ISBN 0-315-55527-0

THE UNIVERSITY OF ALBERTA

A HIGH ACCURACY DIGITAL SAMPLING WATTMETER FOR DISTORTED AND VARYING FREQUENCY SIGNALS

by

DAVID JECTY NYARKO

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1989

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	DAVID JECTY NYARKO
TITLE OF THESIS:	A HIGH ACCURACY DIGITAL SAMPLING WATTMETER FOR DISTORTED AND VARYING FREQUENCY SIGNALS
DEGREE:	MASTER OF SCIENCE
YEAR THIS DEGREE GRANTED:	FALL, 1989

....

(Student's signature)

Student's permanent address:

2B 8916 - 112 Street

EDMONTON, ALBERTA

T6G 2C5

Date: 31st August 1989

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled A HIGH ACCURACY DIGITAL SAMPLING WATTMETER FOR DISTORTED AND VARYING FREQUENCY SIGNALS submitted by DAVID JECTY NYARKO in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.

....

(Supervisor)

....

....

Date: ..21..Aug..1989....

ABSTRACT

This thesis describes an approach to the implementation of digital sampling wattmeters. A numerical integration algorithm employing a trapezoidal rule with an end correction, is shown to considerably improve the accuracy of such instruments. This results in a more precise measurement of distorted and varying frequency waveforms. This approach is employed in the design of a digital sampling wattmeter design primarily intended for power system measurements.

The stand-alone system based on an 8Mhz 68000 microprocessor system, is capable of measuring distorted power signals with a fundamental frequency from dc to 1khz. Auxiliary functions in this GPIB compatible and keypad programmable instrument include voltage, current , power factor and frequency measurements in various modes.

ACKNOWLEDGEMENT

I would like to thank my sponsors, the Association of Universities and Colleges of Canada, for their financial support for my MSc. program. I would also like to thank my supervisor, Dr. Keith Stromsmoe, for his help and encouragement in preparing this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Description
α_k	The kth harmonic sine term error coefficient
β_k	The kth harmonic cosine term error coefficient
f	Frequency
i	Current
j	Sample number
k	Harmonic number
I_{rms}	Root-mean-square current
h	Sampling interval (angular measurement in radians)
P_{ac}	Active electrical power (AC)
P_{dc}	Electrical power (DC)
t	Time
T	Period
v	Voltage
V_{rms}	Root-mean-square voltage

1. INTRODUCTION

The widespread development and use of electric power systems with highly distorted waveforms has rendered the use of the relatively commonplace instruments, particularly the analog type, inappropriate. Basically, two classes of instruments exist [5], based on the computational method employed; the analog and the digital types. Hybrid instruments employing both techniques are also in existence.

Analog instruments although relatively inexpensive with regard to their digital counterparts, suffer from numerous deficiencies. The analog computation approach suffers from amplifier gain bandwidth limitations, matching of analog components as well as a low dynamic range and sensitivity. Furthermore, calibration is not easily carried out.

The alternative digital approach, used in digital sampling meters, entails sampling the input voltage and current signals and employing fully digital arithmetic procedures to compute the required parameters. This results in very precise results. It is also easily effected in digital stored-program systems such as microprocessor-based implementations.

Two distinct approaches to wattmeter sampling timing exist. In the synchronous approach, the sample timing and the waveform frequency are synchronised so that a block of samples represents an integral number of periods.

Alternatively, an asynchronous technique is employed, in which sampling is at a fixed rate independent of the waveform frequency. The first-mentioned approach has been widely employed and analyzed [1],[10],[13]. This approach, although easy to implement, results in accurate meters only if the signal frequency under consideration is stable. The computation of electrical power, voltage and current on systems operated from the electrical mains require an asynchronus technique for an adequate degree of accuracy. This approach is employed in the National Bureau of Standards calibration wattmeters [12]. Since the blocks of samples processed in this approach do not generally represent an integer number of periods, an allowance must be made for the ends of the block. The approach utilized, the modified trapezoidal approach, unlike other processing methods employed, uses all the samples obtained, in computing the required integral.

The device was developed to meet the following objectives.

1) Peak input voltage, current and maximum power of 350v, 10A and 3500W respectively.
2) Input signal frequency from dc to 1khz.
3) Frequency measurement range; 12 - 1000hz.
4) A user-friendly software interface.
5) An easy to maintain hardware.

2 THEORY

The determination of electrical power, voltage and current entails the computation of appropriately scaled integrals. Digital sampling meters utilize values from a discrete set approximating continuous signal values. The application of numerical integration techniques to the sample set results in an approximation to the signal parameters desired.

The fundamental equations for calculating the electrical dc power, the dc voltage, the dc current, the electrical ac power, and the root-mean-square voltage and current are respectively;

$$P_{dc} = \frac{1}{A}\int_0^A v\ i\ dt \qquad (2.1)$$

$$V_{dc} = \frac{1}{B}\int_0^B v\ dt \qquad (2.2)$$

$$I_{dc} = \frac{1}{C}\int_0^C i\ dt \qquad (2.3)$$

$$P_{ac} = \frac{1}{T}\int_0^T v\ i\ dt \qquad (2.4)$$

$$V_{rms} = \sqrt{\frac{1}{T}\int_0^T v^2\ dt} \qquad (2.5)$$

$$I_{rms} = \sqrt{\frac{1}{T}\int_0^T i^2\,dt} \qquad (2.6)$$

where A,B,C are arbitrary time intervals,

T is the period

v is the instantaneous voltage

and i is the instantaneous current

Equations 2.1 to 2.6 involve parameters which are functions of the average of integrals. In general, these equations can be represented as;

$$x = F[I\{y(t)\}] \qquad (2.7)$$

where x is the parameter required

y(t) is the corresponding variable

I is an averaging function dependent on the integral of y(t)

and the function F, represents any extra processing required to compute the parameter.

For example, relating the above equation to equation 2.5;

$$x(t) = V_{rms}$$

$$y(t) = v^2$$

$$I\{y(t)\} = \frac{1}{T}\int_0^T v^2\,dt$$

and F is the square-root function.

2.1 Analysis of sampled waveforms

A convenient approach to the analysis of the periodic signals encountered in the measurement of the system parameters is the Fourier series technique.

A Fourier series expansion for a periodic signal y(t) is:

$$y(t) = a_0 + \sum_{k=1}^{\infty} a_k \sin\left(\frac{2\pi kt}{T}\right) + \sum_{k=1}^{\infty} b_k \cos\left(\frac{2\pi kt}{T}\right) \quad (2.8)$$

where T = the fundamental period

t = time

Sampling the above signal at regularly spaced intervals of length h radians, produces sampled values y_j which can be expressed as:

$$y_j = a_0 + \sum_{k=1}^{\infty} a_k \sin\left(kjh\right) + \sum_{k=1}^{\infty} b_k \cos\left(kjh\right) \quad (2.9)$$

$j = 0,1,2,..,n$ for a signal sampled at n+1 points

The above representation of an ac signal can also be used to represent the power signal which exists as a result of the microprocessor computations on the sampled values of voltage and current. This is due to the simultaneous sampling of the sinusoidal voltage and current signals.

2.2 Error Determination

The basic principle involved in the determination of the root mean square values of the voltage and current as well as the average power, is the application of a linear averaging operator A to the product of sampled values v_j^2, i_j^2 and $v_j i_j$ respectively. This operator computes the mean-value of an integral. It is thus the discrete equivalent of the continuous data averaging function I in equation 2.7. The existence of numerous algorithms for computing the integral of a discrete data set, thus result in a variety of expressions for A. These expressions for A are referred to in this thesis, as the processing methods. The error in any processing method indicates the deviation of the parameters obtained using that method to that obtained by computing the mean value of the integral with continuous signals.

Two schemes for the error determination exist: time domain and frequency domain analysis. The former is suitable for synchronous sampling implementations but difficult to interpret for asynchronous sampling schemes. The latter approach [14] however covers both sampling schemes easily, and is easily interpreted. This is due to the fact that the error coefficients are related directly to the signal harmonics. This approach is used in the subsequent equations.

Let y represent the products of the instantaneous values of the relevant continuous signals, and y_j represent the products of the sampled values. The error **E** resulting from the application of the linear averaging operator **A** on y_j is:

$$E = Ay_j - \frac{1}{T}\int_0^T y(t)\,dt \qquad (2.10)$$

$$= A\sum_{k=1}^{\infty} a_k \sin\left(kjh\right) + A\sum_{k=1}^{\infty} b_k \cos\left(kjh\right) \qquad (2.11)$$

since $Aa_0 = a_0$

$$= \sum_{k=1}^{\infty} a_k\, A\sin\left(kjh\right) + \sum_{k=1}^{\infty} b_k\, A\cos\left(kjh\right) \qquad (2.12)$$

$$= \sum_{k=1}^{\infty} a_k \alpha_k + \sum_{k=1}^{\infty} b_k \beta_k \qquad (2.13)$$

The error coefficients α_k (alpha) and β_k (beta) are thus dependent on the processing method. The equation further indicates the dependence of the error on the Fourier series coefficients and hence the start of the sampling process.

For n+1 regularly spaced samples spanning n intervals of duration h, the following expression is obtained

$$(n + \Delta)h = 2m\pi \qquad (2.14)$$

where n,m are integers

and Δ (delta) is a dimensionless quantity.

2.3 Error expressions for various processing methods

Two distinct cases should be analyzed in the determination of the error expressions.

a) $kh = 2\pi p$ where p is a positive integer

b) $kh \neq 2\pi p$ where p is a positive integer

a) $\underline{kh = 2\pi p}$

In this case, a general error expression can be obtained for all processing methods which satisfy the relation ; $Aa_0 = a_0$.

$$\alpha_k = A \sin(jkh)$$
$$= A \sin(j2\pi p)$$
$$= 0$$

$$\beta_k = A \cos(jkh)$$

$$= A \cos(j2\pi p)$$
$$= 1$$

It should be noted that the error expressions obtained for α_k and β_k above, are applicable to a synchronous sampling scheme provided the following conditions are satisfied.

a) n+1 samples covering m periods

where m and n are coprime.

b) $k = p\left(\frac{n}{m}\right)$.

For example if 521 samples are taken in a period of a waveform, only the 520th ,1040th etc. harmonics will contribute to the error in measurement of the required parameter.

b) <u>$kh \neq 2\pi p$</u>

In this instance, the error is dependent on the processing scheme. Four existing processing schemes will be considered; the average of samples, the trapezoidal rule, the NBS approach and the processing method used in this thesis.

In the average of samples approach the integral is approximated by a series of rectangles of height equal to the sampled values. This approach does not take into account the last sampled value. No correction is made to account for cases in which the total sampling interval does not cover an integral number of cycles. It is however the most widely used approach. The trapezoidal rule approximates the integral using a series of trapezoids. All the samples are used in the computation of the integral. Like the average method, no correction is made for the sampling interval. The NBS method is simply an average method including an end correction Δ in the range [0-1]. Finally, the approach used in this thesis is simply a trapezoidal rule with an end correction. Unlike the NBS method, the end correction Δ is restricted to the range [-0.5,0.5] by adjusting the sample size accordingly. This method will be referred to as the modified trapezoidal approach.

2.3.1 Average of samples

The average value for n+1 samples [2],[6],[8] is given by;

$$Ay_j = \frac{1}{n}\sum_{j=0}^{n-1} y_j \qquad (2.15)$$

The corresponding error coefficient relating to the sine term of the error expression is:

$$\overset{A}{\alpha_k} = A\sin(jkh)$$

$$= A\left[\mathrm{Im}\left(e^{ijkh}\right)\right]$$

$$= A\left[\mathrm{Im}\left(e^{ij\theta}\right)\right]$$

where $\theta = kh$

$$= \frac{1}{n}\sum_{j=0}^{n-1}\mathrm{Im}\left(e^{ij\theta}\right)$$

$$= \frac{1}{n}\,\mathrm{Im}\left(\frac{1-e^{in\theta}}{1-e^{i\theta}}\right)$$

$$= \frac{1}{n}\,\mathrm{Im}\left[\left(\frac{1-\cos(n\theta)-i\sin(n\theta)}{1-\cos\theta-i\sin(n\theta)}\right)\left(\frac{1-\cos\theta+i\sin\theta}{1-\cos\theta+i\sin\theta}\right)\right]$$

$$= \frac{1}{n}\left[\frac{\sin\theta(1-\cos(n\theta))-\sin(n\theta)(1-\cos\theta)}{(1-\cos\theta)^2+\sin^2\theta}\right]$$

now $(n + \Delta)h = 2m\pi$

$\Rightarrow (n + \Delta)kh = 2mk\pi$

$nkh = 2mk\pi - \Delta kh$

$\Rightarrow \quad n\theta = 2mk\pi - \Delta kh$

$\Rightarrow \quad \sin(n\theta) = -\sin(\Delta kh)$

and $\cos(n\theta) = \cos(\Delta kh)$

hence;

$$\hat{\alpha}_k = \frac{1}{n}\left[\frac{\sin\theta(1-\cos(\Delta\theta)) + \sin(\Delta\theta)(1-\cos\theta)}{2(1-\cos\theta)}\right]$$

$$= \frac{1}{n}\left[\cot\left(\frac{k\ h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) + \frac{\sin(\Delta kh)}{2}\right]$$

The corresponding error coefficient relating to the cosine term of the error expression is:

$$\hat{\beta}_k = A\cos(jkh)$$

$$= A\left[\mathrm{Re}\left(e^{ijkh}\right)\right]$$

$$= A\left[\mathrm{Re}\left(e^{ij\theta}\right)\right] \qquad \text{where } \theta = kh$$

$$= \frac{1}{n}\sum_{j=0}^{n-1}\mathrm{Re}\left(e^{ij\theta}\right)$$

$$= \frac{1}{n}\mathrm{Re}\left(\frac{1 - e^{in\theta}}{1 - e^{i\theta}}\right)$$

$$= \frac{1}{n}\mathrm{Re}\left[\left(\frac{1-\cos(n\theta)-i\sin(n\theta)}{1-\cos\theta-i\sin(n\theta)}\right)\left(\frac{1-\cos\theta+i\sin\theta}{1-\cos\theta+i\sin\theta}\right)\right]$$

facility results in simple LCD display routines. The LCD unit also performs the refreshing of the display thus freeing up the microprocessor for other tasks. A further advantage in using this device is the possibility of writing to any of the 80 locations on the screen independently of the others. The 4 line LCD places a limitation on the number of items that can be displayed on the screen at a time. The bargraph display segments indicate the system as well as the program status. Enabling or disabling individual display segments is effected through the PIA. The detailed circuit diagram of this section is shown if Fig. A.11.

The GPIA circuitry provides the interface required to implement the IEEE-488 bus (GPIB) protocol. The heart of this subunit, the Motorola MC68488 GPIA IC, facilitates the implementation of the bus protocol. This circuitry together with the associated software enables the instrument to implement the following IEEE-488 standard features: talker, listener, complete source and acceptor handshake, serial and parallel poll, device trigger, device clear and the remote/local procedure. The detailed circuit diagram of this circuitry is shown in Fig. A.12.

3.3 Timing Controller module

The block diagram of this module is shown in Fig. 3.9 (page 53). This subunit comprises the conversion timing logic, the AC detection logic and the acquisition state logic. A more detailed block diagram showing the control signal interconnections is shown in the appendix as Fig. A.13.

The conversion timing logic produces the signals required for the analog-to-digital (A/D) converters and the sample-and-hold amplifiers. The use of Fairchild Advanced Schottky TTL (FAST) logic as well as the derivation of the various timing signals from the 8 Mhz system clock results in minimal jitter as well as precise timing signals. The A/D converter signals are the 1Mhz analog-to-digital (A/D) converters clock signal and the 2μs start of conversion signal. The primary sample-and-hold amplifier signal is that of the 32μs sampling interval. Some relevant timing signals generated by this logic and their relationship to the analog-to-digital interface circuitry is indicated in Fig. 3.10 (page 54).

The AC detection logic provides the automatic AC or DC mode detection required for the wattmeter operation.

The acquisition state logic provides the status signals employed by the software in determining the state of the current acquisition cycle. This circuit also determines the end correction required following an acquisition cycle involving an AC signal.

The detailed circuitry of this subunit is shown in Figs. A.14 and A.15.

3.4 Analog-to-Digital interface module

The block diagram of this module is shown in Fig. 3.11 (page 55). A more detailed block diagram showing the various signal interconnections is shown in the appendix as Fig. A.16.

The purpose of this section is to provide a digital equivalent of the analog input signal after the required input conditioning has been performed. This subunit comprises 4 similar channel interface sections representing each of the input channels. Each section has as its main devices, a voltage scaling and translation section, a sample-and-hold amplifier, a 12-bit analog-to-digital (A/D) converter, as well as an output latch for the temporary storage of the converted values. An input signal in the range -10v to +10v applied to the respective voltage scaling and translation section produces a 0-5v output. The latter voltage is applied to the corresponding sample-and-hold amplifier and subsequently converted to a 12-bit value in the respective A/D converter. The acquisition scheme employed, results in the sampling and conversion of signals on all 4 channels being performed simultaneously. The circuit logic is set up so that the results of the previous conversions are stored in the latches while the current conversion is taking place. This scheme results in 30μs of the 32μs sampling interval being available for use by the microprocessor.

The detailed circuit diagram of this subunit is shown if Figs. A.17 - A.20.

3.5 Input-conditioning circuitry

Fig. 3.12 (page 56) shows the block diagram of this module. The four sections making up this module are; the Phase 1 circuitry, the Phase 2 circuitry, the analog switches circuitry and the trigger circuitry. A more detailed block diagram showing the various signal interconnections is shown in the appendix as Fig. A.21.

3.5.1 Phase 1 circuitry

The block diagram of the Phase 1 circuitry is shown in Fig. 3.13 (page 56). This section consists of a voltage circuitry and a current circuitry. The voltage circuitry also referred to as the channel 1 circuitry, provides the interface between the digital wattmeter and the input voltage source. This circuitry is employed for single phase measurements or as the first phase in three-phase system measurements. Two input voltage ranges of 200V and 350V peak are provided. Initial signal attenuation is provided by the resistive voltage divider stage. Further buffering and amplification to the -10v to +10v required for the subsequent A/D interface stage is provided by the amplifier and buffer stage [4]. The current circuitry also referred to as the channel 2 circuitry, provides the interface between the digital wattmeter and the input current source. This circuitry is employed for single phase measurements or as the first phase in three-phase system measurements. A current shunt is employed for current measurement. The voltage generated across the shunt, is amplified by the subsequent amplifier and buffer stage circuitry to the -10v to +10v range required for the subsequent A/D interface stage. The detailed circuit diagram of this stage is shown in Fig. A.22.

3.5.2 Phase 2 circuitry

The block diagram of the Phase 2 circuitry is shown in Fig. 3.14 (page 57). This circuitry is identical to the Phase 1 circuitry and used in three-phase measurements or measurements of a second setup. This section comprises a voltage circuitry also referred to as the channel 3 circuitry and a current section referred to as the channel 4 circuitry. The detailed circuit diagram of this stage is shown in Fig. A.23.

3.5.3 Analog switches circuitry

The block diagram of this section is shown in Fig. 3.15 (page 57). This section consists of two main blocks; the trigger source switching circuitry and the system mode switch circuitry. Both switching circuits are controlled through the PIA. The trigger source input selects the output of either channel 1 or 2 as the trigger source. Filtering of the output of this stage is provided by the low-pass filter before subsequent processing. The system mode switch selects the input to the A/D interface stage. Two options are available for the above-mentioned stage. These are ; the calibration terminal or the input-conditioning subunit. The analog switches are controlled by the 6821 PIA IC. This approach enables the selection of the specified sources under software control. The detailed circuit diagram of this circuit is shown in Fig. A.24.

3.5.4 Trigger circuitry

Fig. 3.16 (page 58) shows the block diagram of this circuitry. This circuit is basically that of a variable threshold voltage level detector. The triggering level adjust resistor provides the input switching threshold voltage of the voltage comparator. The comparator includes logic to generate a TTL compatible output signal. The detailed circuitry for this section is shown if Fig. A.25.

