HOLDS SECTOR

    MOV  COUNTS,AX                       ;STORE COUNTS

    MOV  AX,0A000H                       ;
    MOV  ES,AX                           ;SET VIDEO SEGMENT
    MOV  AX,ENERGY                       ;GET ENERGY INFORMATION
    ADD  AX,ENERGY                       ;

    MOV  BIT,AX                          ;DETERMINE BIT OF INTEREST

    MOV  CX,8H                           ;DETERMINE BYTE CONTAINING
    MOV  DX,0H                           ;THE BIT OF INTEREST
    DIV  CX                              ;
```

```
MOV  BITE,AX                  ;

MOV  CL,07H                   ;FIGURE WHICH BIT IN THE
SUB  CL,DL                    ;BYTE IS OF INTEREST
MOV  BYTEBIT,CL               ;

MOV  DX,0H                    ;SCALE THE COUNTS -> SCOUNT
MOV  AX,COUNTS                ;
MOV  CX,PLOT_SCALE            ;
DIV  CX                       ;
MOV  SCOUNT,AL                ;

MOV  AX,050H                  ;FIGURE ADDRESS FROM BITE, AND SCOUNT
MUL  SCOUNT                   ;
MOV  CX,5648H                 ;
MOV  BX,BITE                  ;
ADD  BX,CX                    ;
SUB  BX,AX                    ;

MOV  AL,ES:[BX]               ;GET BYTE TO UNPLOT

MOV  DL,01H                   ;MAKE UNPLOT BIT MASK
MOV  CL,BYTEBIT               ;
ROL  DL,CL                    ;
MOV  CX,0FFH                  ;
XOR  CL,DL                    ;
AND  AL,CL                    ;

MOV  ES:[BX],AL               ;UNPLOT BIT

MOV  DX,0H                    ;SCALE THE COUNTS -> SCOUNT
INC  COUNTS                   ;
MOV  AX,COUNTS                ;
MOV  CX,PLOT_SCALE            ;
DIV  CX                       ;
MOV  SCOUNT,AL                ;

MOV  AX,050H                  ;FIGURE ADDRESS FROM BITE AND SCOUNT
MUL  SCOUNT                   ;
MOV  CX,5648H                 ;
MOV  BX,BITE                  ;
ADD  BX,CX                    ;
SUB  BX,AX                    ;

MOV  AL,ES:[BX]               ;GET BYTE TO PLOT

MOV  DL,01H                   ;MAKE PLOT BIT MASK
MOV  CL,BYTEBIT               ;
ROL  DL,CL                    ;
OR   AL,DL                    ;

MOV  ES:[BX],AL               ;PLOT BIT
```

```
    POP  DX                               ;RESTORE DX REGISTER
    POP  BX                               ;RESTORE BX REGISTER
    POP  AX                               ;RESTORE AX REGISTER
    POP  ES                               ;RESTORE ES REGISTER

    RET                                   ;RETURN FROM PLOT ROUTINE

 PLOT  ENDP                               ;END OF PLOT SUBROUTINE

;************************************************************
;---------------------------------------------------------
 MCAINT7  PROC  NEAR                      ;NEW SOFTWARE INTERRUPT ROUTINE

    MOV  DX,AX                            ;SAVE AX IN DX

    MOV  AX,DATA                          ;SET AX TO POINT TO DATA SEGMENT
    MOV  DS,AX                            ;MOVE THIS TO THE DS REGISTER

    MOV  AX,DX                            ;RESTORE AX

;---------------------------------------------------------

    CMP  AH,00H                           ;
    JNE  M1                               ;RETURNS ADDRESS OF MCA_ARRAY
    MOV  DX,SEG MCA_ARRAY                 ;GET SEGMENT OF MCA_ARRAY
    MOV  CX,OFFSET MCA_ARRAY              ;GET OFFSET OF MCA_ARRAY
    JMP  M8                               ;FUNCTION DONE - GO RESET AND RETURN

;...............................

 M1: CMP  AH,01H                          ;CLEAR MCA_ARRAY
    JNE  M2                               ;
    MOV  BX,OFFSET MCA_ARRAY              ;
    MOV  CX, 3FFH                         ;

 C1: MOV  BYTE PTR [BX],00H               ;
    INC  BX                               ;
    DEC  CX                               ;
    JNZ  C1                               ;
    JMP  M8                               ;FUNCTION DONE - GO RESET AND RETURN

;...............................

 M2: CMP  AH,02H                          ;SET SCALE
    JNE  M3                               ;
    MOV  PLOT_SCALE,CX                    ;CX WILL CONTAIN PLOT SCALE
    JMP  M8                               ;FUNCTION DONE - GO RESET AND RETURN

;...............................

 M3: CMP  AH,03H                          ;TRANSFER DATA TO QUICKBASIC ARRAY
    JNE  M4                               ;
    MOV  BX,OFFSET MCA_ARRAY              ;
    MOV  CX,3FFH                          ;

 C3: MOV  DX,[BX]                         ;
    MOV  ES:[SI],DX                       ;
    INC  BX                               ;
    INC  SI                               ;
    DEC  CX                               ;
    JNZ  C3                               ;
    JMP  M8                               ;FUNCTION DONE - GO RESET AND RETURN

 ;................................

 M4: CMP  AH,04H                          ;SWITCH BACK TO PAC MODE
    JNE  M8                               ;

    PUSH DS                               ;SAVE THE DATA SEGMENT REGISTER

    MOV  AX,SEG PAC_INT                   ;GET SEGMENT OF INTERRUPT ROUTINE
    MOV  DS,AX                            ;MOVE INTO DS FOR INT 21H (25H)
    MOV  DX,OFFSET PAC_INT                ;
    MOV  AL,IRQ3_INT                      ;CHANGE THE OLD VECTOR TO
    MOV  AH,25H                           ;POINT TO THE NEW ROUTINE
    INT  21H                              ;

    MOV  AX,SEG PACINT7                   ;
    MOV  DS,AX                            ;
    MOV  DX,OFFSET PACINT7                ;
    MOV  AL,QB_INT                        ;
    MOV  AH,25H                           ;
    INT  21H                              ;

    POP  DX                               ;
    MOV  DS,DX                            ;RESTORE THE DATA SEGMENT REGISTER

 ;................................

 M8: IRET                                 ;RETURN FROM INTERRUPT

 MCAINT7  ENDP                            ;END OF INTERRUPT ROUTINE

;**************************************************************
;--------------------------------------------------------------
INTSETUP  PROC  NEAR                      ;BEGINNING OF SETUP PROGRAM

    MOV  AX,DATA                          ;SET AX TO POINT TO DATA SEGMENT
    MOV  DS,AX                            ;MOVE THIS TO THE DS REGISTER

;--------------------------------------------------------------

    MOV  BX,OFFSET SEGTABLE               ;GET SEGTABLE ADDRESS FOR MAKING IT
MOV  AX,SEG SECTOR011          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX],AX          ;OF SECTOR #0 - EVENT 011

MOV  AX,SEG SECTOR021          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+2],AX        ;OF SECTOR #1 - EVENT 021

MOV  AX,SEG SECTOR031          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+4],AX        ;OF SECTOR #2 - EVENT 031

MOV  AX,SEG SECTOR012          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+6],AX        ;OF SECTOR #3 - EVENT 012

MOV  AX,SEG SECTOR022          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+8],AX        ;OF SECTOR #4 - EVENT 022

MOV  AX,SEG SECTOR032          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+10],AX       ;OF SECTOR #5 - EVENT 032

MOV  AX,SEG SECTOR101          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+12],AX       ;OF SECTOR #6 - EVENT 101

MOV  AX,SEG SECTOR121          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+14],AX       ;OF SECTOR #7 - EVENT 121

MOV  AX,SEG SECTOR131          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+16],AX       ;OF SECTOR #8 - EVENT 131

MOV  AX,SEG SECTOR102          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+18],AX       ;OF SECTOR #9 - EVENT 102

MOV  AX,SEG SECTOR122          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+20],AX       ;OF SECTOR #10 - EVENT 122

MOV  AX,SEG SECTOR132          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+22],AX       ;OF SECTOR #11 - EVENT 132

MOV  AX,SEG SECTOR201          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+24],AX       ;OF SECTOR #12 - EVENT 201

MOV  AX,SEG SECTOR211          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+26],AX       ;OF SECTOR #13 - EVENT 211

MOV  AX,SEG SECTOR231          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+28],AX       ;OF SECTOR #14 - EVENT 231

MOV  AX,SEG SECTOR202          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+30],AX       ;OF SECTOR #15 - EVENT 202

MOV  AX,SEG SECTOR212          ;STORE THE SEGMENT ADDRESS
MOV  WORD PTR [BX+32],AX       ;OF SECTOR #16 - EVENT 212

MOV  AX,SEG SECTOR232          ;STORE THE SEGMENT ADDRESS
```

```
        MOV  WORD PTR [BX+34],AX              ;OF SECTOR #17 - EVENT 232

        MOV  AX,SEG SECTOR301                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+36],AX              ;OF SECTOR #18 - EVENT 301

        MOV  AX,SEG SECTOR311                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+38],AX              ;OF SECTOR #19 - EVENT 311

        MOV  AX,SEG SECTOR321                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+40],AX              ;OF SECTOR #20 - EVENT 321

        MOV  AX,SEG SECTOR302                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+42],AX              ;OF SECTOR #21 - EVENT 302

        MOV  AX,SEG SECTOR312                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+44],AX              ;OF SECTOR #22 - EVENT 312

        MOV  AX,SEG SECTOR322                 ;STORE THE SEGMENT ADDRESS
        MOV  WORD PTR [BX+46],AX              ;OF SECTOR #23 - EVENT 322

;-------------------------------------------------------------

        MOV  BX,OFFSET START_TABLE            ;LOOK-UP TABLE FOR TDC START CHANNELS

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #0 - 011
        MOV  BYTE PTR [BX],AL                 ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #1 - 021
        MOV  BYTE PTR [BX+1],AL               ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #2 - 031
        MOV  BYTE PTR [BX+2],AL               ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #3 - 012
        MOV  BYTE PTR [BX+3],AL               ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #4 - 022
        MOV  BYTE PTR [BX+4],AL               ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_1                     ;START FOR SECTOR #5 - 032
        MOV  BYTE PTR [BX+5],AL               ;DETECTOR #0 - TDC CHANNEL #1