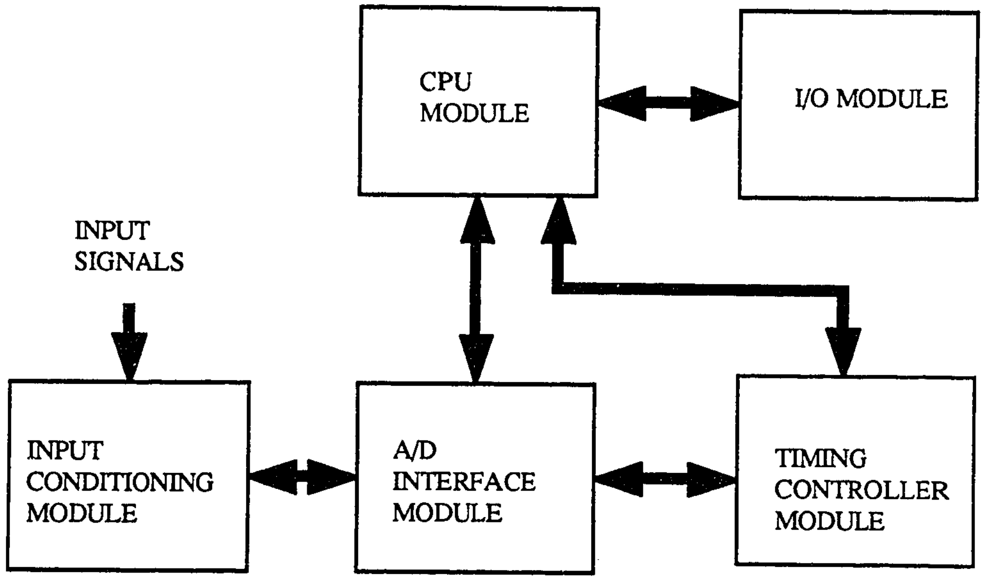

Fig. 3.1 Block diagram of wattmeter

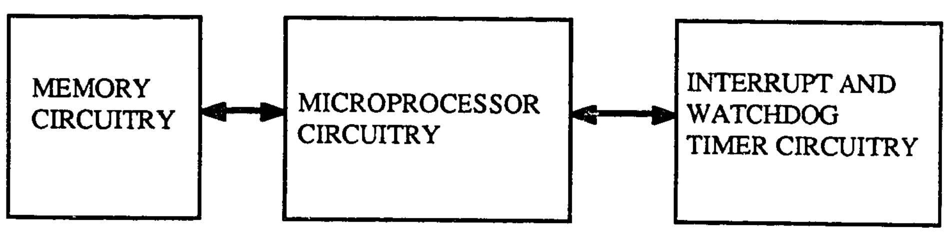

Fig. 3.2 Block diagram of CPU module

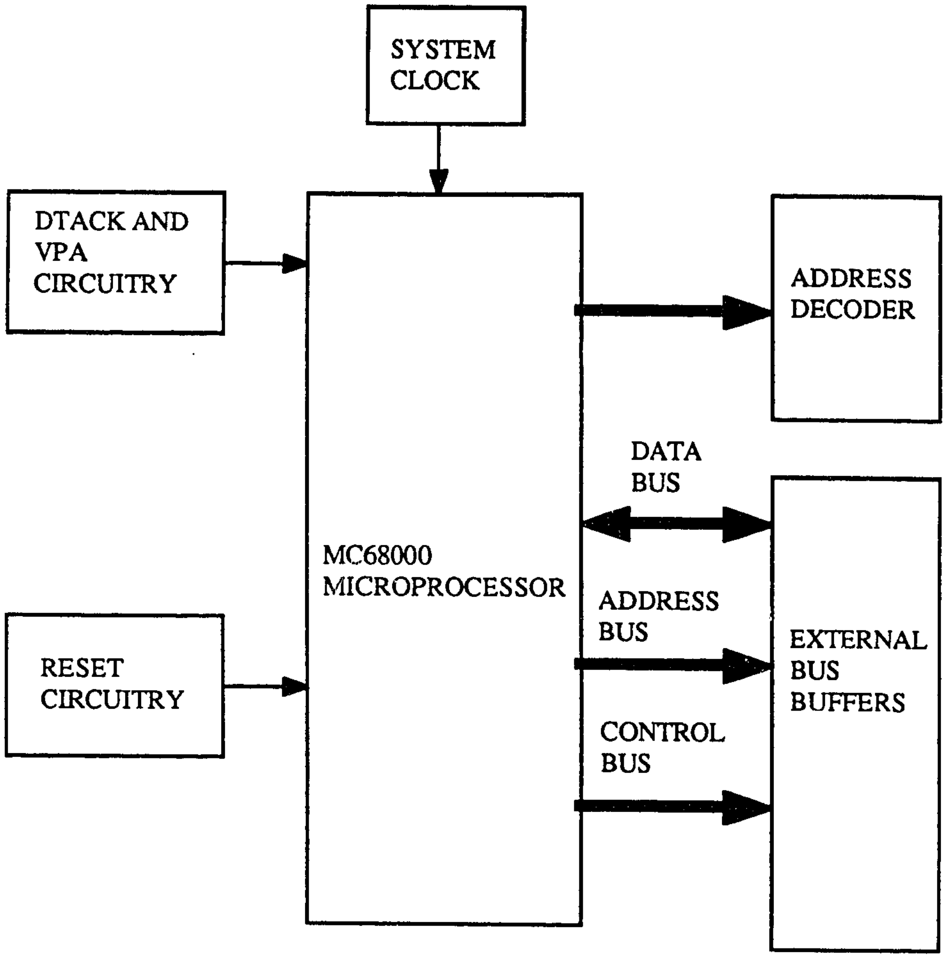

Fig. 3.3 Block diagram of microprocessor circuitry

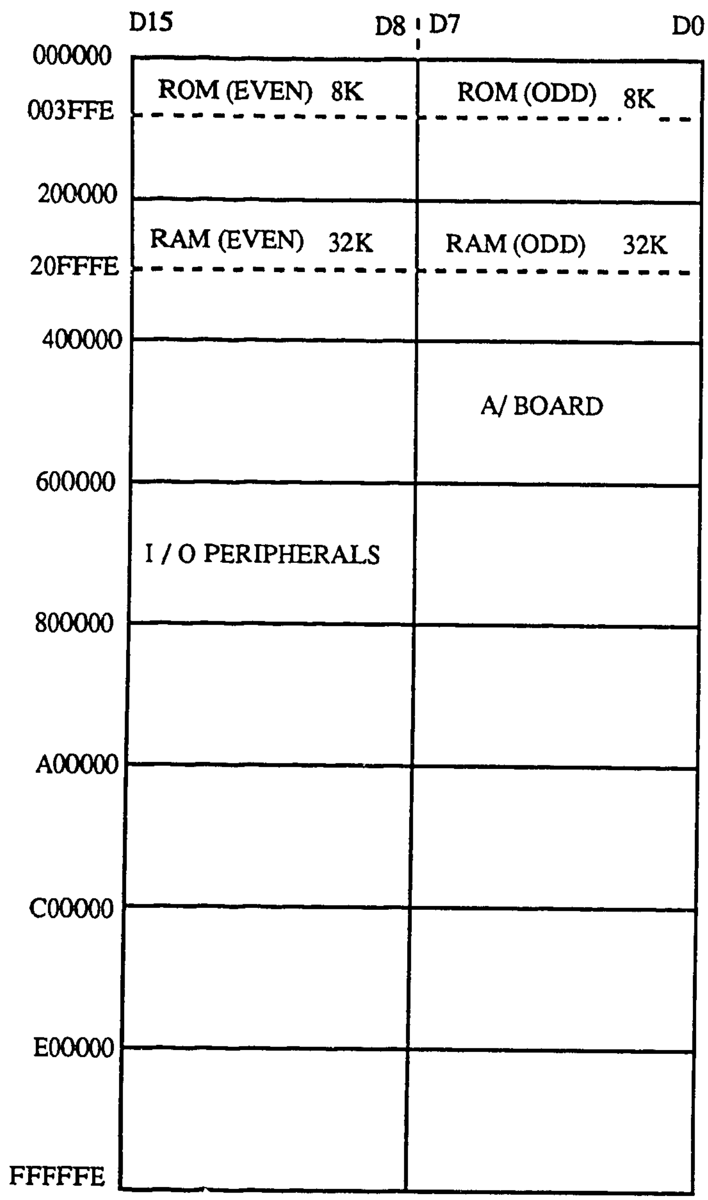

Fig. 3.4 System memory map

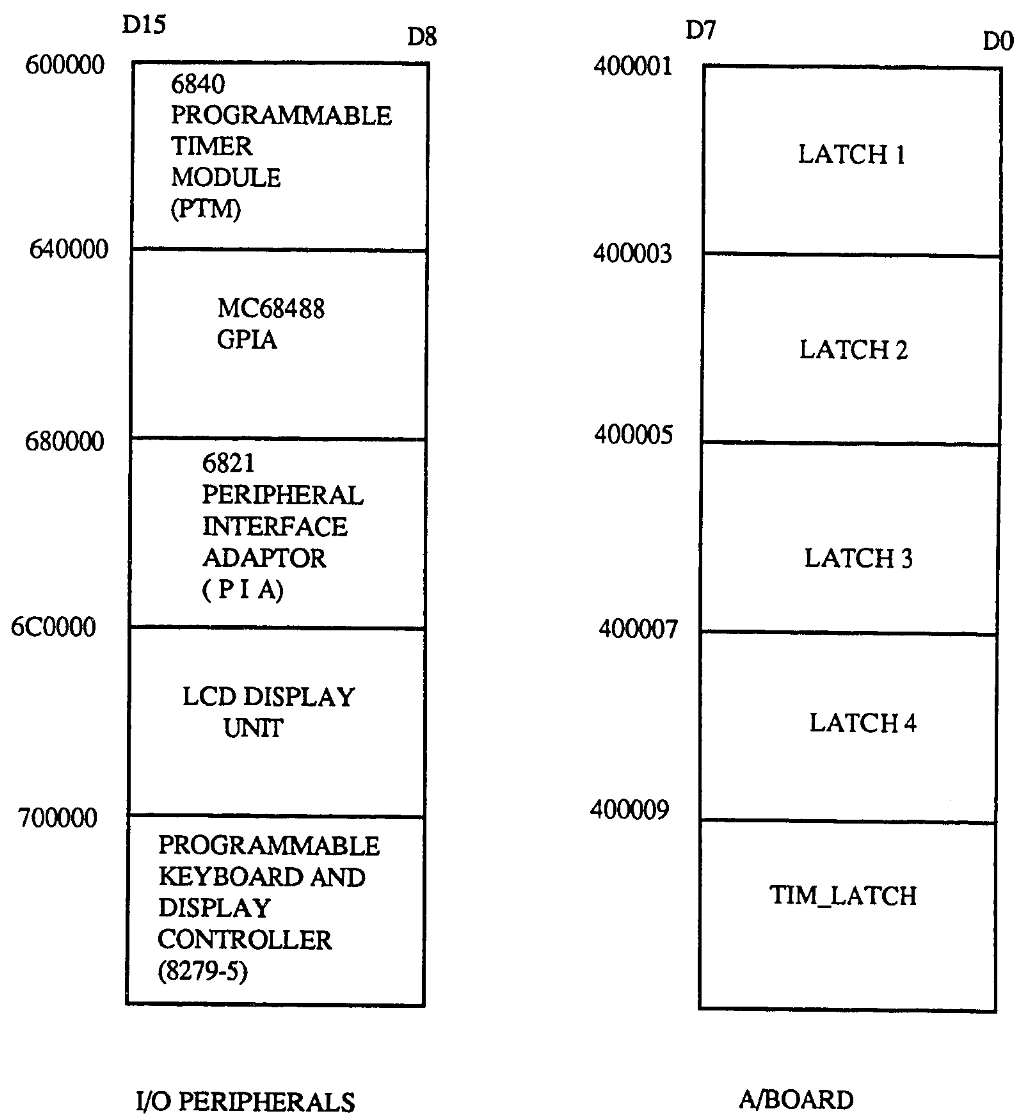

Fig. 3.5 Detailed memory map of I/O peripherals and A/Board

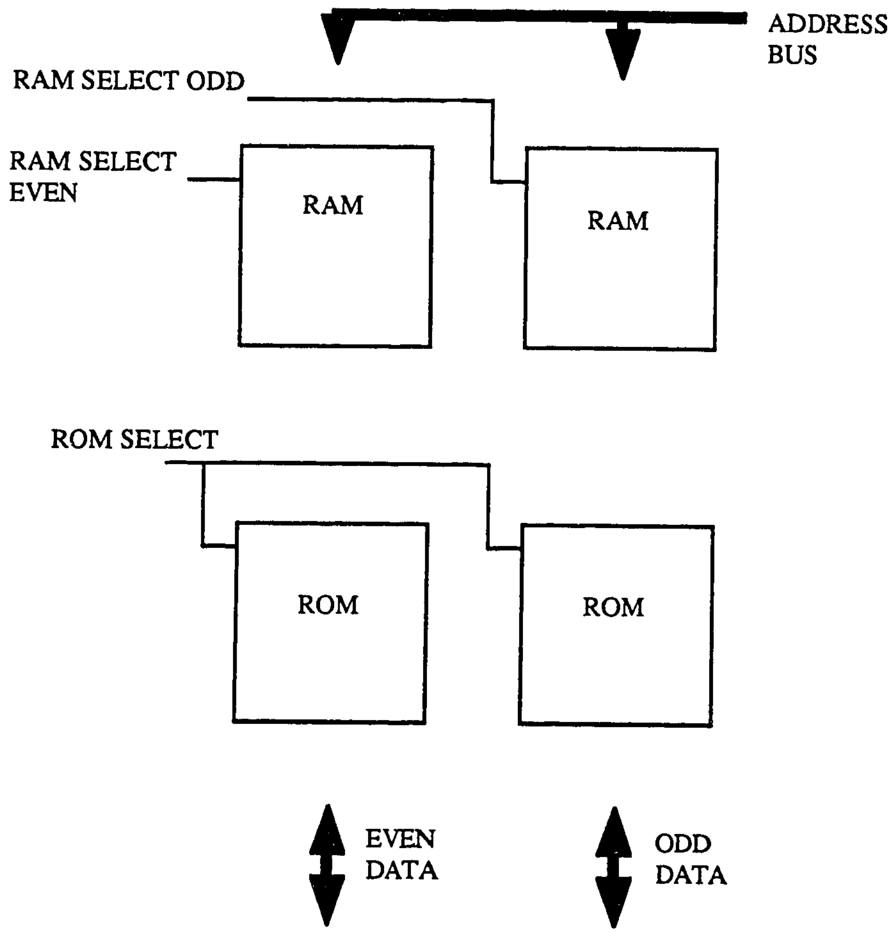

Fig. 3.6 Block diagram of memory circuitry

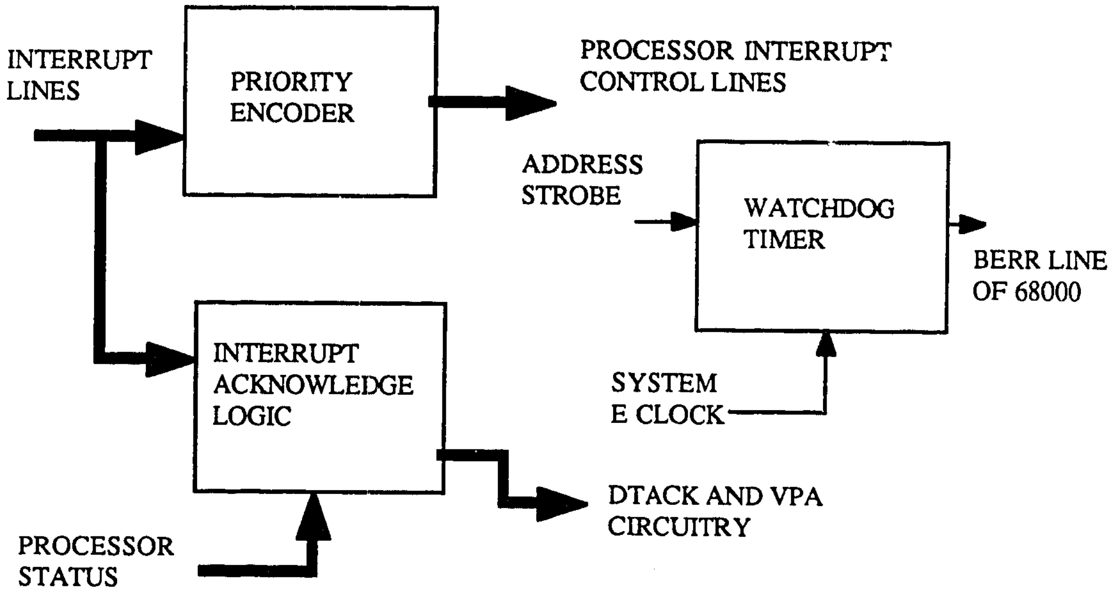

Fig. 3.7 Block diagram of interrupt and watchdog timer circuitry

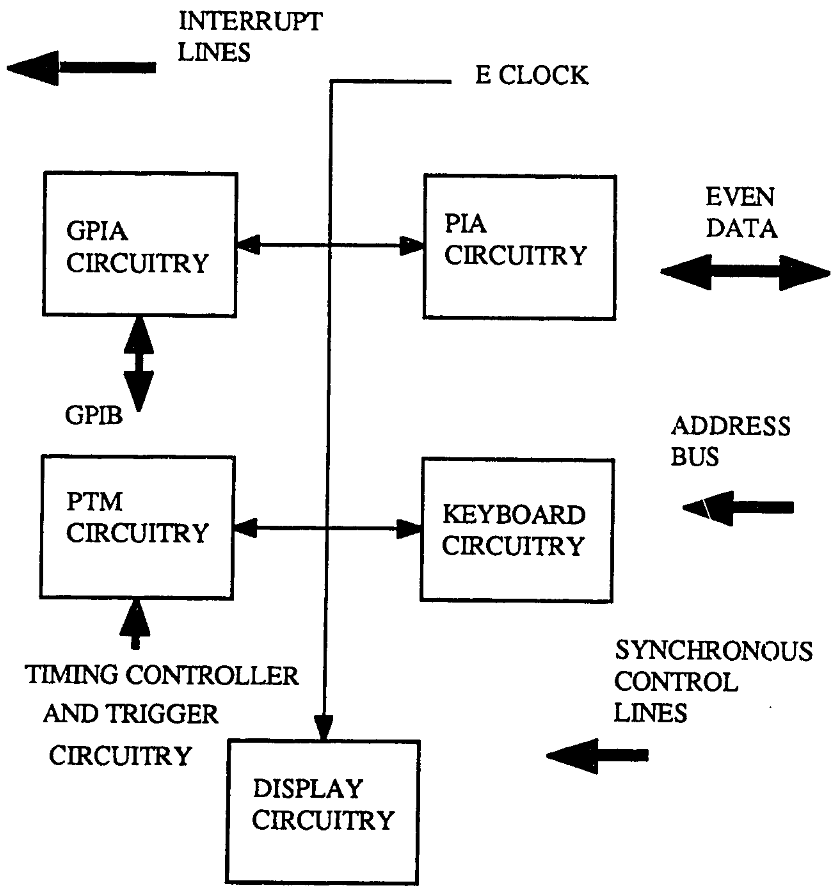

Fig. 3.8 Block diagram of I/O module

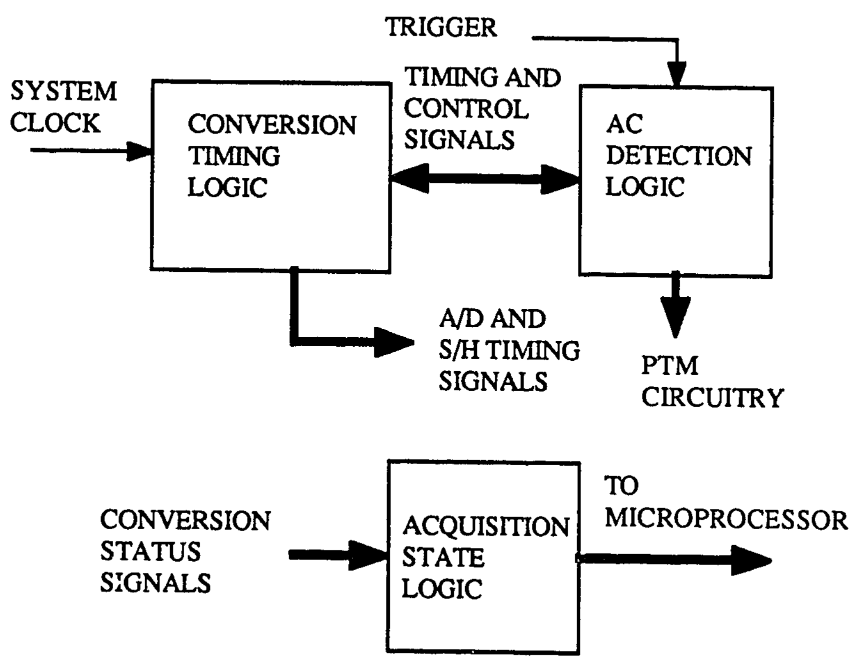

Fig. 3.9 Block diagram of the timing controller module

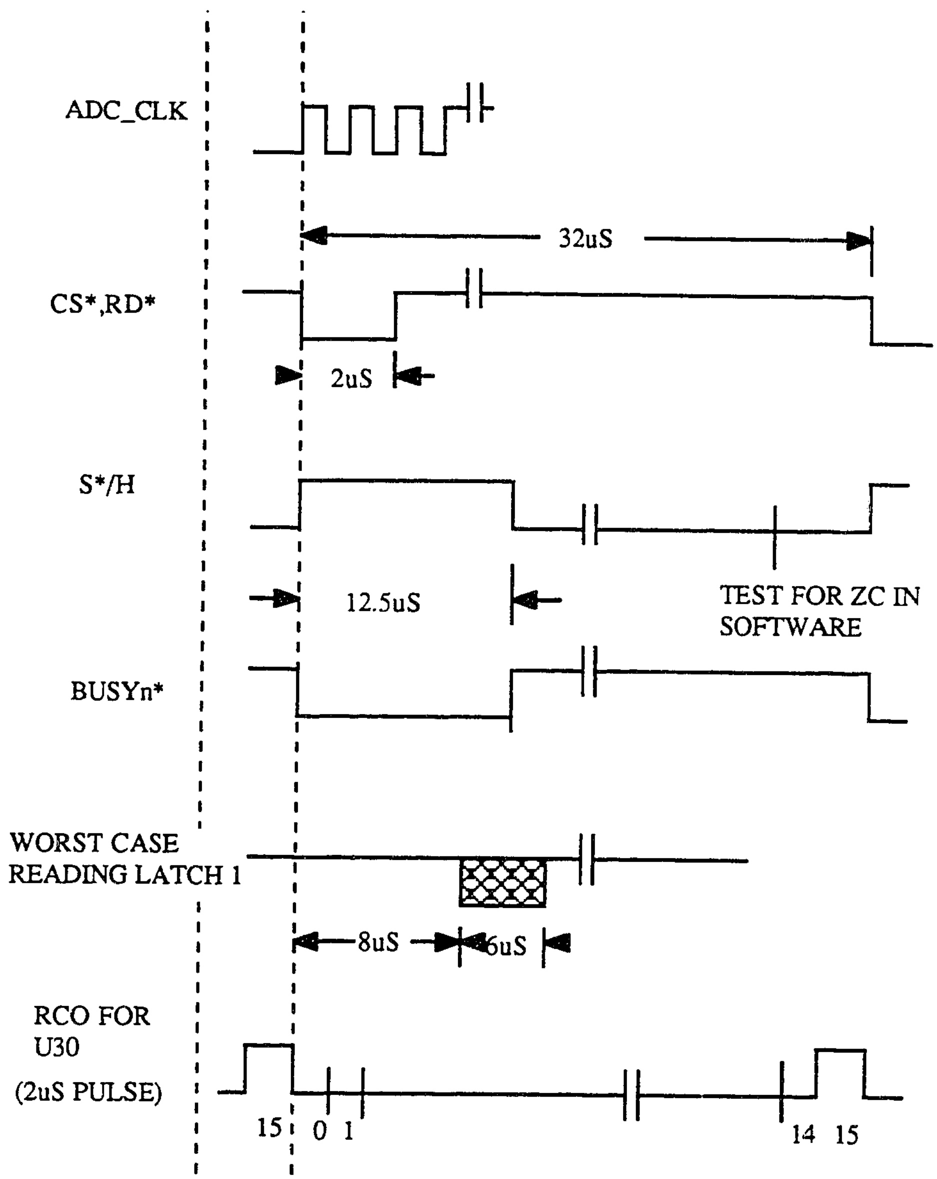

Fig. 3.10 A/D Interface timing diagram

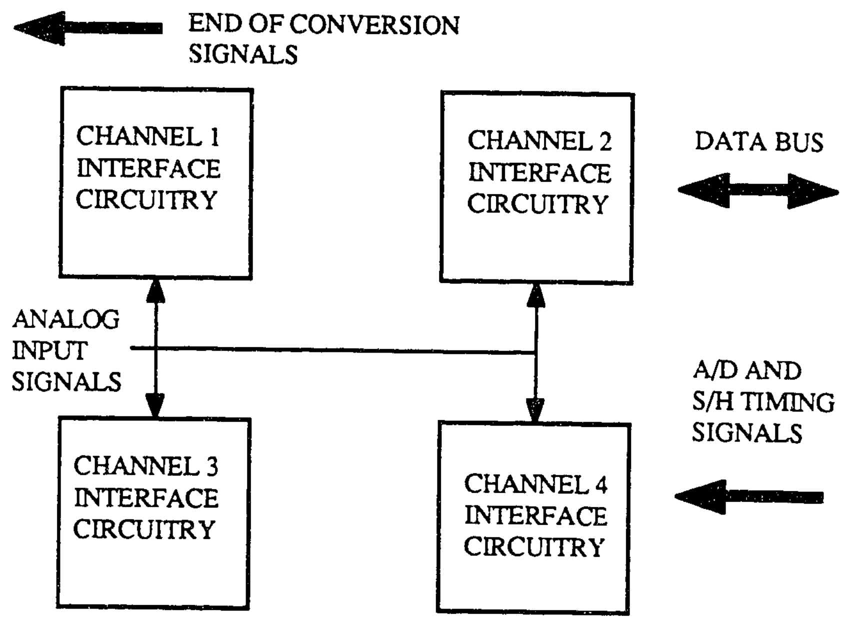

Fig. 3.11 Block diagram of the Analog-to-Digital interface module

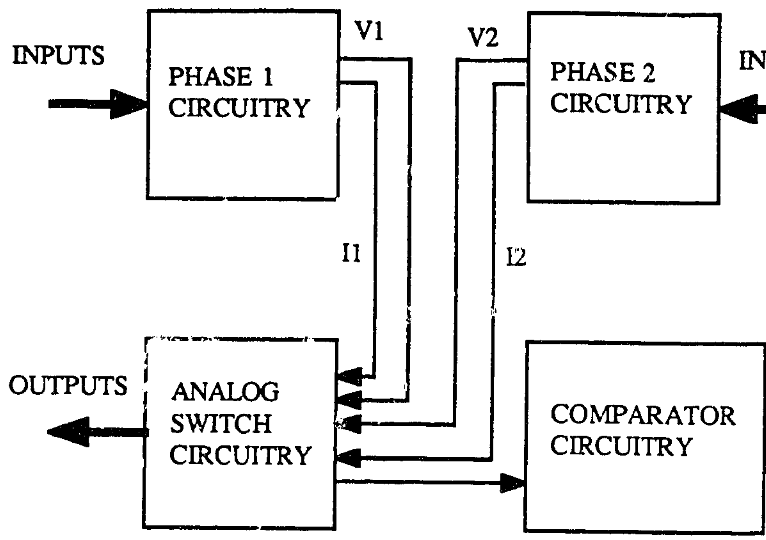

Fig. 3.12 Block diagram of the input conditioning module

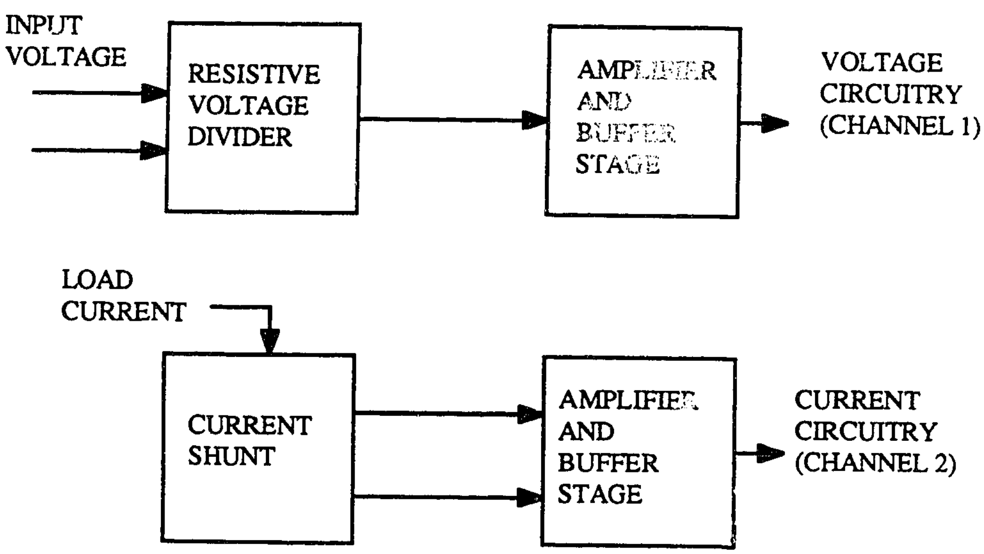

Fig. 3.13 Block diagram of the phase 1 circuitry

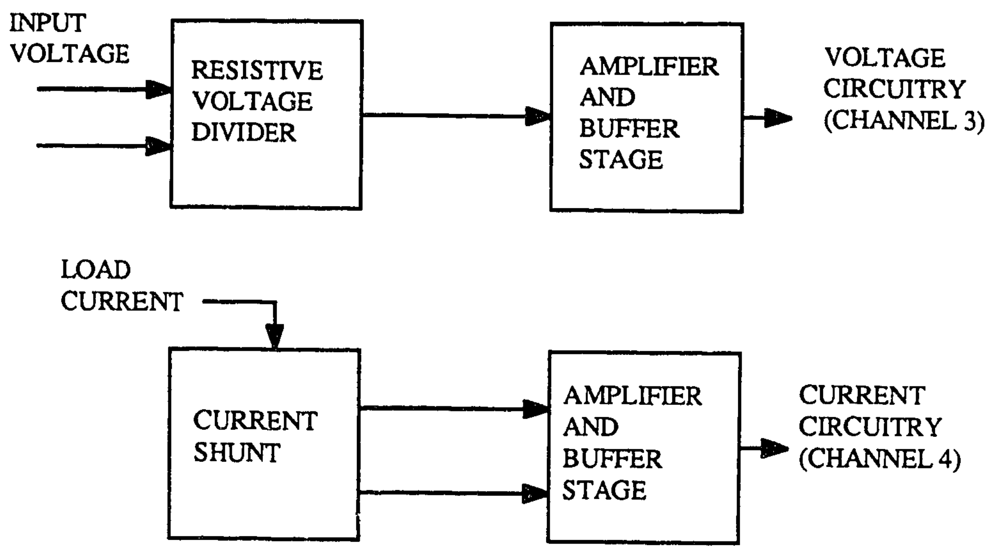

Fig. 3.14 Block diagram of the Phase 2 circuitry

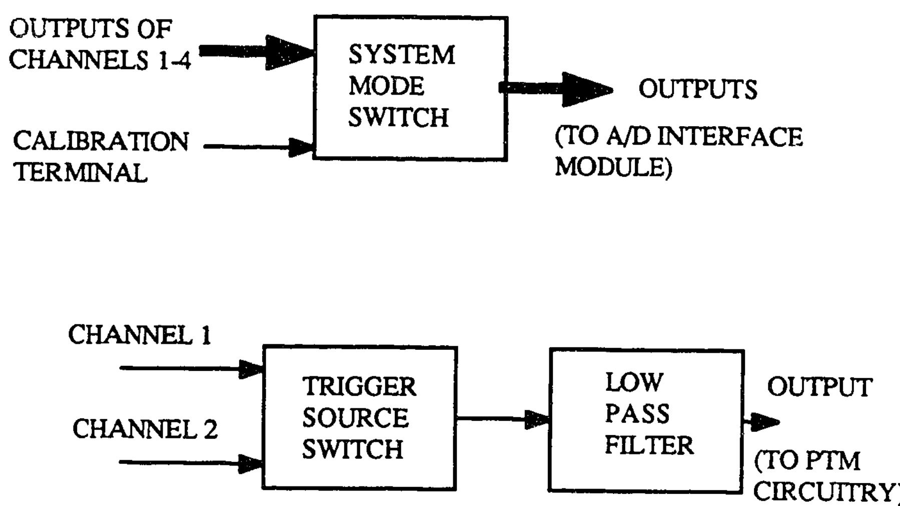

Fig. 3.15 Block diagram of the analog switches circuitry

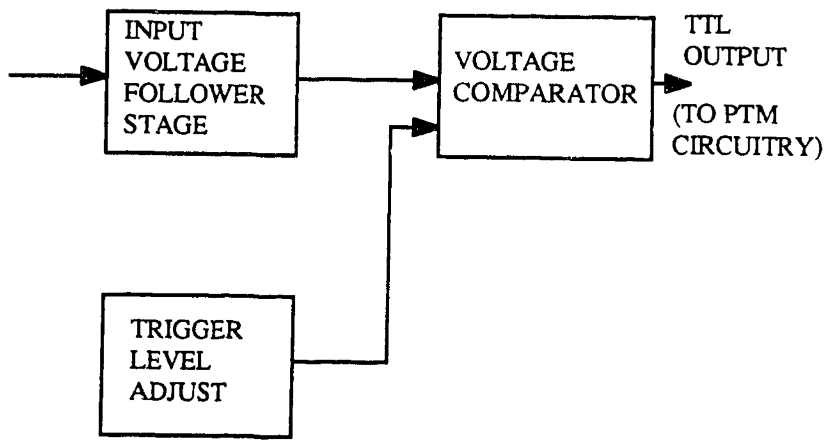

Fig. 3.16 Block diagram of the trigger circuitry

4. DIGITAL SAMPLING WATTMETER SOFTWARE

The software structure is geared towards the efficient and speedy implementation of the necessary mathematical manipulations and wattmeter functions required. This ties in suitably with the design goal of implementing a very accurate system using reasonably priced off-the-shelf components.

The hardware and software marriage is effectively achieved with 'idle' periods such as occur during sampling intervals being utilized for program code execution by the microprocessor.

With the ease of use and maintenance in mind, a user-friendly software interface is provided, as well as routines to simplify the system calibration and maintenance. The wattmeter program source code is included in Appendix B.

4.1 Software Overview

The flowchart in Fig. 4.1 (page 79) indicates the program flow of the digital sampling wattmeter.

On power-up or following a system reset, the system initialization and memory test are performed. This is followed by the execution of the main program.

4.2 Initialization and Memory test

The flow diagram of this section is shown in Fig. 4.2

(page 80). Initially, all system interrupts are disabled. Subsequently, the flip-flops and timing controller are initialized. This is followed by the initialization of the microprocessor peripheral devices namely; the 6821 PIA, the 8279 programmable keyboard and display controller, the 6840 PTM, the MC68488 GPIA ICs as well as the LCD display.

On successful completion of the above procedure, the memory test is performed. This test extensively checks all 64 kilobyte locations in the system static RAMS. A static ram test [3] with 5 data backgrounds is used. An indication of the current data background (0-4) being used is reflected on the corresponding LED of the bargraph display. A failure of this test results in the processor displaying the failing address . In the last-mentioned mode, a system reset is the only option available to the user in restarting the system.

The ensuing program code enables the keypad and performs the second initialization. In this initialization stage the voltage, current and power scale factors are stored in the static RAMS. Next, the program mode variables are initialized and all interrupts are enabled. Subsequently, the analog-to-digital interface module has its inputs switched to the outputs of channels 1 to 4. All microprocessor interrupts are enabled.

4.3 The main program

The flow diagram of this module is indicated in Fig. 4.3 (page 81). The program code interacts directly with the hardware in accepting user inputs as well as computing and displaying results as required through an interaction of system routines.

4.3.1 The main menus

The routines used in this stage are the input and LCD string display routines. The limitations of the LCD mentioned earlier prevent the display of the description of all the 32 possible keypad commands on the LCD display simultaneously. As a result the command summary is divided into 8 menus denoted menu 0 to menu 7. The screen display of these menus are indicated in Fig. 4.4 (page 82). The display of the next menu is always performed using the ^F command. The display of the previous menu being performed using the ^B command. A program command having a caret prefix indicates the simultaneous enabling of the control switch and the corresponding keypad character. Following the execution of a particular keypad command, the previous main menu displayed is always entered.

4.3.2 Command preprocessor

This section of the main program is always entered following the display of a particular main menu. The main routine used is the input routine. The purpose of the command preprocessor is to determine the command entered and transfer program control to the required module. On entering a non-defined command, or in the absence of any command, the software simply loops through the command look-up table. The commands required by the program are passed through the byte-wide key buffer. The contents of the latter memory location is used by the command preprocessor software in transferring program control to the required module. The system hardware interacts with the key buffer either through the keypad or externally through the IEEE-488 bus (GPIB). Both sources operate in an interrupt mode.

The keypad can be operated in either the normal or the keypad programming mode. In the normal mode, commands are entered in response to the displayed menu messages. The command entered is duplicated in the key buffer by the interrupt service routine executed when a key closure occurs. The keypad programming mode is basically an emulation of the GPIB bus programming mode. In the GPIB bus programming mode, the user enters a programming code string through an active talker on the bus. This latter string, comprising the key code commands entered in the preferred order of execution, is stored in the 128 byte programming code buffer. The programming mode semaphore is also set in this mode. This enables the loading of the next program byte from the program code string into the key buffer during the ensuing program execution.

The GPIB program commands are ; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U and V. The corresponding keypad commands are; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, ^0, ^1, ^2, ^3, ^4, ^5, ^6, ^7, ^8, ^9, ^A, ^B, ^C, ^D, ^E and ^F respectively.

4.3.3 Module 0

The flow diagram of this module is shown in Fig. 4.5 (page 83). This module is executed when a '0' is processed by the command preprocessor. This module restarts the wattmeter.

4.3.4 Module 1

The flow diagram of this module is shown in Fig. 4.6 (page 84). This module is executed when a '1' is processed by the command preprocessor. The routines used are the LCD and bargraph display routines. The function of this module is to toggle the display between the on and off states. The status of the display is reflected on LED 0 of the bargraph display using a bit in a byte-wide memory location. The bit manipulation instructions of the 68000 microprocessor enable the ease of implementation of the bargraph display.

4.3.5 Module 2

The flow diagram of this module is shown in Fig. 4.7 (page 85). This module is executed when a '2' is processed by the command preprocessor and uses the display and scale factor routines. The purpose of this module is to set the correct scale factors for the computation of the system parameters when either the 200v or the 350v range on channel 1 is being used. Following a system power-up, the 200v range is used. The range used is reflected on LED 3 of the bargraph display. Subsequently, the required voltage, current and power scale factors are computed using the scale factor routines. Following the latter procedure, the hex value FF is loaded into the key buffer. The hex value FF does not correspond to any key code. Hence, the transfer of program control to a particular module after a return to the command preprocessor module via the main menu, can only be effected when a new command is entered.

4.3.6 Module 3

The flow diagram of this module is shown in Fig. 4.8 (page 86). This module is executed when a '3' is processed by the command preprocessor. The function of this module is similar to that of module 2. However in this case the range indication refers to channel 3. In addition, the range status is indicated on LED 4.

4.3.7 Module 4

The flow diagram of this module is shown in Fig. 4.9 (page 87). This module is executed when a '4' is processed by the command preprocessor. This module uses the bargraph display routine and one of the utility routines namely; the overload test routine. The purpose of this module is to cause an execution of the overload routine or cause the program to skip over this routine following the execution of the acquisition routine. The software uses the state of the overload semaphore in determining the execution or otherwise of the overload routine. The program default mode following a system power-up or reset, is the execution of the overload test. This module is relevant during a system calibration or maintenance. The system status regarding the execution of the overload test is reflected on LED 1 of the bargraph display. Following the latter procedure, the hex value FF is loaded into the key buffer. The latter approach forces a return to the main menu and also prevents the re-execution of this module following the return.

4.3.8 Module 5

The flow diagram of this module is shown in Fig. 4.10 (page 88). This module is executed when a '5' is processed by the command preprocessor and uses the LCD display routine. The primary use of this module is the changing of the numerator of the voltage or current scale factors. This module is relevant when new scale factors need to be programmed as a result of the replacement or aging of components. Following the execution of a '5', the sub-menu in Fig. 4.11 (page 89) is displayed. After selecing the appropriate channel, the user selects 'F', 'B' or '6 to increment, decrement or terminate scale factor updating respectively. Subsequently, the hex value FF is loaded into the key buffer.

4.3.9 Module 6

The flow diagram of this module is shown in Fig. 4.12 (page 88). This module is executed when a '6' is processed by the command preprocessor. This module uses the LCD display routine and is similar to module 5. However, the primary use of this module is the changing of the exponent of the denominator of the voltage or current scale factors. The denominators of each scale factor have a mantissa of 2.

4.3.10 Module 7

The flow diagram of this module is shown in Fig. 4.13 (page 90). This module is executed when a '7' is processed by the command preprocessor. This module uses the bargraph display routine. Under software control using the PIA, the trigger source switch is employed in selecting either channel 1 or channel 2 as the system trigger source. The trigger source status is indicated on LED 1 of the bargraph display. Following a system power-up or reset, channel 1 is used as the system trigger source. The hex value FF is loaded into the key buffer prior to exiting this module.

4.3.11 Module 8

The flow diagram of this module is shown in Fig. 4.14 (page 91). This module is executed when a '8' is processed by the command preprocessor. This module reinitializes the data storage parameters. This module sets the size of the data stored in RAM to zero in addition to setting the data pointer to the base of the RAM location. The data pointer points to the next free memory location for the storage of the LCD display data. Following the execution of this module, a return to the main menu is performed.

4.3.12 Module 9

The flow diagram of this module is shown in Fig. 4.15 (page 92). This module is executed when a '9' is processed by the command preprocessor. This module uses the AC acquisition, display and AC computation routines. The AC acquisition routine enables the acquisition of samples for a single cycle of the input signal. The AC computation routines utilize the procedure mentioned in section 2.3.4. The purpose of this module is to compute and display the rms value of the sampled data in the presence of an AC signal on the calibration terminal. On entering this module, the system mode switch is switched over to the

$$= \frac{1}{n}\left[\frac{(1-\cos(n\theta)(1-\cos\theta) + \sin\theta\sin(n\theta)}{(1-\cos\theta)^2 + \sin^2\theta}\right]$$

now $(n + \Delta)h = 2m\pi$

$\Rightarrow (n + \Delta)kh = 2mk\pi$

$nkh = 2mk\pi - \Delta kh$

$\Rightarrow n\theta = 2mk\pi - \Delta kh$

$\Rightarrow \sin(n\theta) = -\sin(\Delta kh)$

and $\cos(n\theta) = \cos(\Delta kh)$

hence;

$$\overset{A}{\beta}_k = \frac{1}{n}\left[\frac{(1-\cos(\Delta\theta))(1-\cos\theta) - \sin\theta\sin(\Delta\theta)}{2(1-\cos\theta)}\right]$$

$$= \frac{1}{n}\left[\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{1}{2}\cot\left(\frac{k\ h}{2}\right)\sin(\Delta kh)\right]$$

2.3.2 Trapezoidal rule

The expression for n+1 samples using the trapezoidal rule is:

$$Ay_j = \frac{1}{n}\left(0.5y_0 + \sum_{j=1}^{n-1} y_j + 0.5y_n\right) \qquad (2.16)$$

$$= \frac{1}{n}\left(\sum_{j=0}^{n-1} y_j - 0.5y_0 + 0.5y_n\right)$$

The corresponding error coefficient relating to the sine term of the error expression is:

$$=> \quad \alpha_k^T = A \sin(jkh)$$

$$= \frac{1}{n}\left[\cot\left(\frac{k\,h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) + \frac{\sin(\Delta kh)}{2} - \frac{\sin 0}{2} + \frac{\sin(nkh)}{2}\right]$$

$$= \frac{1}{n}\left[\cot\left(\frac{k\,h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) + \frac{\sin(\Delta kh)}{2} - \frac{\sin 0}{2} - \frac{\sin(\Delta kh)}{2}\right]$$

$$= \frac{1}{n}\left[\cot\left(\frac{k\,h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right)\right]$$

The corresponding error coefficient relating to the cosine term of the error expression is:

$$\beta_k^T = A \cos(jkh)$$

$$= \frac{1}{n}\left[\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{1}{2}\cot\left(\frac{k\,h}{2}\right)\sin(\Delta kh) - \frac{\cos 0}{2} + \frac{\cos(nkh)}{2}\right]$$

$$= \frac{1}{n}\left[\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{1}{2}\cot\left(\frac{k\ h}{2}\right)\sin(\Delta kh)\right.$$

$$\left. - \frac{\cos 0}{2} + \frac{\cos(\Delta kh)}{2}\right]$$

$$= - \frac{1}{2n}\left[\cot\left(\frac{k\ h}{2}\right)\sin(\Delta kh)\right]$$

2.3.3 NBS approach

The expression for n+1 samples using the NBS approach [12] is:

$$Ay_j = \frac{1}{n + \Delta}\left(\sum_{j=0}^{n-1} y_j + \Delta * y_n\right) \qquad (2.17)$$

The corresponding error coefficient relating to the cosine term of the error expression is:

$$=> \quad \alpha_k^S = A \sin(jkh)$$

$$= \frac{1}{n + \Delta}\left(\cot\left(\frac{k\ h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) + \frac{\sin(\Delta kh)}{2}\right.$$

$$\left. + \Delta\sin(nkh)\right)$$

$$= \frac{1}{n + \Delta}\left(\cot\left(\frac{k\ h}{2}\right)\ \sin^2\left(\frac{\Delta kh}{2}\right) + \frac{\sin(\Delta kh)}{2}\right.$$

$$\left. - \Delta\sin(\Delta kh)\right)$$

$$= \frac{1}{n + \Delta}\left[\cot\left(\frac{k\ h}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right)\right.$$

$$\left. + \frac{1}{2}(1-2*\Delta)\sin(\Delta kh)\right]$$

The corresponding error coefficient relating to the cosine term of the error expression is:

$$\Rightarrow \quad \beta_k^S = A\cos(jkh)$$

$$= \frac{1}{n + \Delta}\left[\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{1}{2}\cot\left(\frac{k\ h}{2}\right)\sin(\Delta kh)\right.$$

$$\left. + \Delta\cos(nkh)\right]$$

$$= \frac{1}{n + \Delta}\left[\sin^2\left(\frac{\Delta kh}{2}\right) + \Delta\cos(\Delta kh)\right.$$

$$\left. - \frac{1}{2}\cot\left(\frac{k\ h}{2}\right)\sin(\Delta kh)\right]$$

2.3.4. Mc ied Trapezoidal Approach

The expression for n+1 samples for an approach proposed by Zu-Liang [11] and referred to in the Modified

Trapezoidal Method (MTM) is:

$$Ay_j = \frac{1}{n+\Delta}\left(0.5y_0+\sum_{j=1}^{n-1} y_j + 0.5y_n + \frac{\Delta}{2}(y_n + y_{n+\Delta})\right) \quad (2.18)$$

$$= \frac{1}{n+\Delta}\left(\sum_{j=1}^{n-1} y_j + 0.5(1+\Delta)(y_0+y_n)\right) \quad (2.19)$$

since $y_0 = y_{n+\Delta}$ for a measurement interval covering of an integral number of signal cycles.