        MOV  AL,TDCREAD_3                     ;START FOR SECTOR #6 - 101
        MOV  BYTE PTR [BX+6],AL               ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_3                     ;START FOR SECTOR #7 - 121
        MOV  BYTE PTR [BX+7],AL               ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_3                     ;START FOR SECTOR #8 - 131
        MOV  BYTE PTR [BX+8],AL               ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_3                     ;START FOR SECTOR #9 - 102
        MOV  BYTE PTR [BX+9],AL          ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_3                ;START FOR SECTOR #10 - 122
        MOV  BYTE PTR [BX+10],AL         ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_3                ;START FOR SECTOR #11 - 132
        MOV  BYTE PTR [BX+11],AL         ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #12 - 201
        MOV  BYTE PTR [BX+12],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #13 - 211
        MOV  BYTE PTR [BX+13],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #14 - 231
        MOV  BYTE PTR [BX+14],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #15 - 202
        MOV  BYTE PTR [BX+15],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #16 - 212
        MOV  BYTE PTR [BX+16],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_6                ;START FOR SECTOR #17 - 232
        MOV  BYTE PTR [BX+17],AL         ;DETECTOR #2 - TDC CHANNEL #6

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #18 - 301
        MOV  BYTE PTR [BX+18],AL         ;DETECTOR #3 - TDC CHANNEL #7

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #19 - 311
        MOV  BYTE PTR [BX+19],AL         ;DETECTOR #3 - TDC CHANNEL #7

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #20 - 321
        MOV  BYTE PTR [BX+20],AL         ;DETECTOR #3 - TDC CHANNEL #7

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #21 - 302
        MOV  BYTE PTR [BX+21],AL         ;DETECTOR #3 - TDC CHANNEL #7

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #22 - 312
        MOV  BYTE PTR [BX+22],AL         ;DETECTOR #3 - TDC CHANNEL #7

        MOV  AL,TDCREAD_7                ;START FOR SECTOR #23 - 322
        MOV  BYTE PTR [BX+23],AL         ;DETECTOR #3 - TDC CHANNEL #7

;------------------------------------------------------------

        MOV  BX,OFFSET STOP_TABLE        ;LOOK-UP TABLE FOR TDC STOP CHANNELS

        MOV  AL,TDCREAD_3                ;STOP FOR SECTOR #0 - 011
        MOV  BYTE PTR [BX],AL            ;DETECTOR #1 - TDC CHANNEL #3

        MOV  AL,TDCREAD_6                ;STOP FOR SECTOR #1 - 021
MOV  BYTE PTR [BX+1],AL          ;DETECTOR #2 - TDC CHANNEL #6

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #2 - 031
MOV  BYTE PTR [BX+2],AL          ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_3                ;STOP FOR SECTOR #3 - 012
MOV  BYTE PTR [BX+3],AL          ;DETECTOR #1 - TDC CHANNEL #3

MOV  AL,TDCREAD_6                ;STOP FOR SECTOR #4 - 022
MOV  BYTE PTR [BX+4],AL          ;DETECTOR #2 - TDC CHANNEL #6

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #5 - 032
MOV  BYTE PTR [BX+5],AL          ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_1                ;STOP FOR SECTOR #6 - 101
MOV  BYTE PTR [BX+6],AL          ;DETECTOR #0 - TDC CHANNEL #1

MOV  AL,TDCREAD_6                ;STOP FOR SECTOR #7 - 121
MOV  BYTE PTR [BX+7],AL          ;DETECTOR #2 - TDC CHANNEL #6

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #8 - 131
MOV  BYTE PTR [BX+8],AL          ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_1                ;STOP FOR SECTOR #9 - 102
MOV  BYTE PTR [BX+9],AL          ;DETECTOR #0 - TDC CHANNEL #1

MOV  AL,TDCREAD_6                ;STOP FOR SECTOR #10 - 122
MOV  BYTE PTR [BX+10],AL         ;DETECTOR #2 - TDC CHANNEL #6

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #11 - 132
MOV  BYTE PTR [BX+11],AL         ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_1                ;STOP FOR SECTOR #12 - 201
MOV  BYTE PTR [BX+12],AL         ;DETECTOR #0 - TDC CHANNEL #1

MOV  AL,TDCREAD_3                ;STOP FOR SECTOR #13 - 211
MOV  BYTE PTR [BX+13],AL         ;DETECTOR #1 - TDC CHANNEL #3

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #14 - 231
MOV  BYTE PTR [BX+14],AL         ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_1                ;STOP FOR SECTOR #15 - 202
MOV  BYTE PTR [BX+15],AL         ;DETECTOR #0 - TDC CHANNEL #1

MOV  AL,TDCREAD_3                ;STOP FOR SECTOR #16 - 212
MOV  BYTE PTR [BX+16],AL         ;DETECTOR #1 - TDC CHANNEL #3

MOV  AL,TDCREAD_7                ;STOP FOR SECTOR #17 - 232
MOV  BYTE PTR [BX+17],AL         ;DETECTOR #3 - TDC CHANNEL #7

MOV  AL,TDCREAD_1                ;STOP FOR SECTOR #18 - 301
MOV  BYTE PTR [BX+18],AL         ;DETECTOR #0 - TDC CHANNEL #1
```

```
    MOV  AL,TDCREAD_3                  ;STOP FOR SECTOR #19 - 311
    MOV  BYTE PTR [BX+19],AL           ;DETECTOR #1 - TDC CHANNEL #3

    MOV  AL,TDCREAD_6                  ;STOP FOR SECTOR #20 - 321
    MOV  BYTE PTR [BX+20],AL           ;DETECTOR #2 - TDC CHANNEL #6

    MOV  AL,TDCREAD_1                  ;STOP FOR SECTOR #21 - 302
    MOV  BYTE PTR [BX+21],AL           ;DETECTOR #0 - TDC CHANNEL #1

    MOV  AL,TDCREAD_3                  ;STOP FOR SECTOR #22 - 312
    MOV  BYTE PTR [BX+22],AL           ;DETECTOR #1 - TDC CHANNEL #3

    MOV  AL,TDCREAD_6                  ;STOP FOR SECTOR #23 - 322
    MOV  BYTE PTR [BX+23],AL           ;DETECTOR #2 - TDC CHANNEL #6

;------------------------------------------------------------

    MOV  AL,INTERRUPT_OFF              ;INTERRUPT_OFF=08H=00001000
    MOV  DX,FUNCONTROL                 ;FUNCONTROL=0303H=771 (D)
    OUT  DX,AL                         ;TURN OFF THE INTERRUPT ON 8255

;------------------------------------------------------------

    PUSH  DS                           ;SAVE THE DATA SEGMENT REGISTER

    MOV  AX,SEG PAC_INT                ;GET SEGMENT OF INTERRUPT ROUTINE
    MOV  DS,AX                         ;MOVE INTO DS FOR INT 21H (25H)
    MOV  DX,OFFSET PAC_INT             ;
    MOV  AL,IRQ3_INT                   ;CHANGE THE OLD VECTOR TO
    MOV  AH,25H                        ;POINT TO THE NEW ROUTINE
    INT  21H                           ;

    MOV  AX,SEG PACINT7                ;GET SEGMENT OF INTERRUPT ROUTINE
    MOV  DS,AX                         ;MOVE INTO DS FOR INT 21H (25H)
    MOV  DX,OFFSET PACINT7             ;
    MOV  AL,QB_INT                     ;CHANGE THE OLD VECTOR TO
    MOV  AH,25H                        ;POINT TO THE NEW ROUTINE
    INT  21H                           ;

    POP  DX                            ;
    MOV  DS,DX                         ;RESTORE THE DATA SEGMENT REGISTER

;------------------------------------------------------------

    MOV  AX,0                          ;CLEAR THE AX REGISTER
    IN   AL,I8259_IMASKR               ;GET THE CURRENT INTERRUPT MASK
    AND  AX,0F7H                       ;SET THE 4TH BIT TO ZERO (ENABLE IRQ3)
    OUT  I8259_IMASKR,AL               ;SEND THIS RESULT TO MASK REGISTER

    MOV  AL,SOFTRESET_HIGH             ;************************************
    MOV  DX,FUNCONTROL                 ;
    OUT  DX,AL                         ;
```

```
                                            ;*          RESET THE SYSTEM            *
    MOV  AL,SOFTRESET_LOW                   ;
    MOV  DX,FUNCONTROL                      ;
    OUT  DX,AL                              ;************************************

    MOV  AX,3100H                           ;TERMINATE AND STAY RESIDENT FUNCTION
    MOV  DX,30D4H                           ;DX CONTAINS THE NUMBER OF RESERVED
    INT  21h                                ;PARAGRAPHS (256 BYTES/PARAGRAPH)

INTSETUP  ENDP                              ;END OF INTERRUPT SET-UP ROUTINE

;************************************************************
;--------------------------------------------------------------
CODE  ENDS                                  ;END OF EXECUTABLE CODE

END  INTSETUP                               ;INTSETUP WILL BE THE START OF EXECUTION
```

E. Listing of the QuickBASIC module SETWIND.BAS (module of VMEMCA.BAS)

In this appendix, the QuickBASIC source code for the SETWIND.BAS module can be found. It is **not** a stand-alone program; It is used in conjunction with the spectrometer control program (VMEMCA.BAS). The reason it is present here is that it contains all of the subroutines that are necessary to communicate with the boards in the VME card cage. This program is responsible for setting the enrgy windows (in the WINDOW RAM) and setting up the contents of the VALID RAM. Again, this listing is presented without detailed explanation, but it too can be understood by an experienced programmer by reading the commented source code.