The corresponding error coefficient relating to the sine term of the error expression is:

$$\alpha_k^N = A \sin(jkh)$$

$$= \frac{1}{n+\Delta}\left[\cot\left(\frac{kh}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{\Delta}{2}(\sin(nkh) + \sin(0))\right]$$

$$= \frac{1}{n+\Delta}\left[\cot\left(\frac{kh}{2}\right)\sin^2\left(\frac{\Delta kh}{2}\right) - \frac{\Delta}{2}\sin(\Delta kh)\right]$$

The corresponding error coefficient relating to the cosine term of the error expression is:

$$\beta_k^{N} = \mathbf{A} \cos(jkh)$$

$$= \frac{1}{n + \Delta} \left[- \frac{1}{2} \cot\left(\frac{k\ h}{2} \right) \sin(\Delta kh) + \frac{\Delta}{2}\left(\cos(nkh) + \cos(0) \right) \right]$$

$$= \frac{1}{n + \Delta} \left[- \frac{1}{2} \cot\left(\frac{k\ h}{2} \right) \sin(\Delta kh) + \frac{\Delta}{2}\left(\cos(\Delta kh) + 1 \right) \right]$$

$$= \frac{1}{n + \Delta} \left[- \frac{1}{2} \cot\left(\frac{k\ h}{2} \right) \sin(\Delta kh) + \Delta\cos^2\left(\frac{\Delta kh}{2} \right) \right]$$

$$= \frac{1}{n + \Delta} \left[\Delta\cos^2\left(\frac{\Delta kh}{2} \right) - \frac{1}{2} \cot\left(\frac{k\ h}{2} \right) \sin(\Delta kh) \right]$$

2.4 Determination of delta

Delta (Δ) is determined by employing a hardware scheme which resolves each sampling interval into 16 discrete sub-intervals.The sub-interval in which the end of a period occurs, is divided by 16 to obtain delta. Delta can thus obtained to the nearest one-sixteenth .

This scheme results in a more precise estimate of delta as opposed to the interpolation schemes suggested in the implementation of precision digital sampling meters [12] [14].

In order to minimize the errors resulting from a deviation of the computed interpolated value Δ and the actual value of Δ, the sample size is adjusted so that $|\Delta| \leq 0.5$.

2.5 Computation of the desired parameters

The DC parameters of voltage, current and power are computed using the trapezoidal rule, while the corresponding AC parameters are computed using the processing method outlined in section 2.3.4.

The relationship between the parameters required, the value of y_j in equation 2.8 ,and the further computations required after applying the required processing method, are summarized below.

Parameter required	y_j	*Further computations required*
AC voltage	$(v_j)^2$	compute square-root
AC current	$(i_j)^2$	compute square-root
AC power	$v_j i_j$	
DC voltage	v_j	
DC current	i_j	
DC power	$v_j i_j$	

v_j and i_j are the respective sampled values of voltage and current.

Due to the use of fixed point arithmetic for the computation of most of the AC parameters,the following modified formula is used.

$$Ay_j = \frac{1}{16n + 16\Delta}\left(16\sum_{j=1}^{n-1} y_j + 0.5(16+16\Delta)(y_0+y_n) \right) \qquad (2.20)$$

The above equation is used in the determination of power.

The determination of the rms values however, includes a correction for any dc terms present. The resulting expression is thus;

$$\text{RMS value} = \sqrt{A(y_j^2) - (Ay_j)^2} \qquad (2.21)$$

where y_j is v_j or i_j depending on the parameter required.

and **A** is the processing method used

Another parameter, the power factor (P.F) [11] for sinusoidal signals, is determined as follows:

a) A Single-phase system

$$\text{P.F.} = \frac{\text{POWER}}{v_{jm} i_{jm}} \qquad (2.22)$$

b) A Balanced three-phase three-wire system

$$\text{P.F.} = \left[\frac{(P1 + P2)^2}{(P1+P2)^2 + 3(P2-P1)^2} \right]^{\frac{1}{2}} \qquad (2.23)$$

where v_{jm} is the maximum sampled voltage value

i_{jm} is the maximum sampled current value

P1 and P2 are the phase powers,with P2 being the larger value.

2.6 Summary of results

The relationships obtained for the various processing methods in section 2.3.4., are compared using the relevant parameters employed in the practical implementation of the meter. The error values thus obtained would have been obtained if error-free analog input devices as well as a perfect analog-to-digital conversion process occurred in the meter.

From the equations obtained in section 2.3.4, it is observed that the error coefficients are functions of Δ(delta), k (the harmonic number), h (the sampling interval in radians) and n the number of samples processed. In the actual meter implementation, a fixed sampling interval of 32μs and an acquisition cycle of one period of the input signal cycle are used. In addition, for the processing method used, the sample size is adjusted to ensure $|\Delta| \leq 0.5$. Frequencies around the nominal power line frequency of 60 hz which result in values of delta which satisfy the following relation are used.

$\{ 0 < |\Delta| \leq 0.5 \}$

The frequencies employed range from 59.925 hz to 60.035 hz in increments of 0.005 hz.

Graphical representations of the results deduced in section 2.3.4 and the relationships between the various processing method and the approach employed, are indicated in figs. 2.1 - 2.12. The 32μs sampling interval used in the instrument is employed for these representations.

Figs. 2.1 - 2.8 are plotted for signal harmonics of 1 (the fundamental), 2, 10 and 20. In figs. 2.9 - 2.12, harmonics from 1 to the number of samples processed in a cycle (521 in the cases analyzed) are used. Figs. 2.1 - 2.6 indicate the relationship between the ratios of the error coefficients of the average, trapezoidal and NBS approach to that of the procedure used in this work. Figs 2.1 - 2.3 refer to the alpha ratios while figs. 2.4 - 2.6 are for the beta ratios. Figs 2.7 and 2.8 indicate the relationship between the values of error coefficients α_k^N and β_k^N respectively for signal harmonics of 1 (fundamental), 2, 10 and 20.

Figs. 2.1 - 2.5 indicate a general decrease in the error ratios with increasing harmonic number. The processing method used in this work results in an improvement of the alpha error coefficient of the signal fundamental frequency component by a factor of about 10^5 compared with the component calculated from the average, trapezoidal and NBS methods. The beta error coefficients that result from using the average and trapezoidal methods are identical, with an improvement of about 10^5 being

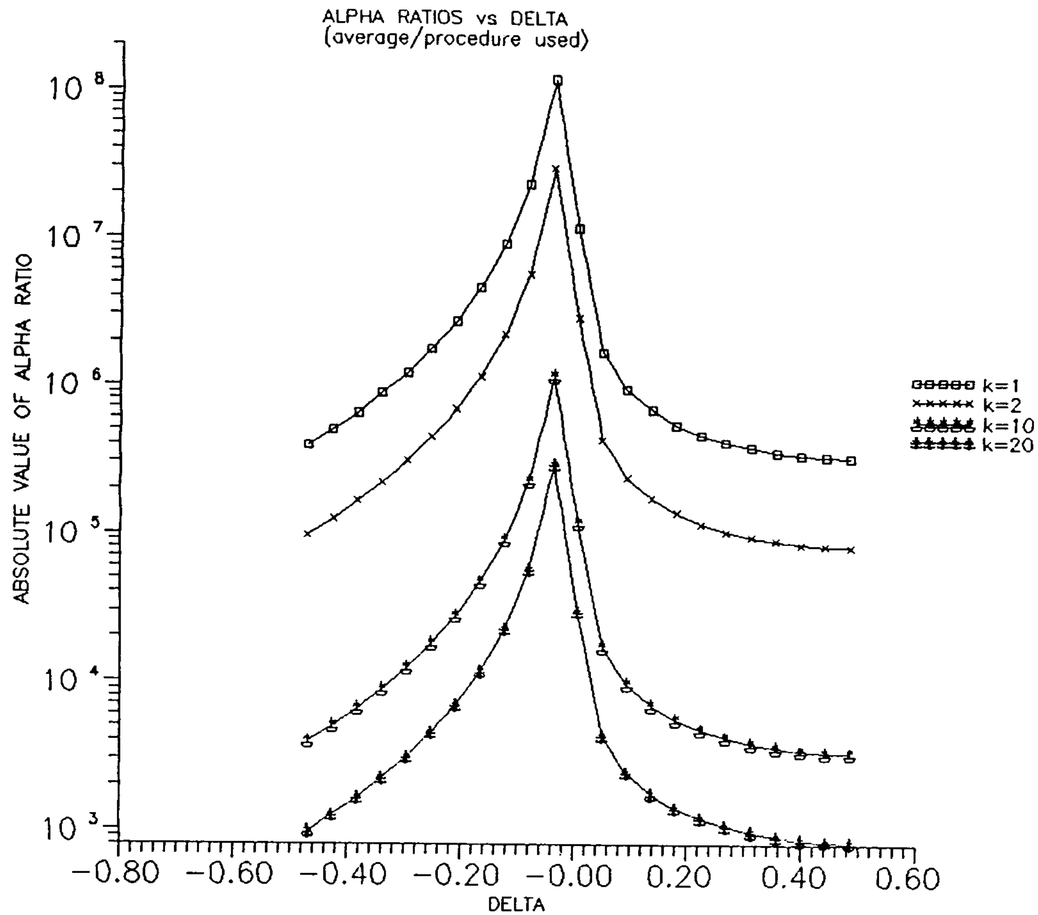

FIG. 2.1 GRAPH OF ALPHA RATIOS vs DELTA (AVERAGE/APPROACH USED)

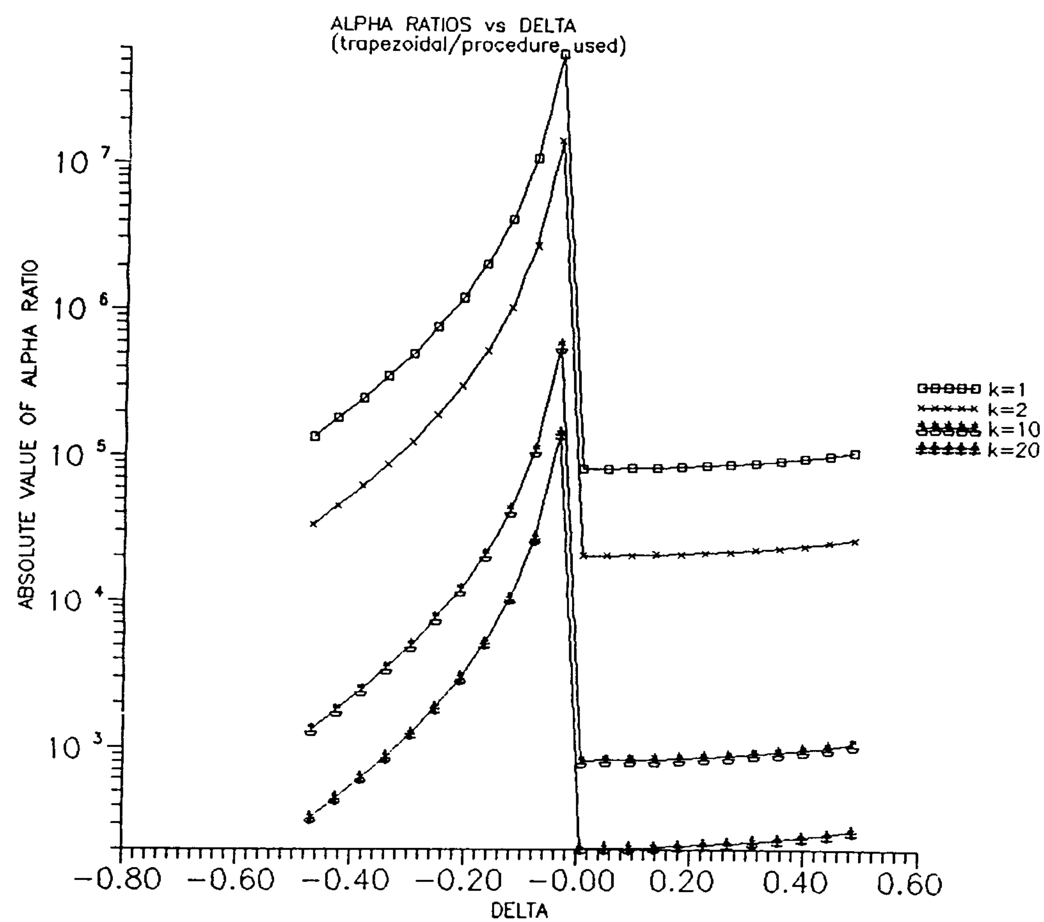

FIG. 2.2 GRAPH OF ALPHA RATIOS vs DELTA (TRAPEZOIDAL/PROCEDURE USED)

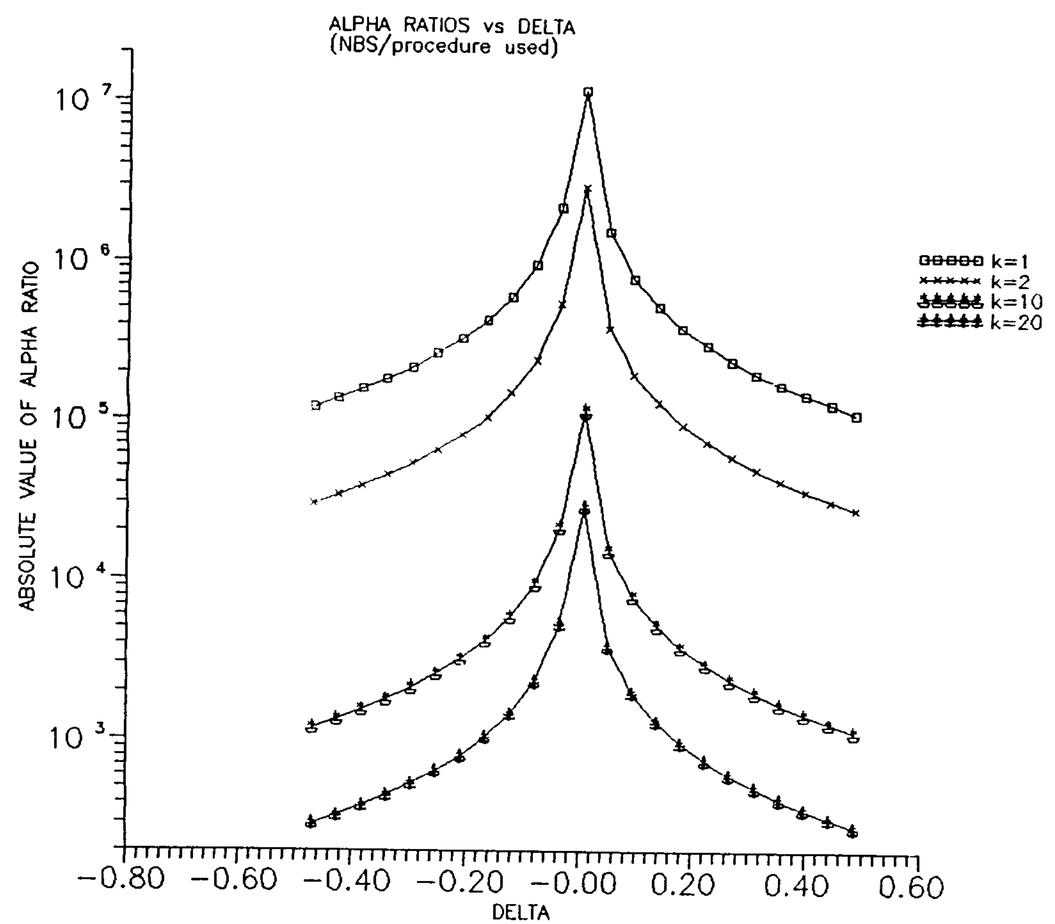

FIG. 2.3 GRAPH OF ALPHA RATIOS vs DELTA (NBS/PROCEDURE USED)

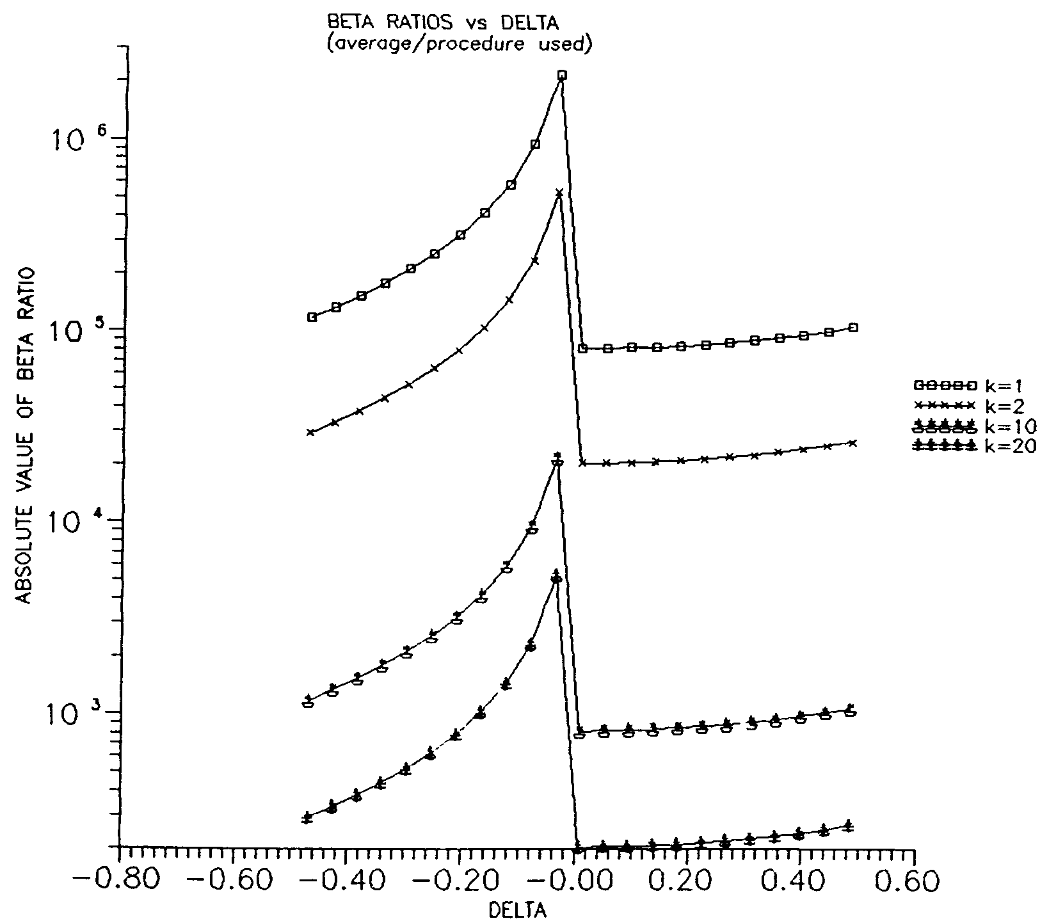

FIG. 2.4 GRAPH OF BETA RATIOS vs DELTA (AVERAGE/PROCEDURE USED)