```
'***************************************************************************
'*      SETWIND.BAS                         2/9/92                         *
'*      written by  DARREN W. STEVENS                                      *
'***************************************************************************

'*****  THE SUBROUTINES USED IN THE PROGRAM  ***********************

DECLARE SUB set3windows (ch%, W1L%, W1R%, W2L%, W2R%, W3L%, W3R%)
DECLARE SUB set3validram ()
DECLARE SUB set2windows (ch%, W1L%, W1R%, W2L%, W2R%)
DECLARE SUB set2validram ()
DECLARE SUB setchannel (ch%)
DECLARE SUB setrunmode ()
DECLARE SUB read3windows (ch%, W1L%, W1R%, W2L%, W2R%, W3L%, W3R%)

'*****  QUATECH ADDRESSES BY PORT LOCATION  (I/O REFERS TO COMPUTER END) *******

CONST PORTA1 = &H300                    'QUATECH HEADER #1 - PORT A
CONST INTDATA = &H300                   'INTERRUPT DATA IS INPUT HERE

CONST PORTB1 = &H301                    'QUATECH HEADER #1 - PORT B
CONST FUNKTION = &H301                  'VME FUNCTION IS OUTPUT HERE

CONST PORTC1 = &H302                    'QUATECH HEADER #1 - PORT C
CONST FUNINTSTUFF = &H302               'FUNCTION/INTERRUPT CONTROL STUFF

CONST CONTROL1 = &H303                  'QUATECH HEADER #1 - CONTROL PORT
CONST FUNCONTROL = &H303                'FUNCTION HEADER CONTROL PORT

CONST PORTA2 = &H304                    'QUATECH HEADER #2 - PORT A
CONST VMEDATA = &H304                   'VME DATA BUS IS CONNECTED HERE

CONST PORTB2 = &H305                    'QUATECH HEADER #2 - PORT B
CONST VMEADDLOW = &H305                 'VME ADDRESS BUS (LOW) CONNECTED HERE

CONST PORTC2 = &H306                    'QUATECH HEADER #2 - PORT C
CONST VMEADDHIGH = &H306                'VME ADDRESS BUS (HIGH) CONNECTED HERE

CONST CONTROL2 = &H307                  'QUATECH HEADER #2 - CONTROL PORT
CONST ADDATACONTROL = &H307             'ADDRESS/DATA HEADER CONTROL PORT

CONST PORTA3 = &H308                    'QUATECH HEADER #3 - PORT A
CONST TDCOUT = &H308                    'TDC CONTROL BITS ARE OUTPUT HERE

CONST PORTB3 = &H309                    'QUATECH HEADER #3 - PORT B
CONST TDCHIGHIN = &H309                 'TDC DATA AND STATUS ARE INPUT HERE

CONST PORTC3 = &H30A                    'QUATECH HEADER #3 - PORT C
CONST TDCLOWIN = &H30A                  'TDC DATA (LOW 8 BITS) ARE INPUT HERE
CONST CONTROL3 = &H30B                 'QUATECH HEADER #3 - CONTROL PORT
CONST TDCCONTROL = &H30B               'TDC HEADER CONTROL PORT

'*****  THE HEADER (QUATECH) CONTROL WORDS USED IN RUNNING  *****************

CONST FUNMODE = &HB8                   'FUNCTION HEADER CONTROL WORD

CONST TDCMODE = &H8B                   'TDC HEADER CONTROL WORD

CONST READRAM = &H90                   'READ FUNCTION - ADDRESS/DATA HEADER

CONST WRITERAM = &H80                  'WRITE FUNCTION - ADDRESS/DATA HEADER

CONST CALIBRATION = &H9B               'CALIBRATE FUNCTION - ADD/DATA HEADER

CONST RUNMODE = &H9B                   'RUN FUNCTION - ADDRESS/DATA HEADER

'*****  THE BIT SET/RESET FUNCTIONS SENT TO HEADER CONTROL PORTS ***************

CONST PORTC0RESET = &H0                'PORT C - BIT 0 - LOGIC LOW
CONST FUNCLOCKLOW = &H0                'VME FUNCTION CLOCK LOW
CONST PORTC0SET = &H1                  'PORT C - BIT 0 - LOGIC HIGH
CONST FUNCLOCKHIGH = &H1               'VME FUNCTION CLOCK HIGH

CONST PORTC1RESET = &H2                'PORT C - BIT 1 - LOGIC LOW
CONST FUNENABLE = &H2                  'VME FUNCTION ENABLE LOW (ENABLED)
CONST PORTC1SET = &H3                  'PORT C - BIT 1 - LOGIC HIGH
CONST FUNDISABLE = &H3                 'VME FUNCTION ENABLE HIGH (DISABLED)

CONST PORTC2RESET = &H4                'PORT C - BIT 2 - LOGIC LOW
CONST SOFTRESETLOW = &H4               'SOFT RESET LOW
CONST PORTC2SET = &H5                  'PORT C - BIT 2 - LOGIC HIGH
CONST SOFTRESETHIGH = &H5              'SOFT RESET HIGH

CONST PORTC4RESET = &H8                'PORT C - BIT 4 - LOGIC LOW
CONST INTERRUPTOFF = &H8               'INTERRUPT ENABLE LOW (DISABLED AT 8255)
CONST PORTC4SET = &H9                  'PORT C - BIT 4 - LOGIC HIGH
CONST INTERRUPTON = &H9                'INTERRUPT ENABLE HIGH (ENABLED AT 8255)

'*****  THE VME FUNCTIONS (EPROM ADDRESSES) OUTPUT TO VME BUS *****************

CONST NULL = &H0                       'EVERYTHING DISABLED - VME BUS ISOLATED

CONST VRAMWRSETUP = &H1                'VALID RAM WRITE SET-UP
CONST VRAMWRITE = &H2                  'VALID RAM WRITE
CONST VRAMREAD = &H3                   'VALID RAM READ

CONST E0RAMWRSETUP = &H4               'ENERGY 0 (WINDOW) RAM WRITE SET-UP
CONST E0RAMWRITE = &H5                 'ENERGY 0 (WINDOW) RAM WRITE
CONST E0RAMREAD = &H6                  'ENERGY 0 (WINDOW) RAM READ

CONST E1RAMWRSETUP = &H7               'ENERGY 1 (WINDOW) RAM WRITE SET-UP
```

```
  CONST E1RAMWRITE = &H8                    'ENERGY 1 (WINDOW) RAM WRITE
  CONST E1RAMREAD = &H9                     'ENERGY 1 (WINDOW) RAM READ

  CONST E2RAMWRSETUP = &HA                  'ENERGY 2 (WINDOW) RAM WRITE SET-UP
  CONST E2RAMWRITE = &HB                    'ENERGY 2 (WINDOW) RAM WRITE
  CONST E2RAMREAD = &HC                     'ENERGY 2 (WINDOW) RAM READ

  CONST E3RAMWRSETUP = &HD                  'ENERGY 3 (WINDOW) RAM WRITE SET-UP
  CONST E3RAMWRITE = &HE                    'ENERGY 3 (WINDOW) RAM WRITE
  CONST E3RAMREAD = &HF                     'ENERGY 3 (WINDOW) RAM READ

  CONST DATARUN = &H10                      'SETS UP VME BOARDS FOR RUN MODE

  CONST ADCRAMCH0 = &H20                    'ENERGY BOARD 0 - ADC AND RAM CHECK
  CONST ADCRAMCH1 = &H21                    'ENERGY BOARD 1 - ADC AND RAM CHECK
  CONST ADCRAMCH2 = &H22                    'ENERGY BOARD 2 - ADC AND RAM CHECK
  CONST ADCRAMCH3 = &H23                    'ENERGY BOARD 3 - ADC AND RAM CHECK

  CONST CALIBRATE0 = &H80                   'ENERGY BOARD 0 - ADC CALIBRATE
  CONST CALIBRATE1 = &HA0                   'ENERGY BOARD 1 - ADC CALIBRATE
  CONST CALIBRATE2 = &HC0                   'ENERGY BOARD 2 - ADC CALIBRATE
  CONST CALIBRATE3 = &HE0                   'ENERGY BOARD 3 - ADC CALIBRATE

'************************************************************

END                                         'END OF THE SETWIND.BASMAIN MODULE

'---------------------------------------------
SUB read3windows (ch%, W1L%, W1R%, W2L%, W2R%, W3L%, W3R%)

  OUT FUNKTION, NULL                        'SEND OUT THE NULL FUNCTION
                                            '
  OUT FUNCONTROL, FUNCLOCKLOW               '
  OUT FUNCONTROL, FUNCLOCKHIGH              'CLOCK THE FUNCTION INTO THE LATCHES

  OUT ADDATACONTROL, READRAM                'SET QUATECH TO READ THE RAM

END SUB                                     '

'---------------------------------------------
SUB set2validram

***** THIS SUBROUINE WILL BE WRITTEN FOR TWO WINDOW STUDIES (FUTURE) *****

END SUB                                     'END OF set2validram

'---------------------------------------------
SUB set2windows (ch%, W1L%, W1R%, W2L%, W2R%)

***** THIS SUBROUINE WILL BE WRITTEN FOR TWO WINDOW STUDIES (FUTURE) *****

END SUB                                     'END OF set2windows
```

```
'-----------------------------------------------
SUB set3validram

'*****  THE VARIABLES USED IN THIS PROCEDURE  ******************

  NOEVENT% = 255                        'NOT ANY EVENT - RAM WILL STORE &HFF

  EVENT011% = 0                         'START 0 - STOP 1 - LOW ENERGY STOP
  EVENT021% = 1                         'START 0 - STOP 2 - LOW ENERGY STOP
  EVENT031% = 2                         'START 0 - STOP 3 - LOW ENERGY STOP
  EVENT012% = 3                         'START 0 - STOP 1 - HIGH ENERGY STOP
  EVENT022% = 4                         'START 0 - STOP 2 - HIGH ENERGY STOP
  EVENT032% = 5                         'START 0 - STOP 3 - HIGH ENERGY STOP
  EVENT101% = 6                         'START 1 - STOP 0 - LOW ENERGY STOP
  EVENT121% = 7                         'START 1 - STOP 2 - LOW ENERGY STOP
  EVENT131% = 8                         'START 1 - STOP 3 - LOW ENERGY STOP
  EVENT102% = 9                         'START 1 - STOP 0 - HIGH ENERGY STOP
  EVENT122% = 10                        'START 1 - STOP 2 - HIGH ENERGY STOP
  EVENT132% = 11                        'START 1 - STOP 3 - HIGH ENERGY STOP
  EVENT201% = 12                        'START 2 - STOP 0 - LOW ENERGY STOP
  EVENT211% = 13                        'START 2 - STOP 1 - LOW ENERGY STOP
  EVENT231% = 14                        'START 2 - STOP 3 - LOW ENERGY STOP
  EVENT202% = 15                        'START 2 - STOP 0 - HIGH ENERGY STOP
  EVENT212% = 16                        'START 2 - STOP 1 - HIGH ENERGY STOP
  EVENT232% = 17                        'START 2 - STOP 3 - HIGH ENERGY STOP
  EVENT301% = 18                        'START 3 - STOP 0 - LOW ENERGY STOP
  EVENT311% = 19                        'START 3 - STOP 1 - LOW ENERGY STOP
  EVENT321% = 20                        'START 3 - STOP 2 - LOW ENERGY STOP
  EVENT302% = 21                        'START 3 - STOP 0 - HIGH ENERGY STOP
  EVENT312% = 22                        'START 3 - STOP 1 - HIGH ENERGY STOP
  EVENT322% = 23                        'START 3 - STOP 2 - HIGH ENERGY STOP

  MASK011% = &H11                       '0000 0001 0001 = 000 000 010 001 (011 - 0)
  MASK021% = &H81                       '0000 1000 0001 = 000 010 000 001 (021 - 1)
  MASK031% = &H401                      '0100 0000 0001 = 010 000 000 001 (031 - 2)
  MASK012% = &H19                       '0000 0001 1001 = 000 000 011 001 (012 - 3)
  MASK022% = &HC1                       '0000 1100 0001 = 000 011 000 001 (022 - 4)
  MASK032% = &H601                      '0110 0000 0001 = 011 000 000 001 (032 - 5)
  MASK101% = &HA                        '0000 0000 1010 = 000 000 001 010 (101 - 6)
  MASK121% = &H88                       '0000 1000 1000 = 000 010 001 000 (121 - 7)
  MASK131% = &H408                      '0100 0000 1000 = 010 000 001 000 (131 - 8)
  MASK102% = &HB                        '0000 0000 1011 = 000 000 001 011 (102 - 9)
  MASK122% = &HC8                       '0000 1100 1000 = 000 011 001 000 (122 - 10)
  MASK132% = &H608                      '0110 0000 1000 = 011 000 001 000 (132 - 11)
  MASK201% = &H42                       '0000 0100 0010 = 000 001 000 010 (201 - 12)
  MASK211% = &H50                       '0000 0101 0000 = 000 001 010 000 (211 - 13)
  MASK231% = &H440                      '0100 0100 0000 = 010 001 000 000 (231 - 14)
  MASK202% = &H43                       '0000 0100 0011 = 000 001 000 011 (202 - 15)
  MASK212% = &H58                       '0000 0101 1000 = 000 001 011 000 (212 - 16)
  MASK232% = &H640                      '0110 0100 0000 = 011 001 000 000 (232 - 17)
  MASK301% = &H202                      '0010 0000 0010 = 001 000 000 010 (301 - 18)
  MASK311% = &H210                      '0010 0001 0000 = 001 000 010 000 (311 - 19)
  MASK321% = &H280                       '0010 1000 0000 = 001 010 000 000 (321 - 20)
  MASK302% = &H203                       '0010 0000 0011 = 001 000 000 011 (302 - 21)
  MASK312% = &H218                       '0010 0001 1000 = 001 000 011 000 (312 - 22)
  MASK322% = &H2C0                       '0010 1100 0000 = 001 011 000 000 (322 - 23)

'***** THE ACTUAL CODE OF SUBROUTINE ***************************

  OUT FUNKTION, NULL                     'SEND OUT THE NULL FUNCTION
                                         '
  OUT FUNCONTROL, FUNCLOCKLOW            '
  OUT FUNCONTROL, FUNCLOCKHIGH           'CLOCK THE FUNCTION INTO THE LATCHES

  OUT ADDATACONTROL, WRITERAM            'SET QUATECH TO WRITE THE RAM

  FOR J% = 0 TO 15                       'THE HIGH ADDRESS VARIES FROM 0 TO 15

    OUT VMEADDHIGH, J%                   'SET UP THE HIGH ADDRESS BITS

    FOR I% = 0 TO 255                    'THE LOW ADDRESS VARIES FROM 0 TO 255

      OUT VMEADDLOW, I%                  'SET UP THE LOW ADDRESS BITS

      EVENTTEST% = J% * 256 + I%         'THIS WILL BE THE TESTED WORD

      SELECT CASE EVENTTEST%             'SELECT WHAT TO LOAD INTO VALID RAM

        CASE MASK011%                    'LOAD SECTOR NUMBER OF EVENT 011
          OUT VMEDATA, EVENT011%         'START 0 - STOP 1 - LOW ENERGY STOP

        CASE MASK021%                    'LOAD SECTOR NUMBER OF EVENT 021
          OUT VMEDATA, EVENT021%         'START 0 - STOP 2 - LOW ENERGY STOP

        CASE MASK031%                    'LOAD SECTOR NUMBER OF EVENT 031
          OUT VMEDATA, EVENT031%         'START 0 - STOP 3 - LOW ENERGY STOP

        CASE MASK012%                    'LOAD SECTOR NUMBER OF EVENT 012
          OUT VMEDATA, EVENT012%         'START 0 - STOP 1 - HIGH ENERGY STOP

        CASE MASK022%                    'LOAD SECTOR NUMBER OF EVENT 022
          OUT VMEDATA, EVENT022%         'START 0 - STOP 2 - HIGH ENERGY STOP

        CASE MASK032%                    'LOAD SECTOR NUMBER OF EVENT 032
          OUT VMEDATA, EVENT032%         'START 0 - STOP 3 - HIGH ENERGY STOP

        CASE MASK101%                    'LOAD SECTOR NUMBER OF EVENT 101
          OUT VMEDATA, EVENT101%         'START 1 - STOP 0 - LOW ENERGY STOP

        CASE MASK121%                    'LOAD SECTOR NUMBER OF EVENT 121
          OUT VMEDATA, EVENT121%         'START 1 - STOP 2 - LOW ENERGY STOP

        CASE MASK131%                    'LOAD SECTOR NUMBER OF EVENT 131
          OUT VMEDATA, EVENT131%         'START 1 - STOP 3 - LOW ENERGY STOP
```

```
CASE MASK102%                   'LOAD SECTOR NUMBER OF EVENT 102
  OUT VMEDATA, EVENT102%        'START 1 - STOP 0 - HIGH ENERGY STOP

CASE MASK122%                   'LOAD SECTOR NUMBER OF EVENT 122
  OUT VMEDATA, EVENT122%        'START 1 - STOP 2 - HIGH ENERGY STOP

CASE MASK132%                   'LOAD SECTOR NUMBER OF EVENT 132
  OUT VMEDATA, EVENT132%        'START 1 - STOP 3 - HIGH ENERGY STOP

CASE MASK201%                   'LOAD SECTOR NUMBER OF EVENT 201
  OUT VMEDATA, EVENT201%        'START 2 - STOP 0 - LOW ENERGY STOP

CASE MASK211%                   'LOAD SECTOR NUMBER OF EVENT 211
  OUT VMEDATA, EVENT211%        'START 2 - STOP 1 - LOW ENERGY STOP

CASE MASK231%                   'LOAD SECTOR NUMBER OF EVENT 231
  OUT VMEDATA, EVENT231%        'START 2 - STOP 3 - LOW ENERGY STOP

CASE MASK202%                   'LOAD SECTOR NUMBER OF EVENT 202
  OUT VMEDATA, EVENT202%        'START 2 - STOP 0 - HIGH ENERGY STOP

CASE MASK212%                   'LOAD SECTOR NUMBER OF EVENT 212
  OUT VMEDATA, EVENT212%        'START 2 - STOP 1 - HIGH ENERGY STOP

CASE MASK232%                   'LOAD SECTOR NUMBER OF EVENT 232
  OUT VMEDATA, EVENT232%        'START 2 - STOP 3 - HIGH ENERGY STOP

CASE MASK301%                   'LOAD SECTOR NUMBER OF EVENT 301
  OUT VMEDATA, EVENT301%        'START 3 - STOP 0 - LOW ENERGY STOP

CASE MASK311%                   'LOAD SECTOR NUMBER OF EVENT 311
  OUT VMEDATA, EVENT311%        'START 3 - STOP 1 - LOW ENERGY STOP

CASE MASK321%                   'LOAD SECTOR NUMBER OF EVENT 321
  OUT VMEDATA, EVENT321%        'START 3 - STOP 2 - LOW ENERGY STOP

CASE MASK302%                   'LOAD SECTOR NUMBER OF EVENT 302
  OUT VMEDATA, EVENT302%        'START 3 - STOP 0 - HIGH ENERGY STOP

CASE MASK312%                   'LOAD SECTOR NUMBER OF EVENT 312
  OUT VMEDATA, EVENT312%        'START 3 - STOP 1 - HIGH ENERGY STOP

CASE MASK322%                   'LOAD SECTOR NUMBER OF EVENT 322
  OUT VMEDATA, EVENT322%        'START 3 - STOP 2 - HIGH ENERGY STOP

CASE ELSE                       'LOAD NO EVENT FLAG INTO RAM
  OUT VMEDATA, NOEVENT%         'NO EVENT, LOAD RAM ACCORDINGLY

END SELECT                      'END OF THE CHOICES (24 SECTORS)

OUT FUNKTION, VRAMWRSETUP       'SET UP THE VME BUS FOR VALID RAM WRITE
```

```
    OUT FUNCONTROL, FUNCLOCKLOW       '
    OUT FUNCONTROL, FUNCLOCKHIGH      'CLOCK THE FUNCTION INTO THE LATCHES

    OUT FUNKTION, VRAMWRITE           'ACTUALLY WRITE THE VALID RAM LOCATION


    OUT FUNCONTROL, FUNCLOCKLOW       '
    OUT FUNCONTROL, FUNCLOCKHIGH      'CLOCK THE FUNCTION INTO THE LATCHES

    OUT FUNKTION, NULL                'SEND OUT THE NULL FUNCTION
                                      '
    OUT FUNCONTROL, FUNCLOCKLOW       '
    OUT FUNCONTROL, FUNCLOCKHIGH      'CLOCK THE FUNCTION INTO THE LATCHES