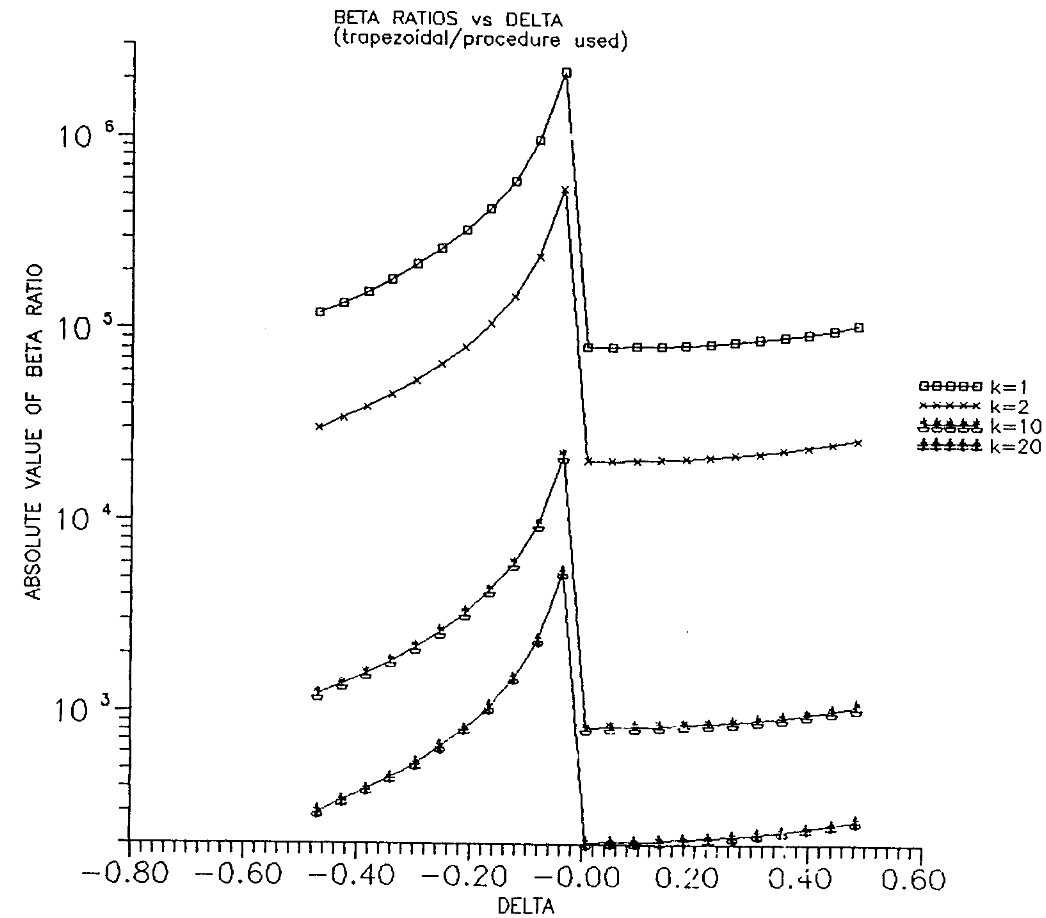

FIG. 2.5 GRAPH OF BETA RATIOS vs DELTA (TRAPEZOIDAL/PROCEDURE USED)

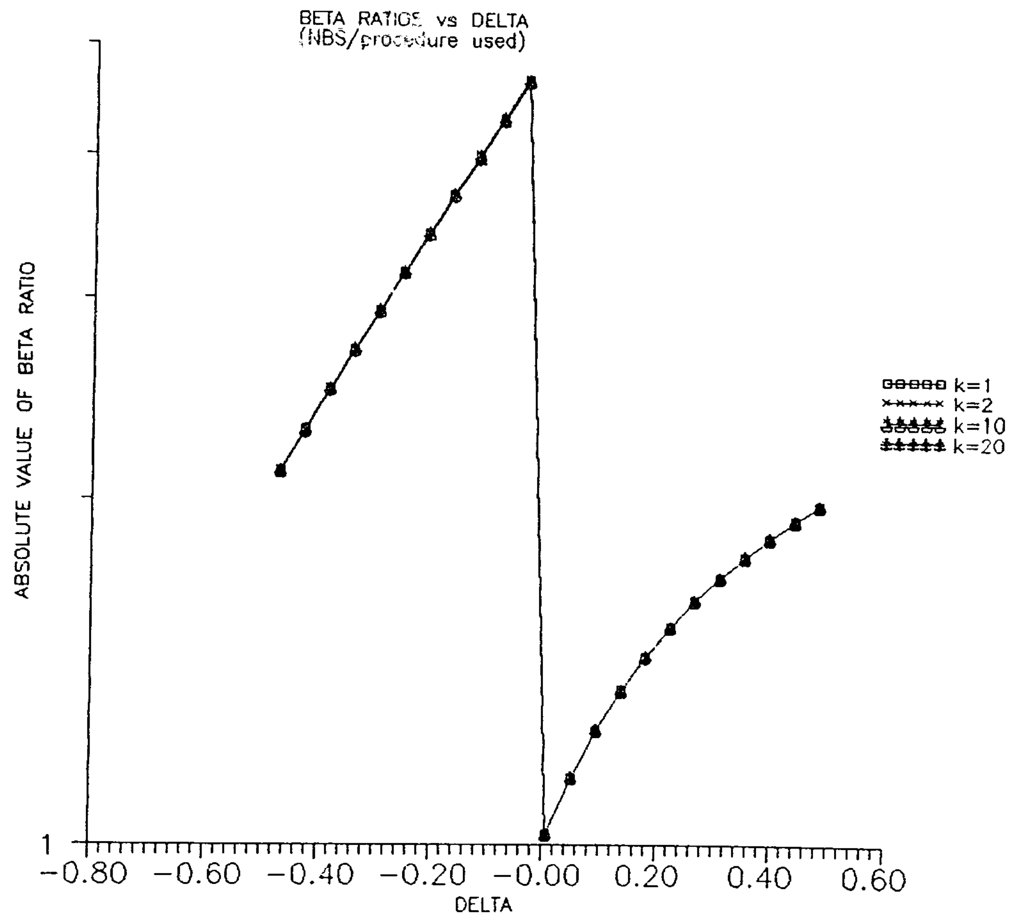

FIG. 2.6 GRAPH OF BETA RATIOS vs DELTA (NBS/PROCEDURE USED)

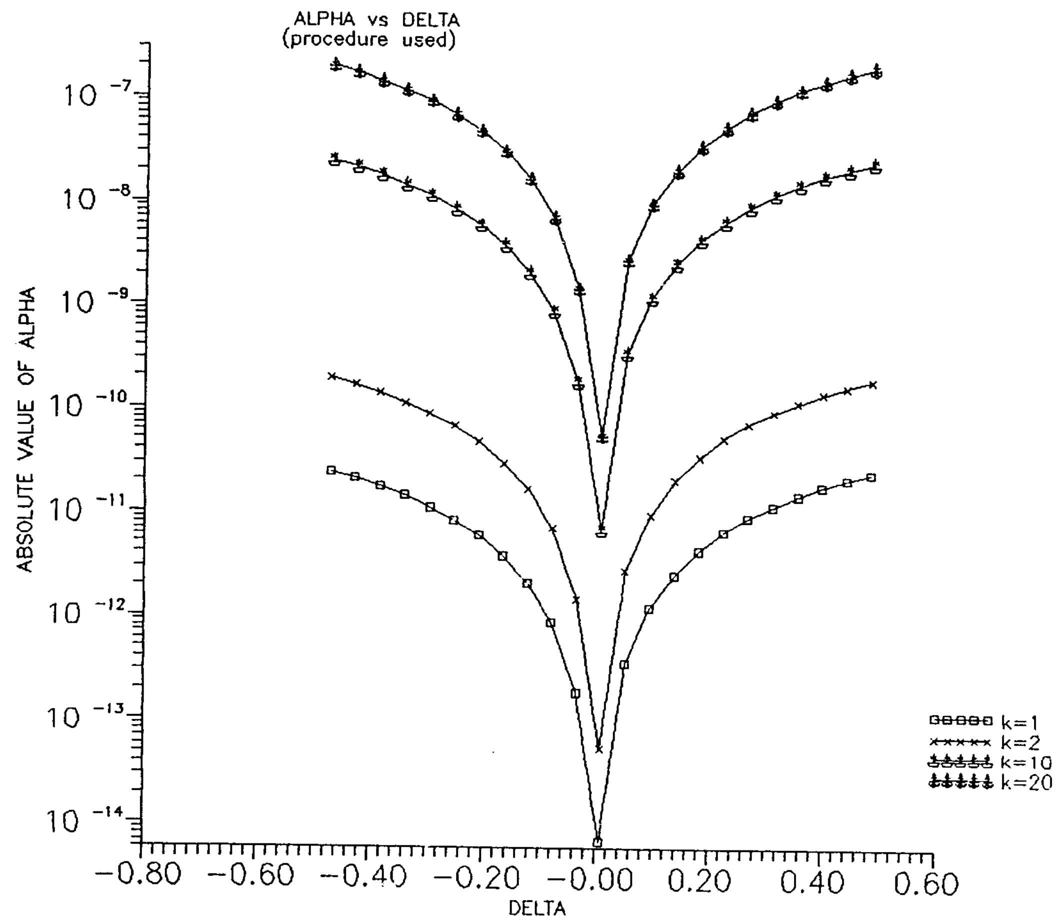

FIG. 2.7 GRAPH OF ALPHA vs DELTA FOR THE PROCEDURE USED (MTM)

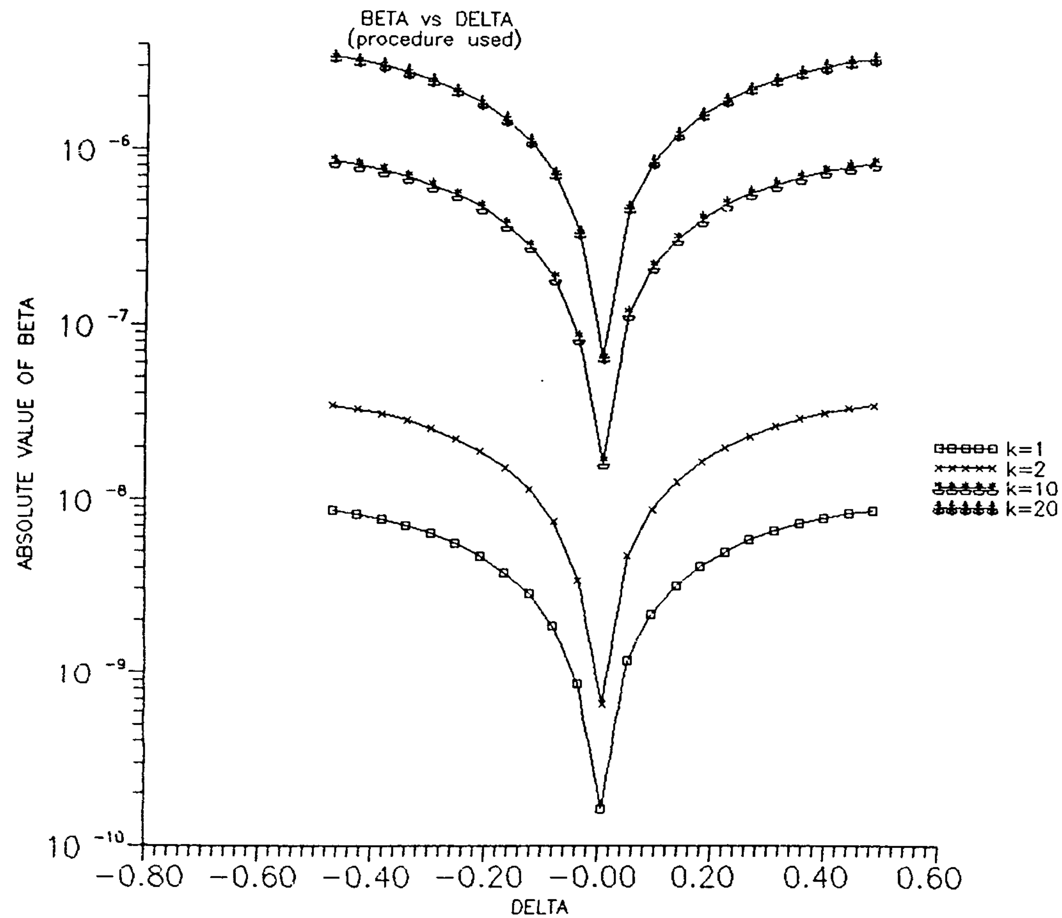

FIG. 2.8 GRAPH OF BETA vs DELTA FOR THE PROCEDURE USED (MTM)

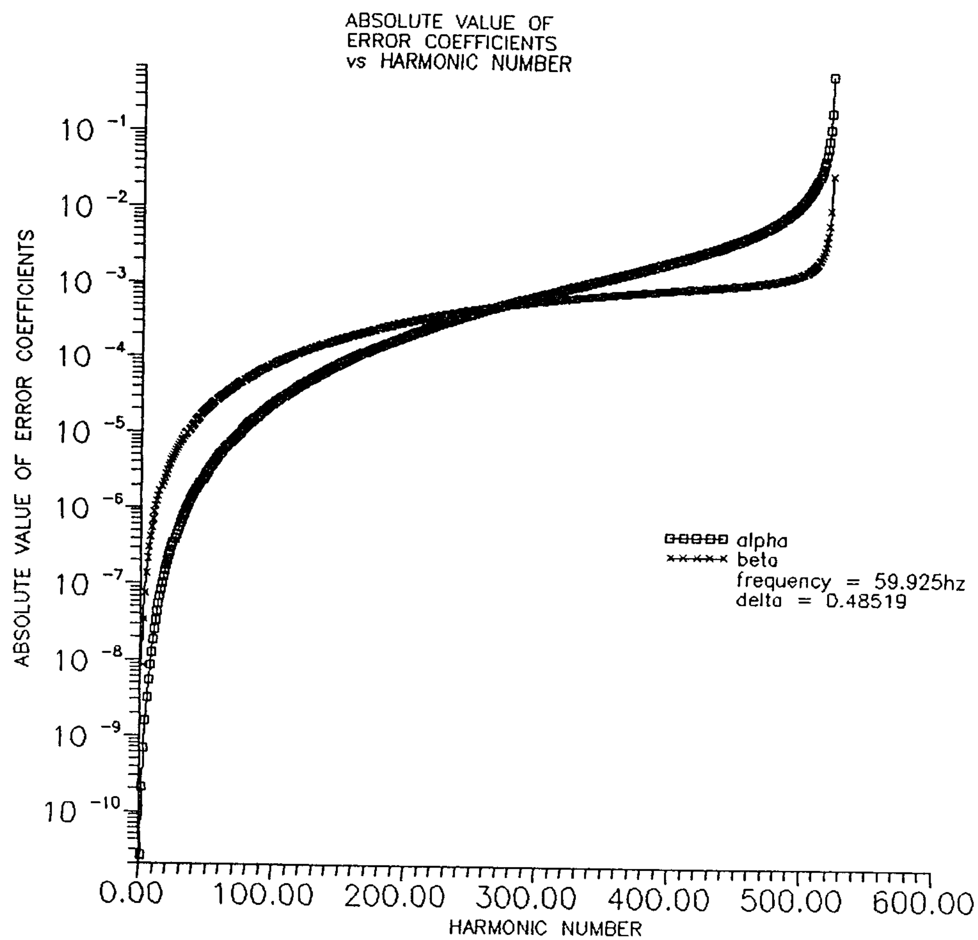

FIG. 2.9 GRAPH OF ABSOLUTE VALUE OF ERROR COEFFICIENTS vs HARMONIC NUMBER FOR A FREQUENCY OF 59.925 hz

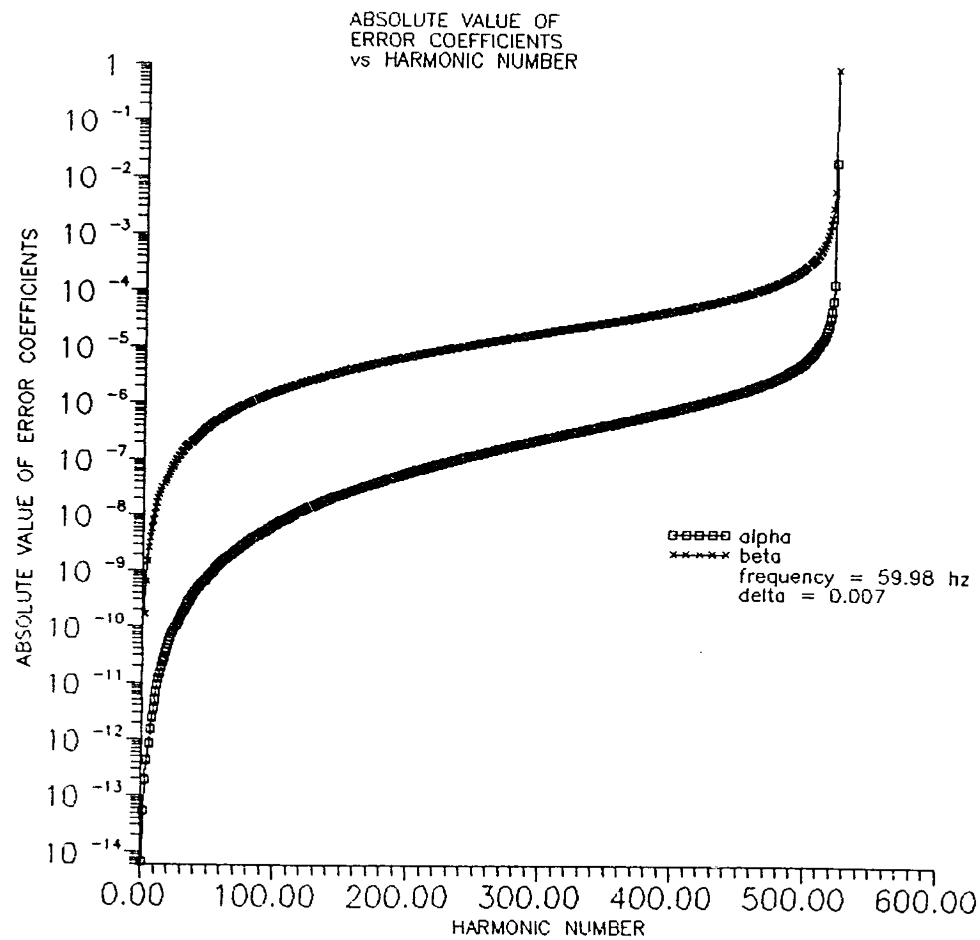

FIG. 2.10 GRAPH OF ABSOLUTE VALUE OF ERROR COEFFICIENTS vs HARMONIC NUMBER FOR A FREQUENCY OF 59.98 hz

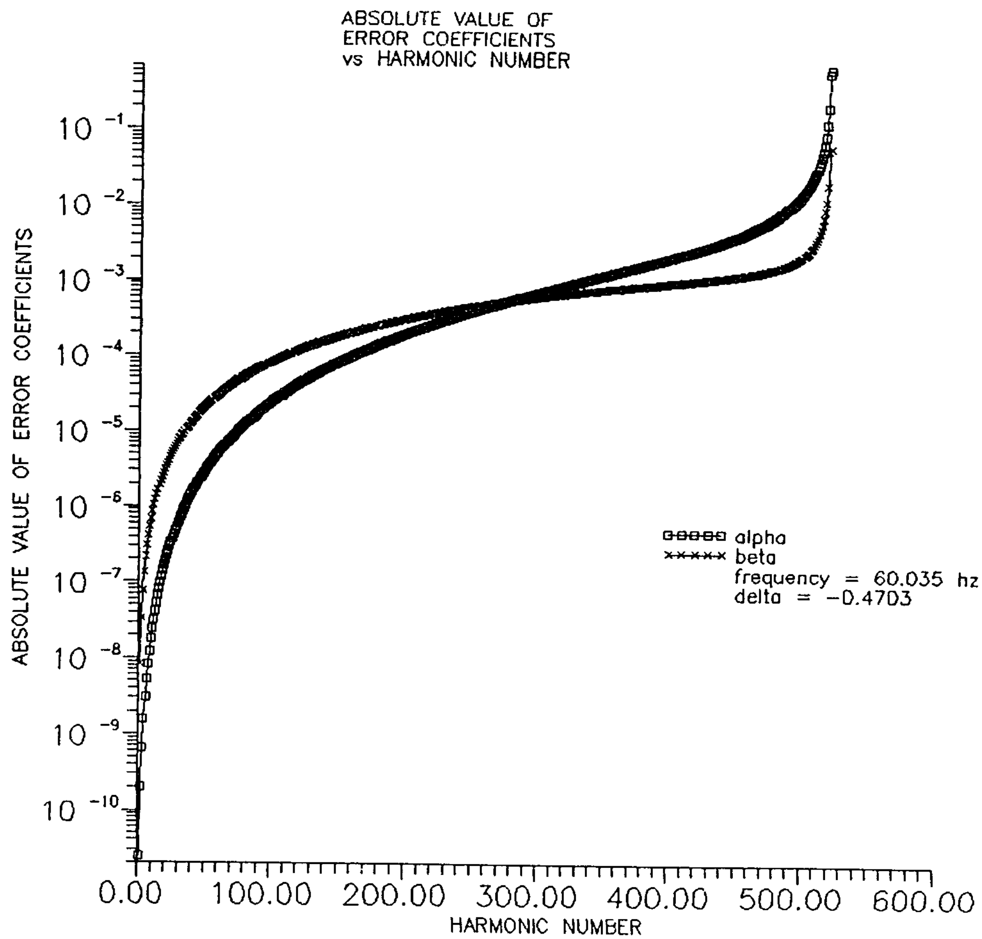

FIG. 2.11 GRAPH OF ABSOLUTE VALUE OF ERROR COEFFICIENTS vs HARMONIC NUMBER FOR A FREQUENCY OF 60.035 hz

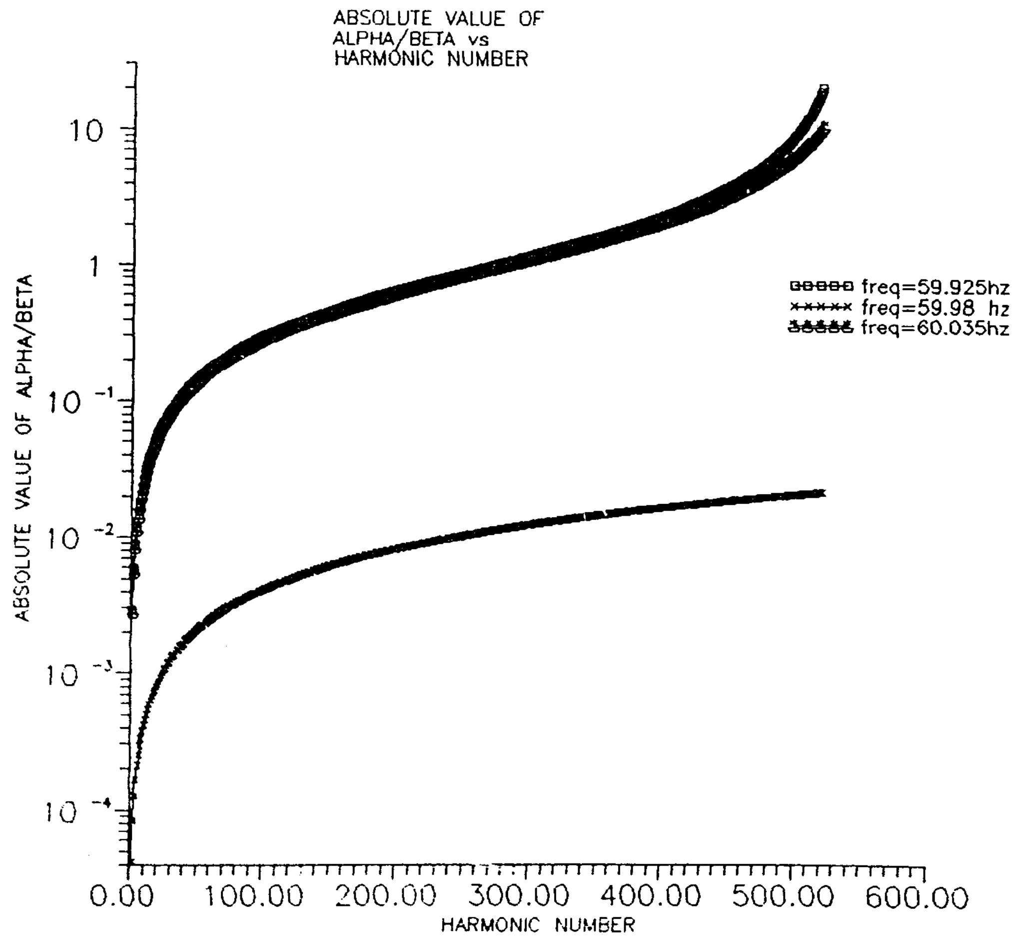

FIG. 2.12 GRAPH OF ABSOLUTE VALUE OF THE RATIO ALPHA/BETA vs HARMONIC NUMBER

obtained with the processing method used in this work. From fig. 2.6, it is observed that the beta error ratios are approximately equal irrespective of the order of the harmonic. Figs. 2.9 - 2.12 relate to the processing method used. The absolute values of alpha and beta (figs. 2.9 - 2.11) as well as their ratios (fig. 2.12), are obtained for harmonics from 1 to 521. The choice of frequencies of 59.925 hz, 59.98hz and 60.035 hz was made because they result in delta values close to 0.5, 0 and -0.5 respectively. Fig 2.12 indicates the absolute values of the ratios of alpha and beta obtained in figs. 2.9 - 2.11.