   NEXT I%                            'END OF INNER LOOP

  NEXT J%                             'END OF OUTER LOOP

  OUT FUNKTION, NULL                  'SEND OUT THE NULL FUNCTION
                                      '
  OUT FUNCONTROL, FUNCLOCKLOW         '
  OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

END SUB                               'END OF set3validram

'-----------------------------------------------
SUB set3windows (ch%, W1L%, W1R%, W2L%, W2R%, W3L%, W3R%)

  OUT FUNKTION, NULL                  'SEND OUT THE NULL FUNCTION
                                      '
  OUT FUNCONTROL, FUNCLOCKLOW         '
  OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

  OUT ADDATACONTROL, WRITERAM         'SET QUATECH TO ALL OUTPUT

  SELECT CASE ch%                     'CHECK AND SET UP THE CHANNEL

   CASE 0                             'CHANNEL #0

     RAMSETUP% = E0RAMWRSETUP         'RAMSETUP VARIABLE USED LATER IN SET-UP
     RAMWRITE% = E0RAMWRITE           'RAMWRITE VARIABLE USED LATER IN WRITE

   CASE 1                             'CHANNEL #1

     RAMSETUP% = E1RAMWRSETUP         'RAMSETUP VARIABLE USED LATER IN SET-UP
     RAMWRITE% = E1RAMWRITE           'RAMWRITE VARIABLE USED LATER IN WRITE

   CASE 2                             'CHANNEL #2
     RAMSETUP% = E2RAMWRSETUP         'RAMSETUP VARIABLE USED LATER IN SET-UP
     RAMWRITE% = E2RAMWRITE           'RAMWRITE VARIABLE USED LATER IN WRITE
  CASE 3                                'CHANNEL #3

    RAMSETUP% = E3RAMWRSETUP            'RAMSETUP VARIABLE USED LATER IN SET-UP
    RAMWRITE% = E3RAMWRITE              'RAMWRITE VARIABLE USED LATER IN WRITE

 END SELECT                             'END OF INITIALIZING THOSE TWO VARIABLES

 OUT VMEADDHIGH, 0                      'ZERO THE HIGH ADDRESS BITS (UNUSED)

 FOR I% = 0 TO 255                      'LOOP FOR THE 256 RAM LOCATIONS

  OUT FUNKTION, NULL                    'SEND OUT THE NULL FUNCTION
                                        '
  OUT FUNCONTROL, FUNCLOCKLOW           '
  OUT FUNCONTROL, FUNCLOCKHIGH          'CLOCK THE FUNCTION INTO THE LATCHES

  OUT VMEADDLOW, I%                     'SET UP THE LOW ADDRESS BITS

  IF ((I% > W1L%) AND I% < W1R%) THEN   'WINDOW #1 (LOWEST ENERGY)
    OUT VMEDATA, 1                      'OUTPUT THE DATA VALUE FOR WINDOW # 1

  ELSEIF ((I% > W2L%) AND I% < W2R%) THEN      'WINDOW #2 (MIDDLE ENERGY)
    OUT VMEDATA, 2                             'OUTPUT VALUE FOR WINDOW # 2

  ELSEIF ((I% > W3L%) AND I% < W3R%) THEN      'WINDOW #3 (HIGHEST ENERGY)
    OUT VMEDATA, 3                             'OUTPUT VALUE FOR WINDOW # 3

  ELSE                                  'WITHIN NO WINDOW
    OUT VMEDATA, 0                      'OUTPUT THE DATA VALUE FOR NO WINDOW

  END IF                                'END OF DATA OUTPUT CHOICES

  OUT FUNKTION, RAMSETUP%               'SET UP THE RAM WRITING

  OUT FUNCONTROL, FUNCLOCKLOW           '
  OUT FUNCONTROL, FUNCLOCKHIGH          'CLOCK THE FUNCTION INTO THE LATCHES

  OUT FUNKTION, RAMWRITE%               'WRITE VALUE INTO THE RAM
                                        '
  OUT FUNCONTROL, FUNCLOCKLOW           '
  OUT FUNCONTROL, FUNCLOCKHIGH          'CLOCK THE FUNCTION INTO THE LATCHES

 NEXT I%                                'END OF LOOP

 OUT FUNKTION, NULL                     'SEND OUT THE NULL FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW            '
 OUT FUNCONTROL, FUNCLOCKHIGH           'CLOCK THE FUNCTION INTO THE LATCHES

END SUB                                 'END OF set3windows

'----------------------------------------------
```

```
SUB setchannel (ch%)

  OUT FUNKTION, NULL                      'SEND OUT THE NULL FUNCTION
                                          '
  OUT FUNCONTROL, FUNCLOCKLOW             '
  OUT FUNCONTROL, FUNCLOCKHIGH            'CLOCK THE FUNCTION INTO THE LATCHES

  OUT ADDATACONTROL, CALIBRATION          'SET QUATECH TO CALIBRATE MODE

  SELECT CASE ch%                         'CHECK AND SET UP THE CHANNEL

    CASE 0                                'CHANNEL #0
      OUT FUNKTION, CALIBRATE0            'OUTPUT THE CALIBRATE 0 FUNCTION
      OUT FUNCONTROL, FUNCLOCKLOW         '
      OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

    CASE 1                                'CHANNEL #1
      OUT FUNKTION, CALIBRATE1            'OUTPUT THE CALIBRATE 1 FUNCTION
      OUT FUNCONTROL, FUNCLOCKLOW         '
      OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

    CASE 2                                'CHANNEL #2
      OUT FUNKTION, CALIBRATE2            'OUTPUT THE CALIBRATE 2 FUNCTION
      OUT FUNCONTROL, FUNCLOCKLOW         '
      OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

    CASE 3                                'CHANNEL #3
      OUT FUNKTION, CALIBRATE3            'OUTPUT THE CALIBRATE 3 FUNCTION
      OUT FUNCONTROL, FUNCLOCKLOW         '
      OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

  END SELECT                              'END OF THE CHOICES

END SUB                                   'END OF setchannel

'----------------------------------------------
SUB setrunmode

  OUT FUNCONTROL, INTERRUPTOFF            'TURN OFF THE INTERRUPTS

  OUT FUNKTION, NULL                      'SEND OUT THE NULL FUNCTION
                                          '
  OUT FUNCONTROL, FUNCLOCKLOW             '
  OUT FUNCONTROL, FUNCLOCKHIGH            'CLOCK THE FUNCTION INTO THE LATCHES

  OUT ADDATACONTROL, RUNMODE              'SET ADDRESS/DATA HEADER TO RUN MODE

  OUT FUNKTION, DATARUN                   'SEND OUT THE RUN FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW             '
  OUT FUNCONTROL, FUNCLOCKHIGH            'CLOCK THE FUNCTION INTO THE LATCHES
```

```
END SUB                                    'END OF setrunmode
```

F. Listing of the QuickBASIC program READRAM.BAS (a stand-alone program)

In this appendix, the listing of the stand-alone program READRAM.BAS can be found. It is a program that will be incorporated into the module SETWIND.BAS in the future. It is listed here to illustrate the way that the VALID RAM and WINDOW RAMs can be interrogated to find out information about current status of their contents. The program was written in such a way that the contents of each of the WINDOW RAMs is read and then the current window settings for each of the ENERGY channels (0-3) are displayed. The valid ram contents is then checked for accuracy and the results displayed. If an error is detected, then a message is displayed and the user can take appropriate action. All subroutines are written with the assumption that there is a single START window and two STOP windows. The purpose of this program is to give the user a way to check the status of the system after the program VMEMCA.EXE has been run and before the data collection control program (VMEPAC.EXE) is executed. This program is also presented with commented source code for analysis by the reader.

```
'**************************************************************
'*    READRAM.BAS                      2/9/92                 *
'*    written by  DARREN W. STEVENS                           *
'**************************************************************

'*****  THE SUBROUTINES USED IN THE PROGRAM  **********************

DECLARE SUB SETRUNMODE ()
DECLARE SUB READENERGY ()
DECLARE SUB READVALID ()

'**********************************************************

CONST TRUE = 0                          'DEFINE TRUE
CONST FALSE = -1                        'DEFINE FALSE

'*****  QUATECH ADDRESSES BY PORT LOCATION  (I/O REFERS TO COMPUTER END) *******

CONST PORTA1 = &H300                    'QUATECH HEADER #1 - PORT A
CONST INTDATA = &H300                   'INTERRUPT DATA IS INPUT HERE

CONST PORTB1 = &H301                    'QUATECH HEADER #1 - PORT B
CONST FUNKTION = &H301                  'VME FUNCTION IS OUTPUT HERE

CONST PORTC1 = &H302                    'QUATECH HEADER #1 - PORT C
CONST FUNINTSTUFF = &H302               'FUNCTION/INTERRUPT CONTROL STUFF

CONST CONTROL1 = &H303                  'QUATECH HEADER #1 - CONTROL PORT
CONST FUNCONTROL = &H303                'FUNCTION HEADER CONTROL PORT

CONST PORTA2 = &H304                    'QUATECH HEADER #2 - PORT A
CONST VMEDATA = &H304                   'VME DATA BUS IS CONNECTED HERE

CONST PORTB2 = &H305                    'QUATECH HEADER #2 - PORT B
CONST VMEADDLOW = &H305                 'VME ADDRESS BUS (LOW) CONNECTED HERE

CONST PORTC2 = &H306                    'QUATECH HEADER #2 - PORT C
CONST VMEADDHIGH = &H306                'VME ADDRESS BUS (HIGH) CONNECTED HERE

CONST CONTROL2 = &H307                  'QUATECH HEADER #2 - CONTROL PORT
CONST ADDATACONTROL = &H307             'ADDRESS/DATA HEADER CONTROL PORT

CONST PORTA3 = &H308                    'QUATECH HEADER #3 - PORT A
CONST TDCOUT = &H308                    'TDC CONTROL BITS ARE OUTPUT HERE

CONST PORTB3 = &H309                    'QUATECH HEADER #3 - PORT B
CONST TDCHIGHIN = &H309                 'TDC DATA AND STATUS ARE INPUT HERE