In determining the error contribution to a meter reading for a particular harmonic term, the worst case contribution will correspond to the largest value of either alpha or beta at that harmonic. As outlined previously, if the starting point of the acquisition cycle could be varied, then in principle the error term due to a particular harmonic could be minimized.

3. DIGITAL SAMPLING WATTMETER HARDWARE

The choice of the system hardware is governed by the following design considerations:

(a) A highly accurate system using off-the-shelf components.

(b) Adequate speed in handling 12-bit values and the mathematical manipulations to be performed on the latter.

An 8 Mhz zero-wait state 68000 microprocessor based system is employed for the current design. The instruction set of this processor incorporates various data manipulation instructions suitable for the mathematical computations involved. The hardware also facilitates the incorporation of debugging support which simplifies system maintenance.

A block diagram of the system hardware is shown in Fig. 3.1 (page 46). The system sub-units are; the CPU module, the I/O module, the timing controller module, the analog-to-digital interface module and the input-conditioning module. A more detailed block diagram showing the interconnecting signal lines are given in Fig. A.1.

3.1 CPU module

The block diagram of this module is shown in Fig. 3.2 (page 46). This circuit consists of the following :

a) the microprocessor circuit,

b) the memory circuit,

c) the interrupt and watchdog timer circuitry.

A more detailed block diagram showing the control signal interconnections is shown in the appendix as Fig. A.2.

3.1.1 Microprocessor circuitry

The block diagram of this circuit is shown in Fig. 3.3 (page 47). The heart of this circuitry is a 68000 microprocessor clocked by an 8 Mhz system clock [9]. The reset circuitry generates the required reset signal on power-up or following the assertion of the externally provided reset switch.

The system memory map shown in Fig. 3.4 (page 48), indicates a partial decoding of the 16 megabyte direct address space. This scheme results in a simple implementation of the decoding scheme required for the microprocessor peripheral devices. The I/O and A/board address spaces are further decoded as indicated in Fig. 3.5 (page 49).

The DTACK and VPA generation circuitry generate the required data transfer acknowledgement signals for asynchronous and synchronous data transfers respectively, between the microprocessor and external devices.

Data, address and control line buffers are provided for communication external to the microprocessor circuitry.

The detailed schematic of this circuitry is shown in

Fig. A.3 and Fig. A.4.

3.1.2 Memory circuitry

The block diagram of this circuitry is shown in Fig. 3.6 (page 50). The 16-bit data bus of the microprocessor necessitates the use of a pair of byte-wide memory devices. Two 8 kB byte-wide EPROMs having a maximum access time of 200ns, as well as a pair of 32 kB byte-wide static RAMs having a maximum access time of 150ns are used. The EPROMs store the program code and permanent data. The latter devices are decoded at a base address of 000000 hex. This scheme results in a simplified reset and interrupt circuitry thus further simplifying the system design. The static rams are decoded at a base address of 200000 hex. The RAMS are employed for the storage of sampled data values as well as temporary variables. 8k words are allocated for the storage of the sampled data values. The detailed circuitry of this sub-unit is indicated in Fig. A.5.

3.1.3 Interrupt and Watchdog timer circuitry

The block diagram of this circuitry is shown in Fig. 3.7 (page 51). This circuitry encodes the interrupt signals and also generates the required interrupt acknowledgement signals. The interrupt encoding scheme results in the following priority scheme for the servicing of interrupts:

the GPIA circuitry, the keyboard controller circuitry and the PTM circuitry.

The watchdog timer asserts the microprocessor bus error line following the absence of a data transfer acknowledge signal. The assertion of the bus error line occurs 40 system clock cycles following the start of a data transfer sequence.

The detailed circuit diagram is shown in Fig. A.6.

3.2 I/O module

This section consists of five individual subunits as indicated in the block diagram of Fig. 3.8 (page 52). The microprocessor peripherals in this section interface with the 68000 microprocessor via the synchronous control signal lines. A more detailed block diagram showing the control signal interconnections is shown in the appendix as Fig. A.7.

The PTM subunit is centered around the 6840 programmable timer module. This unit serves two main functions. It is employed in the detection of the presence of an AC or DC signal on the trigger input line. This simple scheme enables the use of an interrupt approach through a device geared towards the efficient generation of interrupts. Secondly, the timer module is employed in the determination of the trigger input line signal frequency. The detailed circuit diagram of this circuit is shown in Fig. A.8.

The PIA circuitry is centered around the 6821 peripheral interface adapter IC. Both peripheral ports of the latter device are configured as outputs. In essence, this circuitry functions as a microprocessor controlled output latch. The detailed circuit diagram of this circuitry is shown in Fig. A.9

The keyboard circuitry is centered around the Intel 8279 programmable keyboard and display controller IC [7]. The main function of this circuitry is to provide the necessary interface between the microprocessor and the input terminal. This circuit also decodes and debounces the keypad switches. The 4 by 4 keypad matrix and a single pole double throw switch result in 32 unique keypad combinations for the various wattmeter functions. An interrupt driven scheme is employed in determining a key closure. The detailed circuit diagram of this section is shown in Fig. A.10.

The display circuitry uses 2 devices for display of the program results as well as the system status. The primary unit is a 4 line by 20 column LCD display. In addition, a 10 segment tri-colour bargraph is also used. The various segments of the latter device being labelled 0 to 9. The LCD display is connected to the microprocessor as a memory mapped device at address 6C0000 hex. The LCD display unit includes an onboard character generator ROM. The latter calibration terminals under software control. The sub-menu of Fig. 4.11. is next displayed. The user next selects the required channel. This is followed by the initialization of the PTM as well as the timing controller. The acquisition routine is next executed. Prior to the execution of the acquisition routine, the keypad is disabled through the PIA and enabled following the completion of the data acquisition. The multiplication, summation and square-root computation routines are then employed in determining the root-mean-square value of the acquired data. The results are subsequently displayed on the LCD display unit. The displayed values are then stored in the storage location in ram provided the storage of data has previously been enabled. The above procedure is executed until another command is entered. Following that, the system mode switch is connected to channels 1 to 4 and program execution returns to the main menu.

4.3.13 Module 10

The flow diagram of this module is shown in Fig. 4.16 (page 93). This module is executed when an 'A' is processed by the command preprocessor. This module uses the DC acquisition, display and DC computation routines. The DC acquisition routine results in the acquisition of 1024 samples. The DC computation routines utilize the trapezoidal rule procedure mentioned in section 2.3.2. The purpose of this module is to perform a further trimming of the digital value corresponding to a zero input. This is achieved by adding the average of the DC values obtained for 256 acquisition cycles to the default zero value. Following a system power-up, the default zero value is 800hex. On entering this module, the sub-menu of Fig. 4.11. is displayed. Thus the user can select the channel for recalibration. This module has the advantage of enabling the system to be periodically readjusted to its optimum state in software. This is easily performed in a programmable mode of system operation.

4.3.14 Module 11

The flow diagram of this module is shown in Fig. 4.17 (page 94). This module is executed when a 'B' is processed by the command preprocessor and uses the acquisition, display and computation routines. The purpose of this module is to compute and display the voltage, current , power and power factor depending on the mode selected. On entering this module, the following sub-menu is displayed on the LCD unit.

PRESS:

1 FOR PHASE 1

2 FOR PHASE 2

3 FOR 3-PHASE

Following the user's selection, the PTM and timing controller are initialized. The presence or absence of an AC signal is next determined by software. The presence of an AC signal results in the use of the AC mode routines. The absence of the latter signal results in the use of the DC routines. In the AC single-phase mode, following the acquisition of the samples, the voltage, current and power are computed using the required scale factors. The power factor is subsequently computed and all four parameters are displayed. In the 3-phase mode, the power for both phases is computed and displayed. In addition, the power factor assuming a 3-phase three-line balanced system, is computed and displayed. In the DC mode, following the acquisition of the samples, the voltage, current, power are computed using the required scale factors. The above procedure is re-executed till another command is entered.

4.3.15 Module 12

The flow diagram of this module is shown in Fig. 4.18 (page 95). This module is executed when a 'C' is processed by the command preprocessor and uses the acquisition, display and computation routines. This module uses a signal on the calibration terminals. The average of the sampled values obtained during an acquisition cycle are computed and displayed in hexadecimal notation. This procedure serves as an aid to calibrating the device.

4.3.16 Module 13

The flow diagram of this module is shown in Fig. 4.19 (page 96). This module is executed when a 'D' is processed by the command preprocessor and uses the display and data entry mode routines. On entering this module, the user enters a 4-digit number acquisition cycle value. This mode of system operation works with the following keypad commands: 9,B,C and F. The acquisition cycle value is simply the number of acquisition cycles to be displayed in the required module. A zero acquisition cycle value corresponds to the continuous acquisition case. On power-up or following a system reset, the acquisition value is zero.

4.3.17 Module 14

The flow diagram of this module is shown in Fig. 4.20 (page 97). This module is executed when an 'E' is processed by the command preprocessor and uses the display and data entry mode routines. On entering this module, the user enters a 4-digit number delay value. This mode of system operation works with the following keypad commands: 9,B,C and F.The delay value simply corresponds to the number of results computed between successive display updates. Once the meter is operational, the delay value can only be changed by executing this module. On power-up or following a system reset, the delay value is zero.

4.3.18 Module 15

The flow diagram of this module is shown in Fig. 4.21 (page 98). This module is executed when an 'F' is processed by the command preprocessor and uses the computation and display routines. The module computes and displays the frequency of the trigger signal. In addition, the maximum and minimum frequencies computed since entering this module are displayed. The system utilizes one of the timers of the 6840 PTM configured in the frequency measurement mode. A software polling scheme of the internal interrupt bit is employed in determining the counter value corresponding to one cycle of the trigger input signal. This results in a lower measurable frequency limit than can be obtained using a hardware interrupt scheme. The latter improvement is due to the higher interrupt latency.

4.3.19 Module 16

The flow diagram of this module is shown in Fig. 4.22 (page 99). This module is executed when the command '^O' is processed by the command preprocessor and uses the display, the scale factor and data entry mode routines. This module enables the user to enter the current shunt value in use. On entering this module, the following sub-menu is displayed.

^0: ENTER SHUNT VALS

1: FOR CHANNEL 1

2: FOR CHANNEL 2

0 TO EXIT

The user performs the required selection and enters the shunt value. As in all menus and sub-menus, a wrong selection is simply ignored.

4.3.20 Module 17

The flow diagram of this module is shown in Fig. 4.23 (page 100). This module is executed when the command '^1' is processed by the command preprocessor and uses the display routines. This module indicates the size of the stored data in terms of the number of lines they would occupy on the display. The result is displayed for approximately 2.5 seconds. Program execution is subsequently returned to the main menu.

4.3.21 Module 18

The flow diagram of this module is shown in Fig. 4.24 (page 101). This module is executed when the command '^2' is processed by the command preprocessor and uses the display routines. This module displays the size of the stored data in terms of the number of lines they would occupy on the display for approximately 2.5 seconds. In addition, the actual stored data is also displayed for approximately 5 seconds. This procedure is executed until either a new command is entered or all the data is displayed. Program execution is subsequently returned to the main menu.

4.3.22 Module 19

The flow diagram of this module is shown in Fig. 4.25 (page 102). This module is executed when the command '^3' is processed by the command preprocessor and uses the bargraph display routine. This module enables the storage of the appropriate data and indicates this state on LED 5 of the bargraph display. Program execution is subsequently returned to the main menu.

4.3.23 Module 20

The flow diagram of this module is shown in Fig. 4.26 (page 102). This module is executed when the command '^4' is processed by the command preprocessor and uses the display and program entry routines. This module is the keypad programming mode referred to in Module 1. The user enters a program command string corresponding to the GPIB commands. On conclusion, the system automatically executes the commands sequentially.

4.3.24 Module 21

The flow diagram of this module is shown in Fig. 4.27 (page 103). This module is executed when the command '^5' is processed by the command preprocessor. This command is one of the data entry mode routine commands. When this command is entered during data entry, the cursor is backspaced. This enables the user to correct any of the previously entered digits on the LCD display.

4.3.25 Module 22

The flow diagram of this module is shown in Fig. 4.28 (page 103). This module is executed when the command '^6' is processed by the command preprocessor. This command is one of the data entry mode routine commands. This command indicates to the data entry routine, the termination of data entry of digits on the LCD display.

4.3.26 Module 23

The flow diagram of this module is shown in Fig. 4.29 (page 104). This module is executed when the command '^7' is processed by the command preprocessor and uses the LCD display and data entry routines. The primary use of this module is the changing of the numerators and denominators of the voltage scale factors as well as the numerators of the current scale factors. This module is relevant when new scale factors need to be programmed as a result of the replacement or aging of components. In this module the user is prompted to enter four digit numbers for voltage scale factors followed by two 4-digit numbers for the current scale factors. Program execution then returns to the main menu.

4.3.27 Module 24

The flow diagram of this module is shown in Fig. 4.30 (page 105). This module is executed when the command '^8' is processed by the command preprocessor. This module is basically an extension of module 10. The procedure executed by module 10, is performed automatically for all the channels. Thus the sub-menu of Fig. 4.11. is not displayed.

4.3.28 Module 25

The flow diagram of this module is shown in Fig. 4.31 (page 105). This module is executed when the command '^B' is processed by the command preprocessor and uses the input and display routines. The command results in the display of the penultimate menu.

4.3.29 Module 26

The flow diagram of this module is shown in Fig. 4.32 (page 106). This module is executed when the command '^C' is processed by the command preprocessor and uses the input and display routines. The command results in the display of the previous menu.

4.3.30 Module 27

The flow diagram of this module is shown in Fig. 4.33 (page 106). This module is executed when the command '^D' is processed by the command preprocessor. This command is one of the program entry mode routine commands. This command serves a dual function. It indicates the termination of program entry in the keypad programming mode as well as marking the end of the program string in both the keypad and GPIB programming modes.

4.3.31 Module 28

The flow diagram of this module is shown in Fig. 4.34 (page 107). This module is executed when the command '^E' is processed by the command preprocessor. This command is one of the program entry mode routine commands. When this command is entered during program entry, the cursor is backspaced. This enables the user to correct any of the previously entered program command string characters.

4.3.32 Module 29

The flow diagram of this module is shown in Fig. 4.35 (page 107). This module is executed when the command '^F' is processed by the command preprocessor and uses the input and display routines. The command simply results in the display of the next menu.

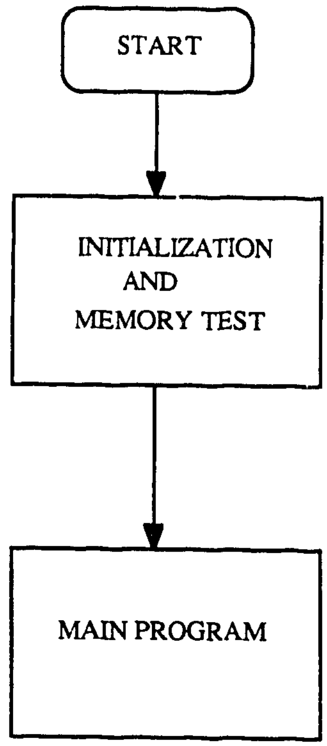

Fig. 4.1 Flowchart of the system software

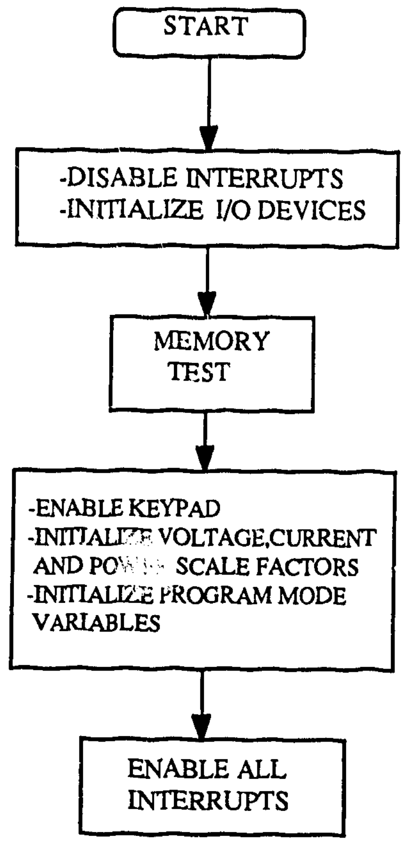

Fig. 4.2 Flow chart of the initialization and memory test

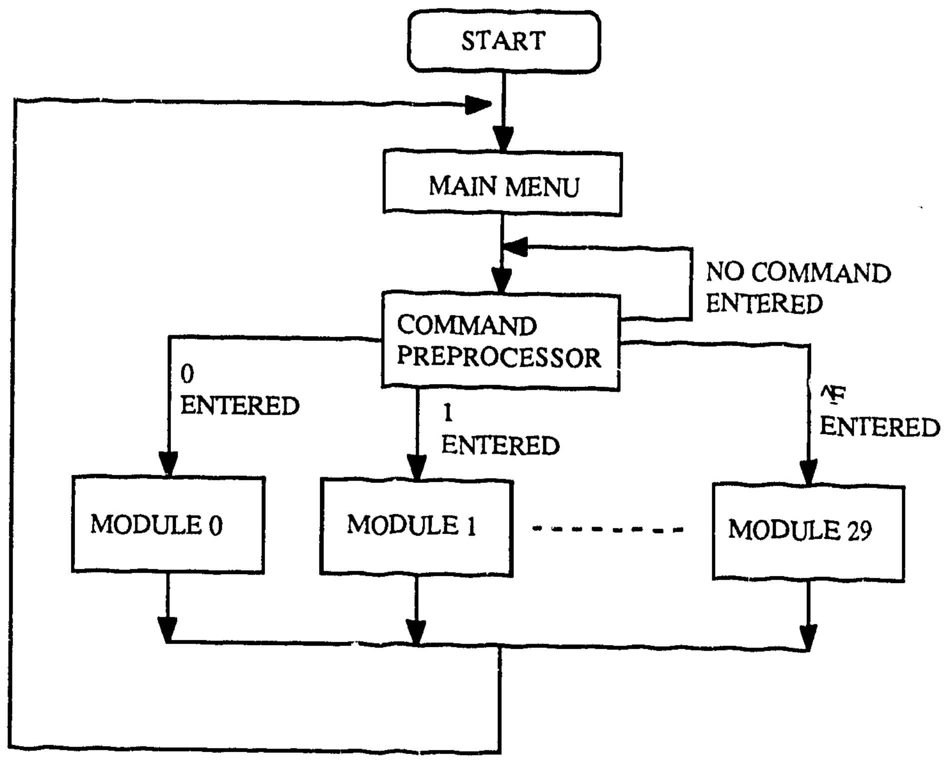

Fig. 4.3 Flowchart of the main program

METER READY

^F: NEXT MENU
^B: PREVIOUS MENU

A) MENU 0

0: RESET
1: DISPLAY ON/OFF
^C: RETURN TO
MENU

B) MENU 1

2: CHAN 1 HIGH/LOW
3: CHAN 3 HIGH/LOW
4: O_LOAD TEST ON/OFF
5: NUMERATOR (S/F)