CONST PORTC3 = &H30A                    'QUATECH HEADER #3 - PORT C
CONST TDCLOWIN = &H30A                  'TDC DATA (LOW 8 BITS) ARE INPUT HERE

CONST CONTROL3 = &H30B                  'QUATECH HEADER #3 - CONTROL PORT
CONST TDCCONTROL = &H30B                'TDC HEADER CONTROL PORT

'***** THE HEADER (QUATECH) CONTROL WORDS USED IN RUNNING *****************

CONST FUNMODE = &HB8                    'FUNCTION HEADER CONTROL WORD

CONST TDCMODE = &H8B                    'TDC HEADER CONTROL WORD

CONST READRAM = &H90                    'READ FUNCTION - ADDRESS/DATA HEADER

CONST WRITERAM = &H80                   'WRITE FUNCTION - ADDRESS/DATA HEADER

CONST CALIBRATION = &H9B                'CALIBRATE FUNCTION - ADD/DATA HEADER

CONST RUNMODE = &H9B                    'RUN FUNCTION - ADDRESS/DATA HEADER

'***** THE BIT SET/RESET FUNCTIONS SENT TO HEADER CONTROL PORTS ***************

CONST PORTC0RESET = &H0                 'PORT C - BIT 0 - LOGIC LOW
CONST FUNCLOCKLOW = &H0                 'VME FUNCTION CLOCK LOW
CONST PORTC0SET = &H1                   'PORT C - BIT 0 - LOGIC HIGH
CONST FUNCLOCKHIGH = &H1                'VME FUNCTION CLOCK HIGH

CONST PORTC1RESET = &H2                 'PORT C - BIT 1 - LOGIC LOW
CONST FUNENABLE = &H2                   'VME FUNCTION ENABLE LOW (ENABLED)
CONST PORTC1SET = &H3                   'PORT C - BIT 1 - LOGIC HIGH
CONST FUNDISABLE = &H3                  'VME FUNCTION ENABLE HIGH (DISABLED)

CONST PORTC2RESET = &H4                 'PORT C - BIT 2 - LOGIC LOW
CONST SOFTRESETLOW = &H4                'SOFT RESET LOW
CONST PORTC2SET = &H5                   'PORT C - BIT 2 - LOGIC HIGH
CONST SOFTRESETHIGH = &H5               'SOFT RESET HIGH

CONST PORTC4RESET = &H8                 'PORT C - BIT 4 - LOGIC LOW
CONST INTERRUPTOFF = &H8                'INTERRUPT ENABLE LOW (DISABLED AT 8255)
CONST PORTC4SET = &H9                   'PORT C - BIT 4 - LOGIC HIGH
CONST INTERRUPTON = &H9                 'INTERRUPT ENABLE HIGH (ENABLED AT 8255)

'***** THE VME FUNCTIONS (EPROM ADDRESSES) OUTPUT TO VME BUS *****************

CONST NULL = &H0                        'EVERYTHING DISABLED - VME BUS ISOLATED

CONST VRAMWRSETUP = &H1                 'VALID RAM WRITE SET-UP
CONST VRAMWRITE = &H2                   'VALID RAM WRITE
CONST VRAMREAD = &H3                    'VALID RAM READ

CONST E0RAMWRSETUP = &H4                'ENERGY 0 (WINDOW) RAM WRITE SET-UP
CONST E0RAMWRITE = &H5                  'ENERGY 0 (WINDOW) RAM WRITE
CONST E0RAMREAD = &H6                   'ENERGY 0 (WINDOW) RAM READ

CONST E1RAMWRSETUP = &H7                'ENERGY 1 (WINDOW) RAM WRITE SET-UP
CONST E1RAMWRITE = &H8                  'ENERGY 1 (WINDOW) RAM WRITE
CONST E1RAMREAD = &H9                      'ENERGY 1 (WINDOW) RAM READ

CONST E2RAMWRSETUP = &HA                   'ENERGY 2 (WINDOW) RAM WRITE SET-UP
CONST E2RAMWRITE = &HB                     'ENERGY 2 (WINDOW) RAM WRITE
CONST E2RAMREAD = &HC                      'ENERGY 2 (WINDOW) RAM READ

CONST E3RAMWRSETUP = &HD                   'ENERGY 3 (WINDOW) RAM WRITE SET-UP
CONST E3RAMWRITE = &HE                     'ENERGY 3 (WINDOW) RAM WRITE
CONST E3RAMREAD = &HF                      'ENERGY 3 (WINDOW) RAM READ

CONST DATARUN = &H10                       'SETS UP VME BOARDS FOR RUN MODE

CONST ADCRAMCH0 = &H20                     'ENERGY BOARD 0 - ADC AND RAM CHECK
CONST ADCRAMCH1 = &H21                     'ENERGY BOARD 1 - ADC AND RAM CHECK
CONST ADCRAMCH2 = &H22                     'ENERGY BOARD 2 - ADC AND RAM CHECK
CONST ADCRAMCH3 = &H23                     'ENERGY BOARD 3 - ADC AND RAM CHECK

CONST CALIBRATE0 = &H80                    'ENERGY BOARD 0 - ADC CALIBRATE
CONST CALIBRATE1 = &HA0                    'ENERGY BOARD 1 - ADC CALIBRATE
CONST CALIBRATE2 = &HC0                    'ENERGY BOARD 2 - ADC CALIBRATE
CONST CALIBRATE3 = &HE0                    'ENERGY BOARD 3 - ADC CALIBRATE

'***********************************************************

READENERGY                                 'EXECUTE READENERGY SUBROUTINE
READVALID                                  'EXECUTE READVALID SUBROUTINE
SETRUNMODE                                 'EXECUTE SETRUNMODE SUBROUTINE

'***********************************************************

END                                        'END OF MAIN PROGRAM

'------------------------------------------------
SUB READENERGY

  DIM E0RAM(0 TO 255) AS INTEGER           'HOLDS WINDOW RAM 0 CONTENTS
  DIM E1RAM(0 TO 255) AS INTEGER           'HOLDS WINDOW RAM 1 CONTENTS
  DIM E2RAM(0 TO 255) AS INTEGER           'HOLDS WINDOW RAM 2 CONTENTS
  DIM E3RAM(0 TO 255) AS INTEGER           'HOLDS WINDOW RAM 3 CONTENTS

  CLS                                      'CLEAR THE SCREEN

  PRINT "PLEASE WAIT WHILE THE WINDOWS ARE CHECKED AND DISPLAYED!"
  PRINT                                    'PRINT MESSAGE, SKIP A LINE

  OUT ADDATACONTROL, READRAM               'SET UP ADD/DATA HEADER FOR RAM READ

'*****************************

  OUT FUNKTION, NULL                       'OUTPUT THE NULL FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW              '
OUT FUNCONTROL, FUNCLOCKHIGH            'CLOCK THE FUNCTION INTO THE LATCHES

OUT VMEDATA, 0                          'SET VME DATA BUS TO 0
OUT VMEADDLOW, 0                        'SET VME ADDRESS (LOW BITS) TO 0
OUT VMEADDHIGH, 0                       'SET VME ADDRESS (HIGH BITS) TO 0

FOR J = 0 TO 255                        'LOOP THROUGH THE WINDOW RAM

 OUT FUNKTION, NULL                     'OUTPUT THE NULL FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW            '
 OUT FUNCONTROL, FUNCLOCKHIGH           'CLOCK THE FUNCTION INTO THE LATCHES

 OUT VMEADDLOW, J                       'OUTPUT THE LOW ADDRESS BITS

 OUT FUNKTION, E0RAMREAD                'OUTPUT THE READ FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW            '
 OUT FUNCONTROL, FUNCLOCKHIGH           'CLOCK THE FUNCTION INTO THE LATCHES

 E0RAM(J) = INP(VMEDATA)                'READ THE DATA FROM THE PORT

 IF J <> 0 THEN                         'CHECK ALL LOCATIONS BUT THE 0TH

  IF E0RAM(J) = 1 AND E0RAM(J - 1) <> 1 THEN W1L = J        'WINDOW 1 LEFT TEST

  IF E0RAM(J) <> 1 AND E0RAM(J - 1) = 1 THEN W1R = J        'WINDOW 1 RIGHT TEST

  IF E0RAM(J) = 2 AND E0RAM(J - 1) <> 2 THEN W2L = J        'WINDOW 2 LEFT TEST

  IF E0RAM(J) <> 2 AND E0RAM(J - 1) = 2 THEN W2R = J        'WINDOW 2 RIGHT TEST

  IF E0RAM(J) = 3 AND E0RAM(J - 1) <> 3 THEN W3L = J        'WINDOW 3 LEFT TEST

  IF E0RAM(J) <> 3 AND E0RAM(J - 1) = 3 THEN W3R = J        'WINDOW 3 RIGHT TEST

 END IF                                 'END OF TEST

NEXT J                                  'END OF LOOP

'*****************************

FOR J = 0 TO 255                        'LOOP THROUGH THE WINDOW RAM

 X% = E0RAM(J)                          'USE X VARIABLE FOR TESTING

 IF ((X <> 0) AND (X <> 1) AND (X <> 2) AND (X <> 3)) THEN   'INVALID CONTENTS

  PRINT "ERROR, J= "; J, " E0RAM(J)= "; E0RAM(J)            'PRINT MESSAGE

 END IF                                 'END OF TEST
```

```
NEXT J                                              'END OF LOOP

IF (W1R = 0 OR W2R = 0 OR W3R = 0) THEN 'TEST IF WINDOWS HAVE BEEN SET

  PRINT "CHANNEL # 0 WINDOWS HAVE NOT BEEN SET!"              'PRINT MESSAGE
  ERAMOK = FALSE                                              'SET FLAG