C) MENU 2

6: DENOMINATOR(S/F)
7: TRIG SRC CHAN1/2
8: CLEAR DATA STORE
9: CALIBRATE RMS

D) MENU 3

A:RECAL B:SAMPLE
C: CALIBRATE
D: ACQ_SIZE
E: DELAY F:FREQ

E) MENU 4

^0: ENTER SHUNT VALS
^1: LINES IN RAM
^2: DISPLAY RAM DATA
^3: STORAGE ON/OFF

F) MENU 5

^4: KEYPAD PROGRAM
^5: CORRECT ENTRY
^6: END ENTRY
^7: CHANGE SC/FACTS

G) MENU 6

^8:RECAL ALL CHANS
^9:
^A:
^D: END PROG ^E: UNDO

H) MENU 7

Fig. 4.4 Detailed main menu displays

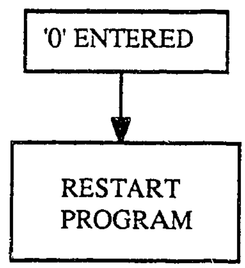

Fig. 4.5 Flowchart of module 0

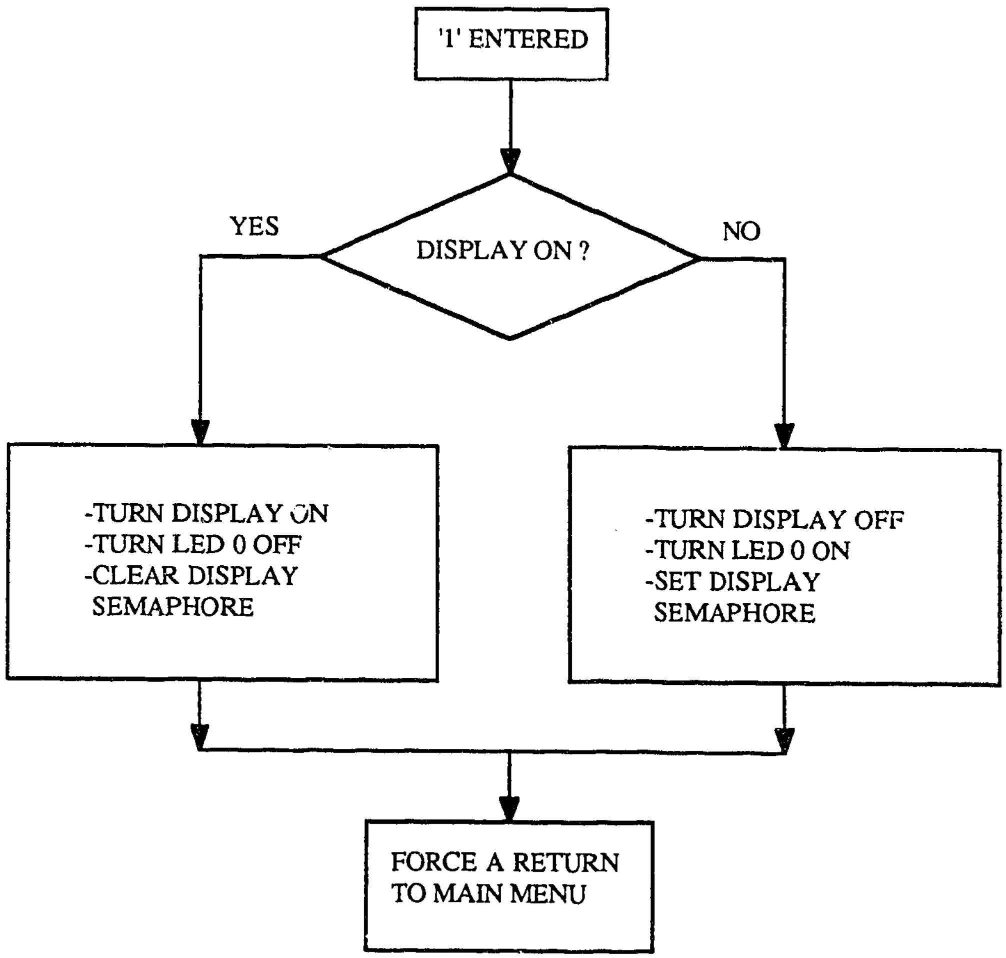

Fig. 4.6 Flowchart of module 1

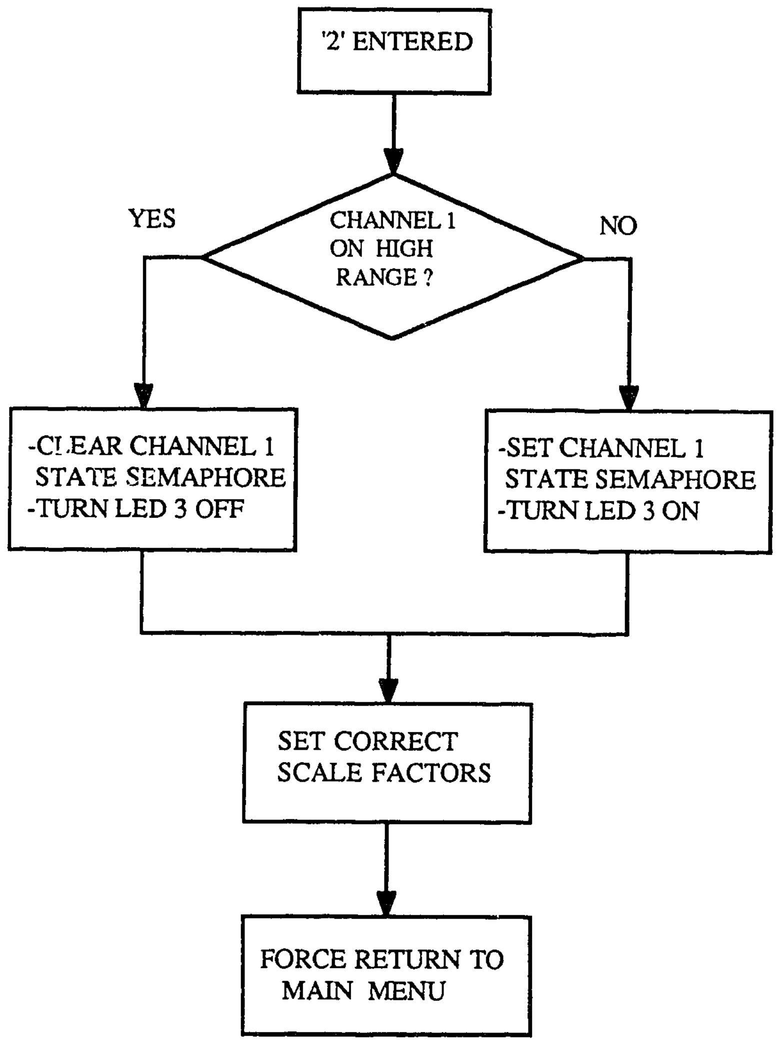

Fig. 4.7 Flowchart of module 2

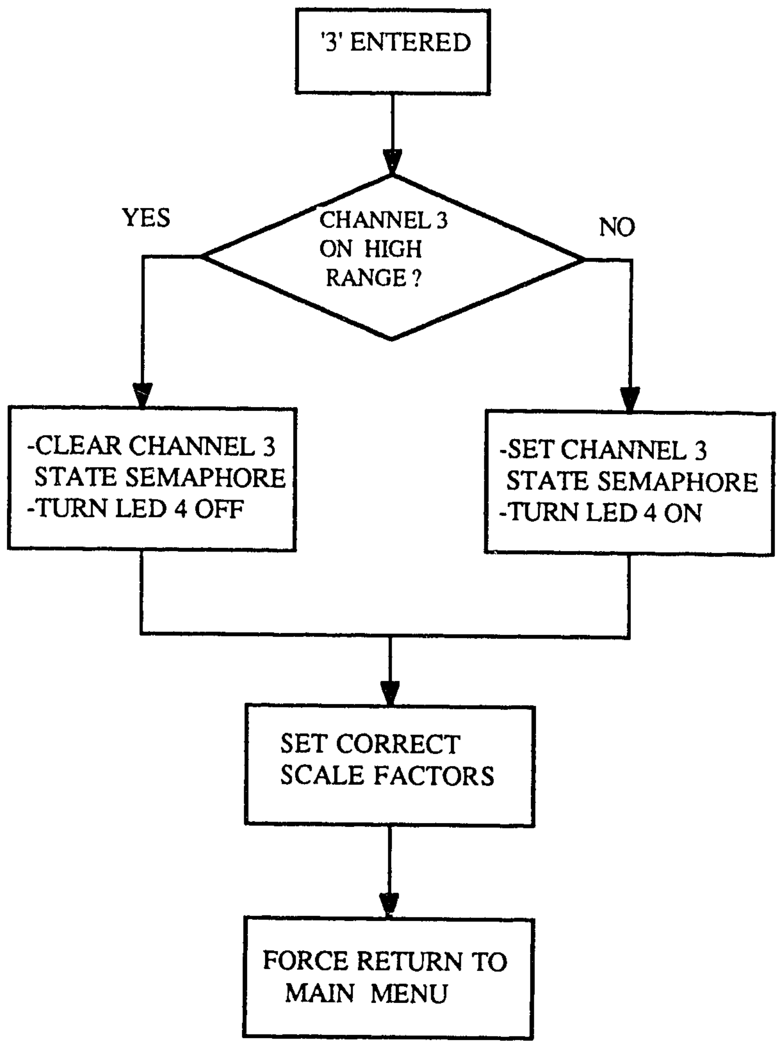

Fig. 4.8 Flowchart of module 3

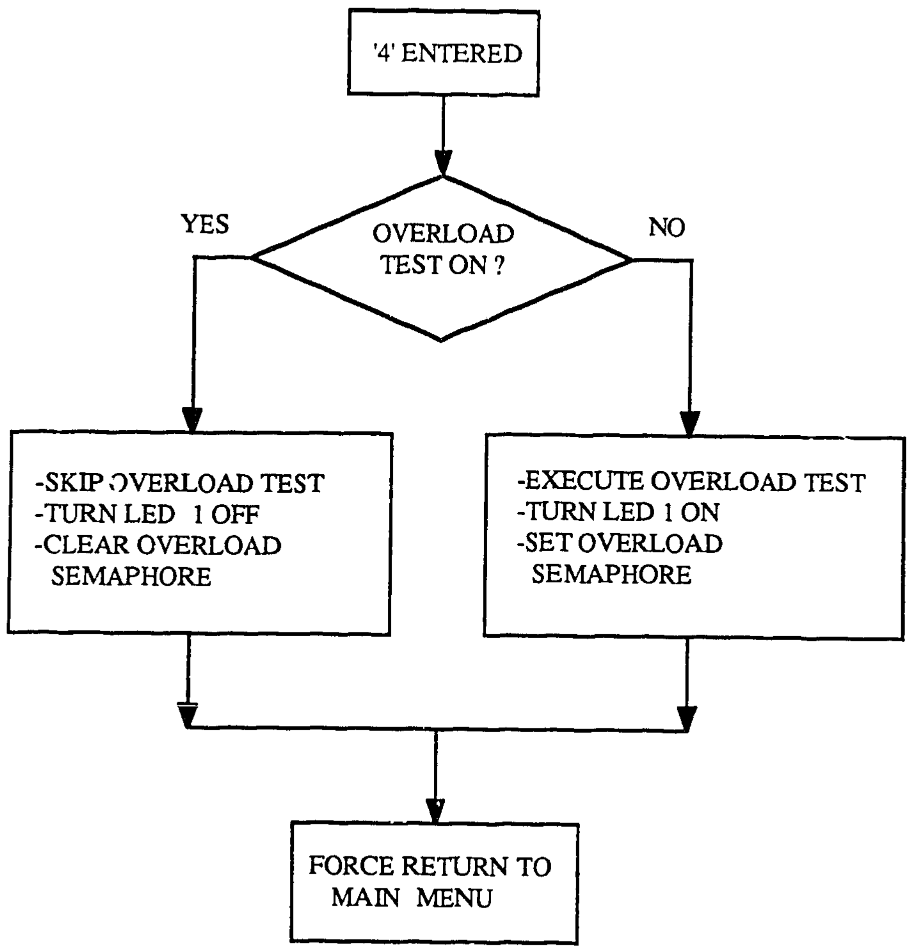

Fig. 4.9 Flowchart of module 4

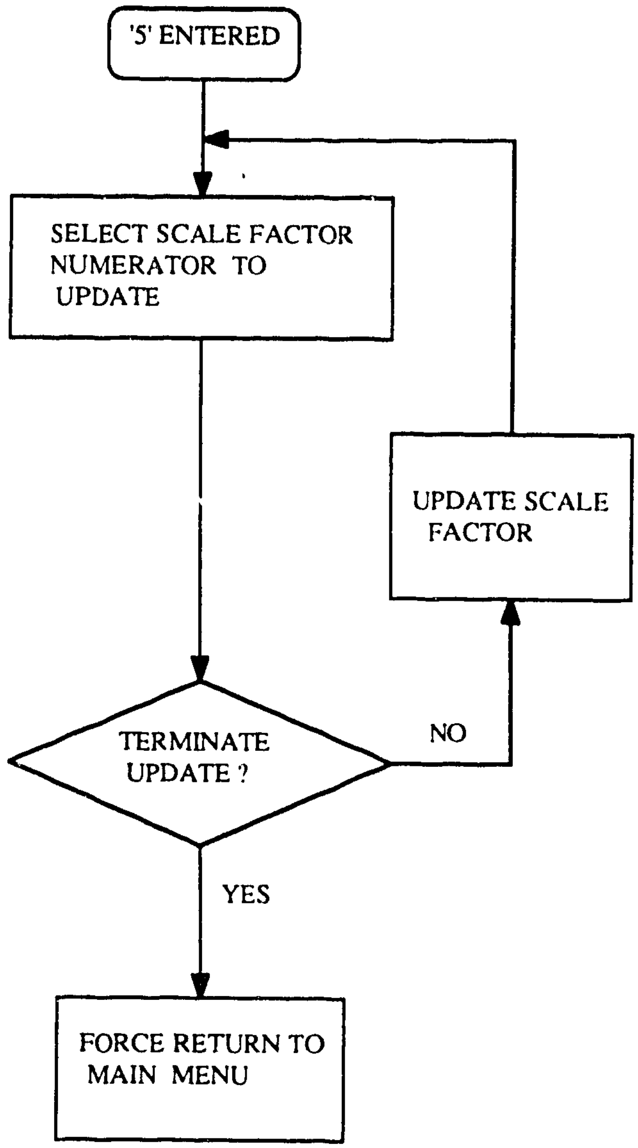

Fig. 4.10 Flowchart of module 5

1 FOR CHANNEL 1
2 FOR CHANNEL 2
3 FOR CHANNEL 3
4 FOR CHANNEL 4

Fig. 4.11 Channel selection sub-menu

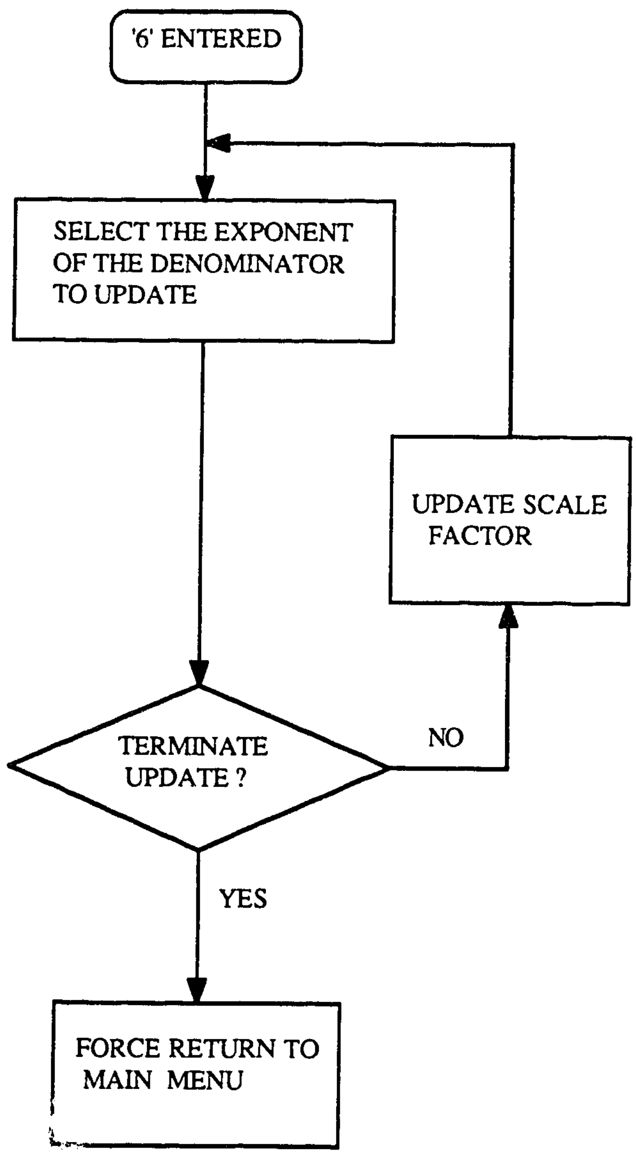