ELSE                                                'CHANNEL 0 WINDOWS WERE SET

  PRINT "FOR ENERGY CHANNEL #0:"                    'PRINT WINDOWS FOR CHANNEL 0

  PRINT "START WINDOW IS SET FROM      "; W1L; " TO "; W1R    'PRINT RANGE

  PRINT "LOW STOP WINDOW IS SET FROM   "; W2L; " TO "; W2R    'PRINT RANGE

  PRINT "HIGH STOP WINDOW IS SET FROM  "; W3L; " TO "; W3R    'PRINT RANGE

 END IF                                             'END OF PRINTING

PRINT                                               'SKIP A LINE

'*****************************

OUT FUNKTION, NULL                                  'OUTPUT THE NULL FUNCTION

OUT FUNCONTROL, FUNCLOCKLOW                         '
OUT FUNCONTROL, FUNCLOCKHIGH                        'CLOCK THE FUNCTION INTO THE LATCHES

OUT VMEDATA, 0                                      'SET VME DATA BUS TO 0
OUT VMEADDLOW, 0                                    'SET VME ADDRESS (LOW BITS) TO 0
OUT VMEADDHIGH, 0                                   'SET VME ADDRESS (HIGH BITS) TO 0

FOR J = 0 TO 255                                    'LOOP THROUGH THE WINDOW RAM

  OUT FUNKTION, NULL                                'OUTPUT THE NULL FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW                       '
  OUT FUNCONTROL, FUNCLOCKHIGH                      'CLOCK THE FUNCTION INTO THE LATCHES

  OUT VMEADDLOW, J                                  'OUTPUT THE LOW ADDRESS BITS

  OUT FUNKTION, E1RAMREAD                           'OUTPUT THE READ FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW                       '
  OUT FUNCONTROL, FUNCLOCKHIGH                      'CLOCK THE FUNCTION INTO THE LATCHES

  E1RAM(J) = INP(VMEDATA)                           'READ THE DATA FROM THE PORT

  IF J <> 0 THEN                                    'CHECK ALL LOCATIONS BUT THE 0TH

   IF E1RAM(J) = 1 AND E1RAM(J - 1) <> 1 THEN W1L = J         'WINDOW 1 LEFT TEST
```

```
      IF E1RAM(J) <> 1 AND E1RAM(J - 1) = 1 THEN W1R = J          'WINDOW 1 RIGHT TEST

      IF E1RAM(J) = 2 AND E1RAM(J - 1) <> 2 THEN W2L = J          'WINDOW 2 LEFT TEST

      IF E1RAM(J) <> 2 AND E1RAM(J - 1) = 2 THEN W2R = J          'WINDOW 2 RIGHT TEST

      IF E1RAM(J) = 3 AND E1RAM(J - 1) <> 3 THEN W3L = J          'WINDOW 3 LEFT TEST

      IF E1RAM(J) <> 3 AND E1RAM(J - 1) = 3 THEN W3R = J          'WINDOW 3 RIGHT TEST

     END IF                                   'END OF TEST

    NEXT J                                    'END OF LOOP

'******************************

    FOR J = 0 TO 255                          'LOOP THROUGH THE WINDOW RAM

     X% = E1RAM(J)                            'USE X VARIABLE FOR TESTING

     IF ((X <> 0) AND (X <> 1) AND (X <> 2) AND (X <> 3)) THEN       '

      PRINT "ERROR, J= "; J, " E1RAM(J)= "; E1RAM(J)                 'PRINT MESSAGE

     END IF                                   'END OF TEST

    NEXT J                                    'END OF LOOP

    IF (W1R = 0 OR W2R = 0 OR W3R = 0) THEN 'TEST IF WINDOWS HAVE BEEN SET

     PRINT "CHANNEL # 1 WINDOWS HAVE NOT BEEN SET!"                  'PRINT MESSAGE
     ERAMOK = FALSE                                                  '

    ELSE                                      'CHANNEL 1 WINDOWS WERE SET

     PRINT "FOR ENERGY CHANNEL #1:"           'PRINT WINDOWS FOR CHANNEL 1

     PRINT "START WINDOW IS SET FROM       "; W1L; " TO "; W1R     'PRINT RANGE

     PRINT "LOW STOP WINDOW IS SET FROM    "; W2L; " TO "; W2R     'PRINT RANGE

     PRINT "HIGH STOP WINDOW IS SET FROM   "; W3L; " TO "; W3R     'PRINT RANGE

    END IF                                    'END OF PRINTING

    PRINT                                     'SKIP A LINE

'******************************

    OUT FUNKTION, NULL                        'OUTPUT THE NULL FUNCTION

    OUT FUNCONTROL, FUNCLOCKLOW               '
OUT FUNCONTROL, FUNCLOCKHIGH          'CLOCK THE FUNCTION INTO THE LATCHES

OUT VMEDATA, 0                        'SET VME DATA BUS TO 0
OUT VMEADDLOW, 0                      'SET VME ADDRESS (LOW BITS) TO 0
OUT VMEADDHIGH, 0                     'SET VME ADDRESS (HIGH BITS) TO 0

FOR J = 0 TO 255                      'LOOP THROUGH THE WINDOW RAM

 OUT FUNKTION, NULL                   'OUTPUT THE NULL FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW          '
 OUT FUNCONTROL, FUNCLOCKHIGH         'CLOCK THE FUNCTION INTO THE LATCHES

 OUT VMEADDLOW, J                     'OUTPUT THE LOW ADDRESS BITS

 OUT FUNKTION, E2RAMREAD              'OUTPUT THE READ FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW          '
 OUT FUNCONTROL, FUNCLOCKHIGH         'CLOCK THE FUNCTION INTO THE LATCHES

 E2RAM(J) = INP(VMEDATA)              'READ THE DATA FROM THE PORT

 IF J <> 0 THEN                       'CHECK ALL LOCATIONS BUT THE 0TH

  IF E2RAM(J) = 1 AND E2RAM(J - 1) <> 1 THEN W1L = J        'WINDOW 1 LEFT TEST

  IF E2RAM(J) <> 1 AND E2RAM(J - 1) = 1 THEN W1R = J        'WINDOW 1 RIGHT TEST

  IF E2RAM(J) = 2 AND E2RAM(J - 1) <> 2 THEN W2L = J        'WINDOW 2 LEFT TEST

  IF E2RAM(J) <> 2 AND E2RAM(J - 1) = 2 THEN W2R = J        'WINDOW 2 RIGHT TEST

  IF E2RAM(J) = 3 AND E2RAM(J - 1) <> 3 THEN W3L = J        'WINDOW 3 LEFT TEST

  IF E2RAM(J) <> 3 AND E2RAM(J - 1) = 3 THEN W3R = J        'WINDOW 3 RIGHT TEST

 END IF                               'END OF TEST

NEXT J                                'END OF LOOP

'****************************

FOR J = 0 TO 255                      'LOOP THROUGH THE WINDOW RAM

 X% = E2RAM(J)                        'USE X VARIABLE FOR TESTING

 IF ((X <> 0) AND (X <> 1) AND (X <> 2) AND (X <> 3)) THEN        '

  PRINT "ERROR, J= "; J, " E2RAM(J)= "; E2RAM(J)        'PRINT MESSAGE

 END IF                               'END OF TEST
```

```
NEXT J                                      'END OF LOOP

IF (W1R = 0 OR W2R = 0 OR W3R = 0) THEN 'TEST IF WINDOWS HAVE BEEN SET

  PRINT "CHANNEL # 2 WINDOWS HAVE NOT BEEN SET!"         'PRINT MESSAGE
  ERAMOK = FALSE                                         '

ELSE                                        'CHANNEL 2 WINDOWS WERE SET

  PRINT "FOR ENERGY CHANNEL #2:"            'PRINT WINDOWS FOR CHANNEL 2

  PRINT "START WINDOW IS SET FROM     "; W1L; " TO "; W1R    'PRINT RANGE

  PRINT "LOW STOP WINDOW IS SET FROM  "; W2L; " TO "; W2R    'PRINT RANGE

  PRINT "HIGH STOP WINDOW IS SET FROM  "; W3L; " TO "; W3R   'PRINT RANGE

END IF                                      'END OF PRINTING

PRINT                                       'SKIP A LINE

'*****************************

OUT FUNKTION, NULL                          'OUTPUT THE NULL FUNCTION

OUT FUNCONTROL, FUNCLOCKLOW                 '
OUT FUNCONTROL, FUNCLOCKHIGH                'CLOCK THE FUNCTION INTO THE LATCHES

OUT VMEDATA, 0                              'SET VME DATA BUS TO 0
OUT VMEADDLOW, 0                            'SET VME ADDRESS (LOW BITS) TO 0
OUT VMEADDHIGH, 0                           'SET VME ADDRESS (HIGH BITS) TO 0

OUT ADDATCONTROL, READRAM                   'SET UP PORTA2 (DATA) AS INPUT

FOR J = 0 TO 255                            'LOOP THROUGH THE WINDOW RAM

  OUT FUNKTION, NULL                        'OUTPUT THE NULL FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW               '
  OUT FUNCONTROL, FUNCLOCKHIGH              'CLOCK THE FUNCTION INTO THE LATCHES

  OUT VMEADDLOW, J                          'OUTPUT THE LOW ADDRESS BITS

  OUT FUNKTION, E3RAMREAD                   'OUTPUT THE READ FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW               '
  OUT FUNCONTROL, FUNCLOCKHIGH              'CLOCK THE FUNCTION INTO THE LATCHES

  E3RAM(J) = INP(VMEDATA)                   'READ THE DATA FROM THE PORT

  IF J <> 0 THEN                            'CHECK ALL LOCATIONS BUT THE 0TH
```

```
      IF E3RAM(J) = 1 AND E3RAM(J - 1) <> 1 THEN W1L = J        'WINDOW 1 LEFT TEST
      IF E3RAM(J) <> 1 AND E3RAM(J - 1) = 1 THEN W1R = J        'WINDOW 1 RIGHT TEST
      IF E3RAM(J) = 2 AND E3RAM(J - 1) <> 2 THEN W2L = J        'WINDOW 2 LEFT TEST
      IF E3RAM(J) <> 2 AND E3RAM(J - 1) = 2 THEN W2R = J        'WINDOW 2 RIGHT TEST
      IF E3RAM(J) = 3 AND E3RAM(J - 1) <> 3 THEN W3L = J        'WINDOW 3 LEFT TEST
      IF E3RAM(J) <> 3 AND E3RAM(J - 1) = 3 THEN W3R = J        'WINDOW 3 RIGHT TEST
    END IF                                   'END OF TEST
  NEXT J                                     'END OF LOOP
'*****************************
  FOR J = 0 TO 255                           'LOOP THROUGH THE WINDOW RAM
    X% = E3RAM(J)                            'USE X VARIABLE FOR TESTING
    IF ((X <> 0) AND (X <> 1) AND (X <> 2) AND (X <> 3)) THEN     '
      PRINT "ERROR,  J= "; J, " E3RAM(J)= "; E3RAM(J)        'PRINT MESSAGE
    END IF                                   'END OF TEST
  NEXT J                                     'END OF LOOP
  IF (W1R = 0 OR W2R = 0 OR W3R = 0) THEN 'TEST IF WINDOWS HAVE BEEN SET
    PRINT "CHANNEL # 3 WINDOWS HAVE NOT BEEN SET!"          'PRINT MESSAGE
    ERAMOK = FALSE                                          '
  ELSE                                       'CHANNEL 3 WINDOWS WERE SET
    PRINT "FOR ENERGY CHANNEL #3:"           'PRINT WINDOWS FOR CHANNEL 3
    PRINT "START WINDOW IS SET FROM      "; W1L; " TO "; W1R   'PRINT RANGE
    PRINT "LOW STOP WINDOW IS SET FROM   "; W2L; " TO "; W2R   'PRINT RANGE
    PRINT "HIGH STOP WINDOW IS SET FROM  "; W3L; " TO "; W3R   'PRINT RANGE
  END IF                                     'END OF PRINTING
  PRINT                                      'SKIP A LINE
'*****************************
  OUT FUNKTION, NULL                         'OUTPUT THE NULL FUNCTION
  OUT FUNCONTROL, FUNCLOCKLOW         '
  OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

  IF ERAMOK = TRUE THEN               'TEST TO SEE IF ALL THE WINDOWS ARE SET
   PRINT "ALL THREE WINDOWS ARE SET"  'PRINT MESSAGE

  ELSE                                'ERROR DETECTED
   PRINT "AT LEAST ONE CHANNEL HAS NO WINDOWS SET!"     'PRINT MESSAGE

  END IF                              'END OF TEST

  PRINT                               'SKIP A LINE

  PRINT "-PRESS ANY KEY TO CHECK THE VALID RAM-"        'PRINT MESSAGE

  DO                                  '
   ANS$ = INKEY$                      'LOOP WHILE NO KEY IS PRESSED
  LOOP WHILE ANS$ = ""                '

END SUB                               'END OF READENERGY SUBROUTINE

'----------------------------------------------
SUB READVALID

  DIM VRAM(0 TO 15, 0 TO 255) AS INTEGER 'HOLDS CONTENTS OF VALID RAM

  VRAMOK = TRUE                       'INITIALIZE VRAMOK TO TRUE

  CLS                                 'CLEAR SCREEN, PRINT MESSAGE

  PRINT "PLEASE WAIT WHILE VALID RAM IS CHECKED FOR CORRECTNESS!"

'*****************************

  OUT FUNKTION, NULL                  'OUTPUT THE NULL FUNCTION

  OUT FUNCONTROL, FUNCLOCKLOW         '
  OUT FUNCONTROL, FUNCLOCKHIGH        'CLOCK THE FUNCTION INTO THE LATCHES

  OUT VMEDATA, 0                      'SET VME DATA BUS TO 0
  OUT VMEADDLOW, 0                    'SET VME ADDRESS (LOW BITS) TO 0
  OUT VMEADDHIGH, 0                   'SET VME ADDRESS (HIGH BITS) TO 0

  FOR I = 0 TO 15                     'OUTER LOOP FOR HIGH ADDRESS BITS

   FOR J = 0 TO 255                   'INNER LOOP FOR LOW ADDRESS BITS

    OUT FUNKTION, NULL                'OUTPUT THE NULL FUNCTION

    OUT FUNCONTROL, FUNCLOCKLOW       '
    OUT FUNCONTROL, FUNCLOCKHIGH      'CLOCK THE FUNCTION INTO THE LATCHES
    OUT VMEADDHIGH, I                      'OUTPUT THE HIGH ADDRESS BITS
    OUT VMEADDLOW, J                       'OUTPUT THE LOW ADDRESS BITS

    OUT FUNKTION, VRAMREAD                 'OUTPUT THE READ FUNCTION

    OUT FUNCONTROL, FUNCLOCKLOW            '
    OUT FUNCONTROL, FUNCLOCKHIGH           'CLOCK THE FUNCTION INTO THE LATCHES

    VRAM(I, J) = INP(VMEDATA)              'READ THE DATA FROM THE PORT

   NEXT J                                  'END OF INNER LOOP

  NEXT I                                   'END OF OUTER LOOP

'****************************

  FOR I = 0 TO 15                          'OUTER LOOP FOR HIGH ADDRESS BITS

   FOR J = 0 TO 255                        'INNER LOOP FOR LOW ADDRESS BITS

    X% = VRAM(I, J)                        'USE X VARIABLE FOR TESTING

     IF X < 255 AND X > 23 THEN            'CHECK TO SEE IF CONTENTS IN VALID RANGE

      PRINT "I= "; I, " J= "; J, " VRAM(I,J)= "; VRAM(I, J)        'INVALID CONTENTS

      PRINT "VALID RAM IS IN ERROR!"                               'PRINT MESSAGE

     END IF                                'END OF TEST

   NEXT J                                  'END OF INNER LOOP

  NEXT I                                   'END OF OUTER LOOP

  IF VRAM(0, 10) <> 6 THEN                 'TEST FOR EVENT 0 IN VALID RAM
   VRAMOK = FALSE                          'ERROR DETECTED
   PRINT "EVENT 0 IS NOT CORRECT!"         'PRINT MESSAGE
  END IF                                   'END OF TEST

  IF VRAM(0, 11) <> 9 THEN                 'TEST FOR EVENT 0 IN VALID RAM
   VRAMOK = FALSE                          'ERROR DETECTED
   PRINT "EVENT 1 IS NOT CORRECT!"         'PRINT MESSAGE
  END IF                                   'END OF TEST

  IF VRAM(0, 17) <> 0 THEN                 'TEST FOR EVENT 0 IN VALID RAM
   VRAMOK = FALSE                          'ERROR DETECTED
   PRINT "EVENT 2 IS NOT CORRECT!"         'PRINT MESSAGE
  END IF                                   'END OF TEST

  IF VRAM(0, 25) <> 3 THEN                 'TEST FOR EVENT 0 IN VALID RAM
   VRAMOK = FALSE                          'ERROR DETECTED
   PRINT "EVENT 3 IS NOT CORRECT!"         'PRINT MESSAGE
```

```
END IF                                  'END OF TEST

IF VRAM(0, 66) <> 12 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 4 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 67) <> 15 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 5 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 80) <> 13 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 6 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 88) <> 16 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 7 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 129) <> 1 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 8 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 136) <> 7 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 9 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 193) <> 4 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 10 IS NOT CORRECT!"      'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(0, 200) <> 10 THEN              'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 11 IS NOT CORRECT!"      'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(2, 2) <> 18 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 12 IS NOT CORRECT!"      'PRINT MESSAGE
END IF                                  'END OF TEST

IF VRAM(2, 3) <> 21 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                        'ERROR DETECTED
  PRINT "EVENT 13 IS NOT CORRECT!"      'PRINT MESSAGE
END IF                                  'END OF TEST
```

```
IF VRAM(2, 16) <> 19 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 14 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(2, 24) <> 22 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 15 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(2, 128) <> 20 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 16 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(2, 192) <> 23 THEN               'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 17 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(4, 1) <> 2 THEN                  'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 18 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(4, 8) <> 8 THEN                  'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 19 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(4, 64) <> 14 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 20 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(6, 1) <> 5 THEN                  'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 21 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(6, 8) <> 11 THEN                 'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 22 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

IF VRAM(6, 64) <> 17 THEN                'TEST FOR EVENT 0 IN VALID RAM
  VRAMOK = FALSE                         'ERROR DETECTED
  PRINT "EVENT 23 IS NOT CORRECT!"       'PRINT MESSAGE
END IF                                   'END OF TEST

'*****************************
```

```
 OUT FUNKTION, NULL                          'OUTPUT THE NULL FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW                 '
 OUT FUNCONTROL, FUNCLOCKHIGH                'CLOCK THE FUNCTION INTO THE LATCHES

 IF VRAMOK = TRUE THEN                       'CHECK ALL OF THE VALID RAM TESTS
  PRINT                                      'SKIP A LINE
  PRINT "ALL IS WELL IN THE VALID RAM!"                      'PRINT MESSAGE
  PRINT                                      'SKIP A LINE

 ELSE                                        'AN ERROR WAS DETECTED
  PRINT                                      'SKIP A LINE
  PRINT "THERE IS A PROBLEM WITH THE VALID RAM!"    'PRINT MESSAGE
  PRINT                                      'SKIP A LINE

 END IF                                      'END OF TEST

 PRINT "-PRESS ANY KEY TO CONTINUE-"         'PRINT MESSAGE

 DO                                          '
  ANS$ = INKEY$                              'LOOP WHILE NO KEY IS PRESSED
 LOOP WHILE ANS$ = ""                        '

END SUB                                      'END OF READVALID SUBROUTINE

'----------------------------------------------
SUB SETRUNMODE

 OUT FUNCONTROL, INTERRUPTOFF                'TURN OFF THE INTERRUPTS

 OUT FUNKTION, NULL                          'SEND OUT THE NULL FUNCTION
                                             '
 OUT FUNCONTROL, FUNCLOCKLOW                 '
 OUT FUNCONTROL, FUNCLOCKHIGH                'CLOCK THE FUNCTION INTO THE LATCHES

 OUT ADDATACONTROL, RUNMODE                  'SET UP THE ADDRESS/DATA HEADER FOR RUN

 OUT FUNKTION, DATARUN                       'SEND OUT THE RUN FUNCTION

 OUT FUNCONTROL, FUNCLOCKLOW                 '
 OUT FUNCONTROL, FUNCLOCKHIGH                'CLOCK THE FUNCTION INTO THE LATCHES

END SUB                                      'END OF SETRUNMODE SUBROUTINE
```