Fig. 4.12 Flowchart of module 6

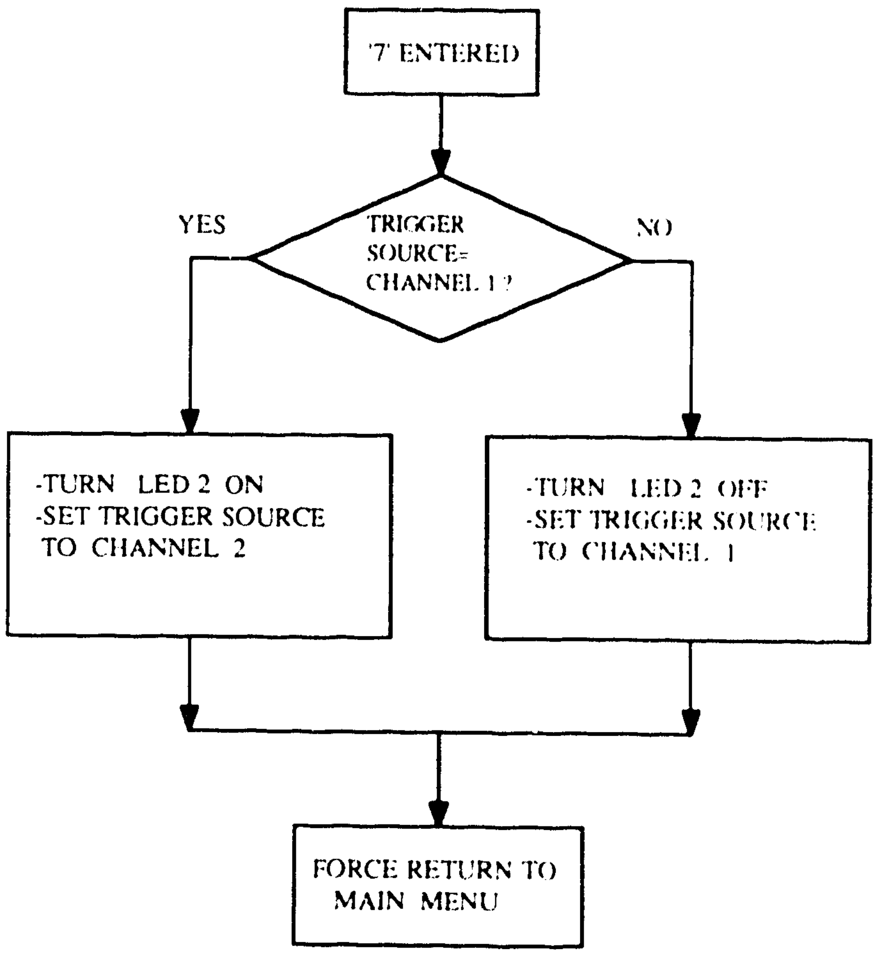

Fig. 4.13 Flowchart of module 7

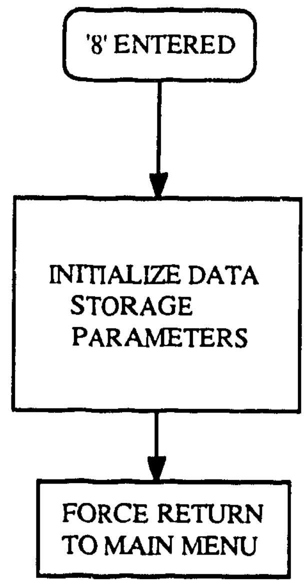

Fig. 4.14 Flowchart of module 8

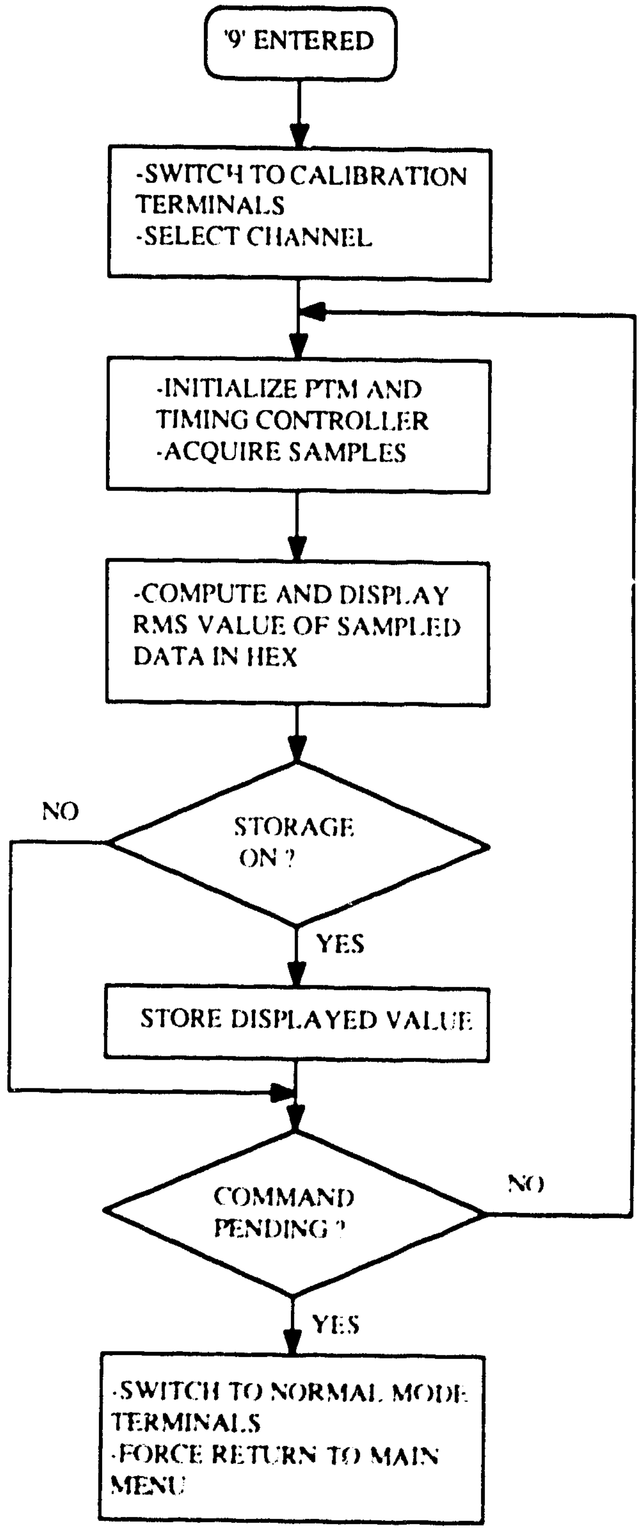

Fig. 4.15 Flowchart of module 9

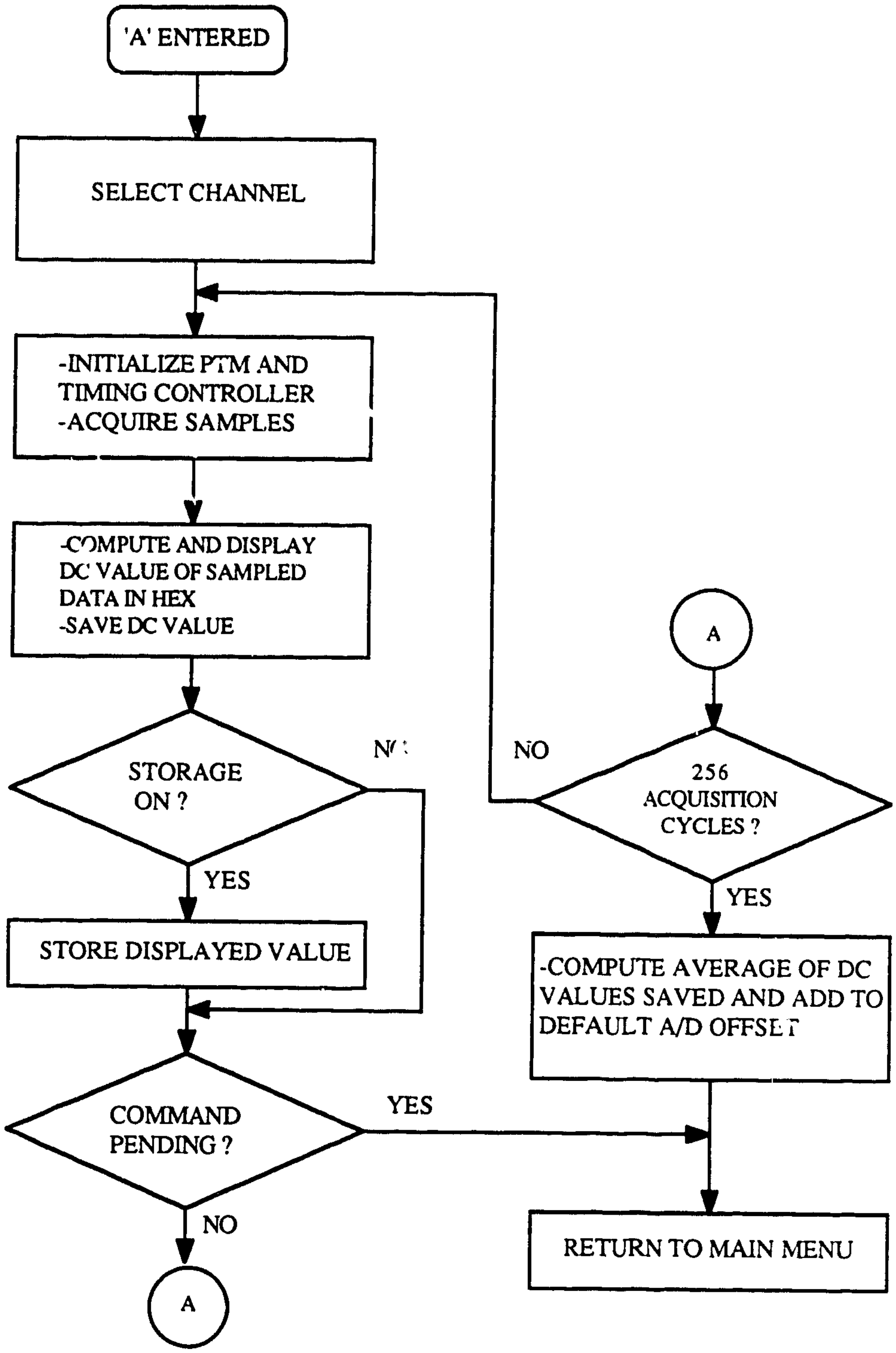

Fig. 4.16 Flowchart of module 10

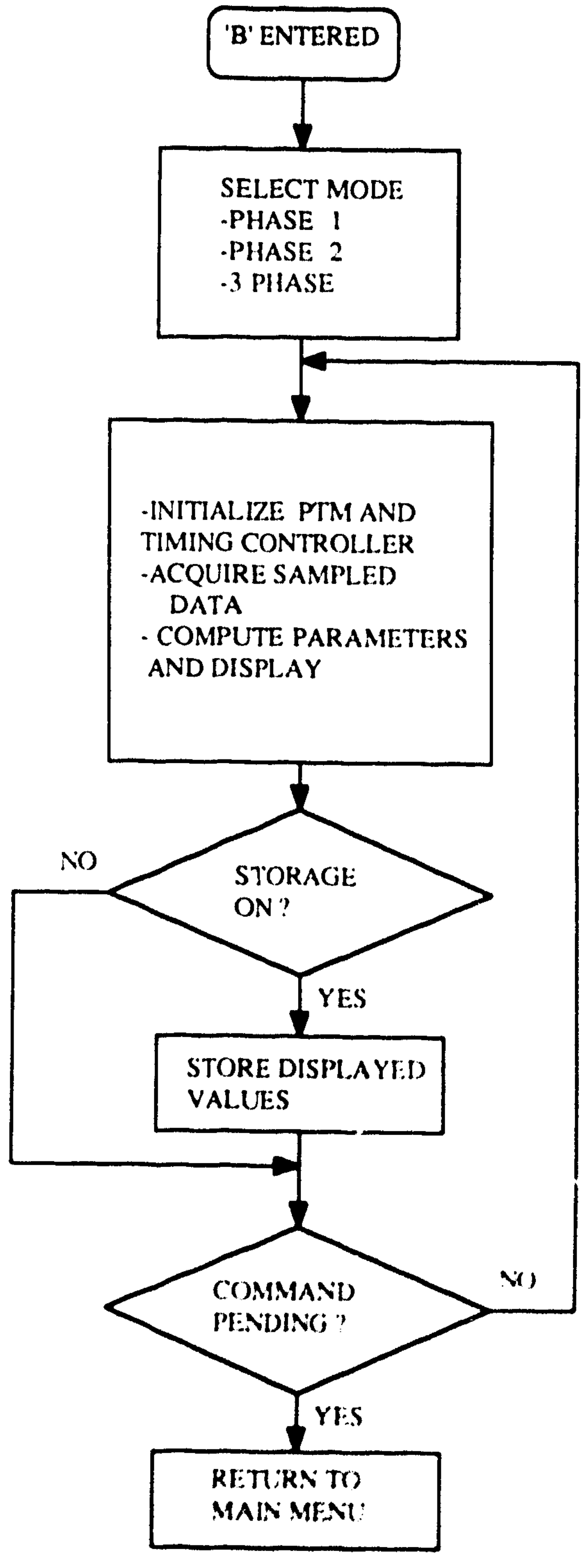

Fig. 4.17 Flowchart of module 11

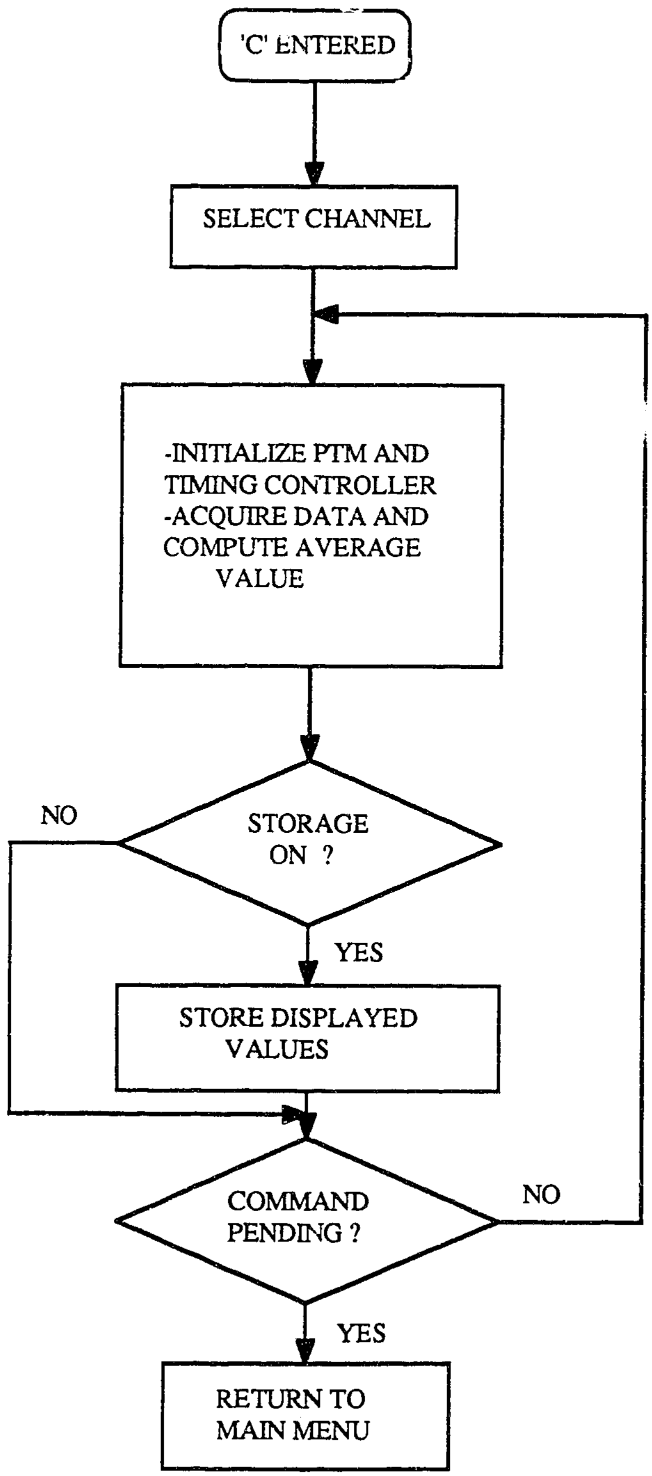

Fig. 4.18 Flowchart of module 12

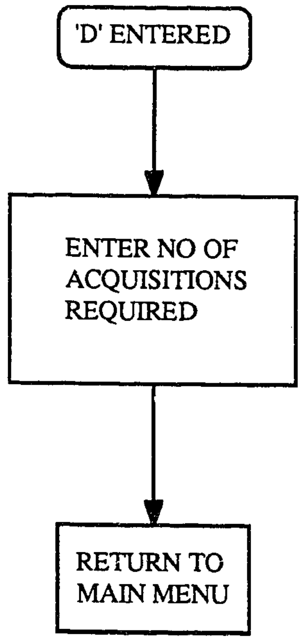

Fig. 4.19 Flowchart of module 13

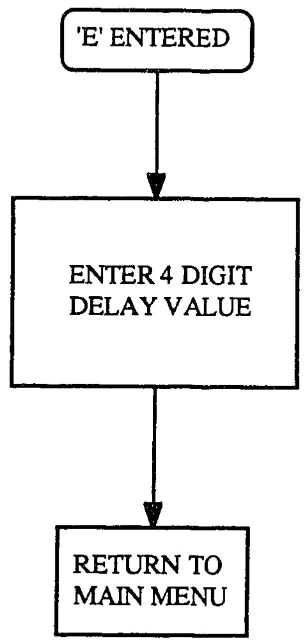

Fig. 4.20 Flowchart of module 14

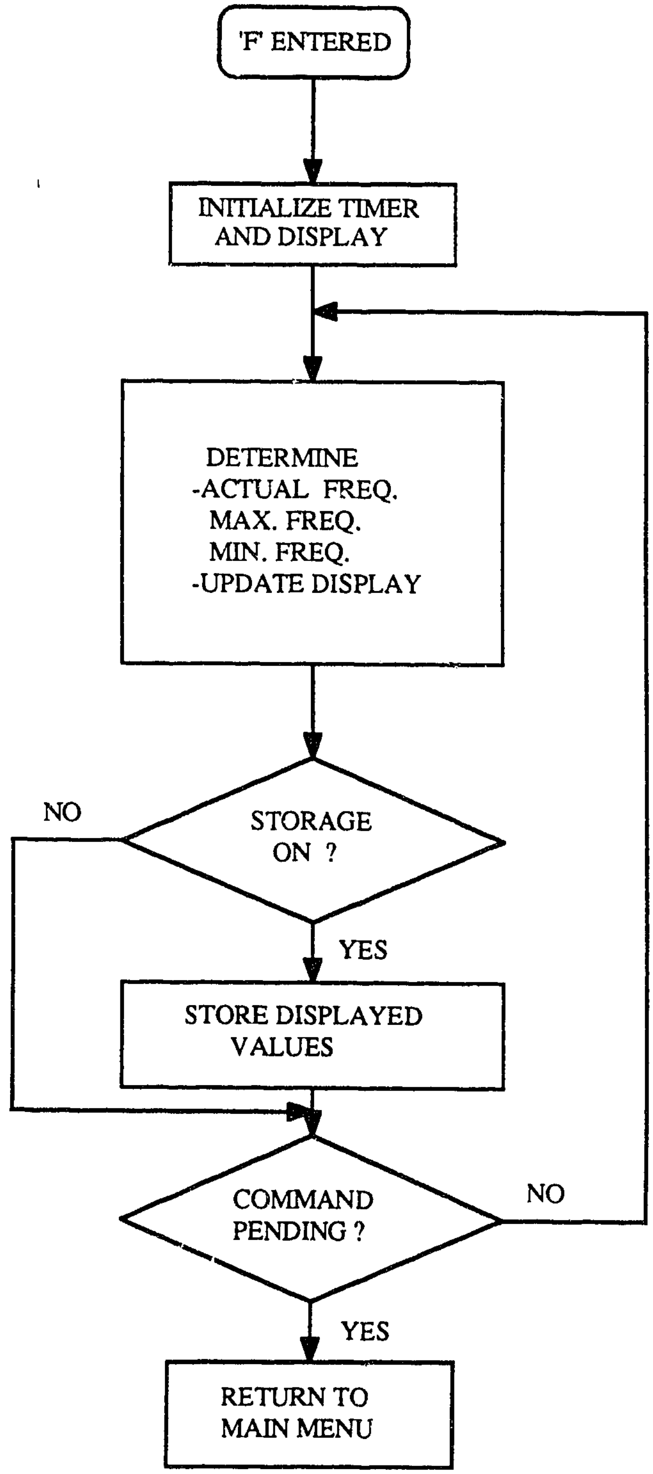

Fig. 4.21 Flowchart of module 15

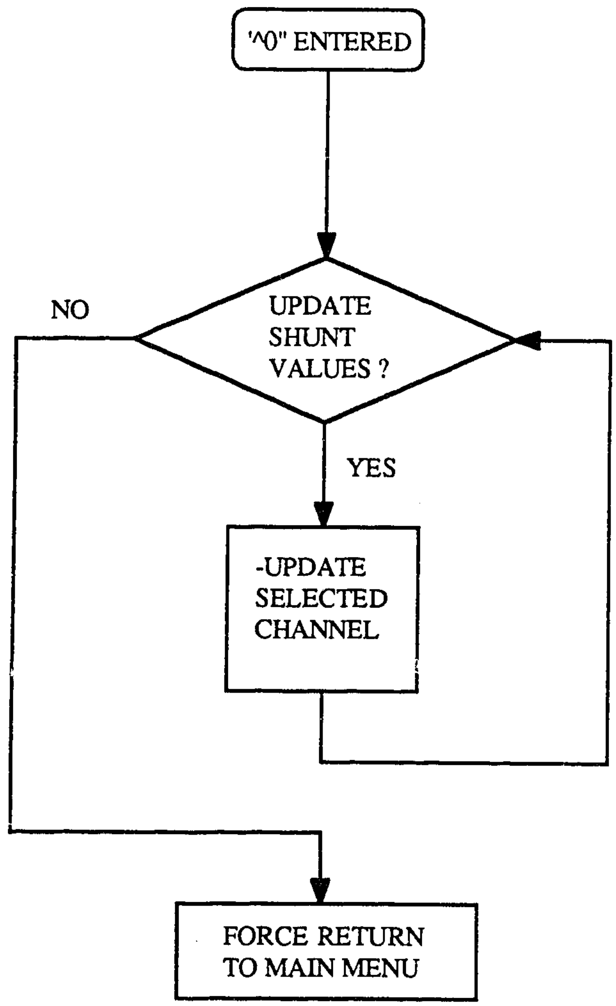

Fig. 4.22 Flowchart of module 16

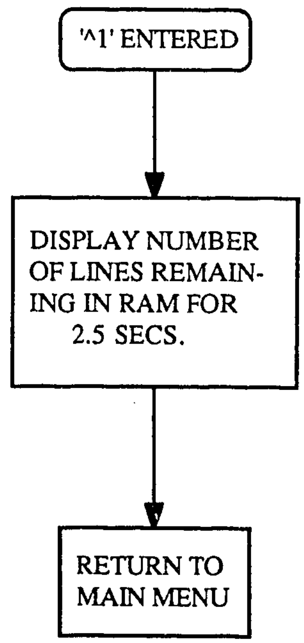

Fig. 4.23 Flowchart of module 17

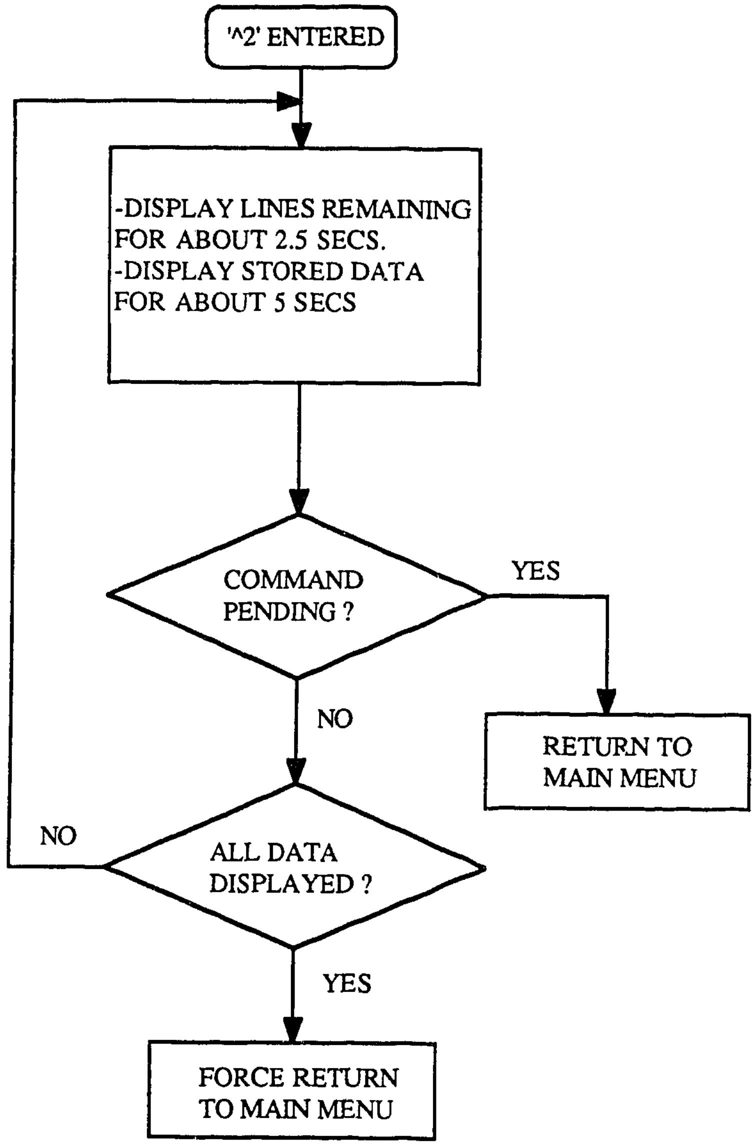

Fig. 4.24 Flowchart of module 18

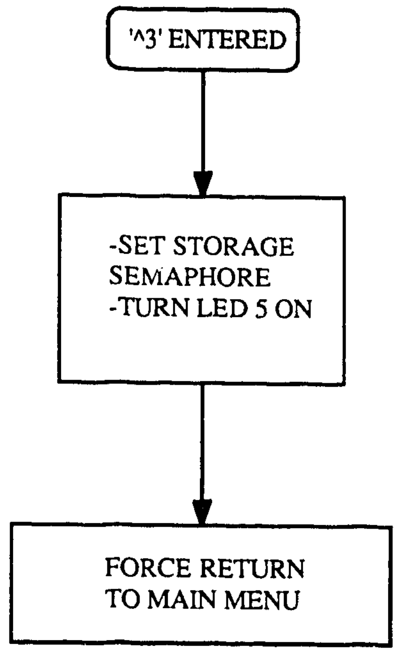

Fig. 4.25 Flowchart of module 19

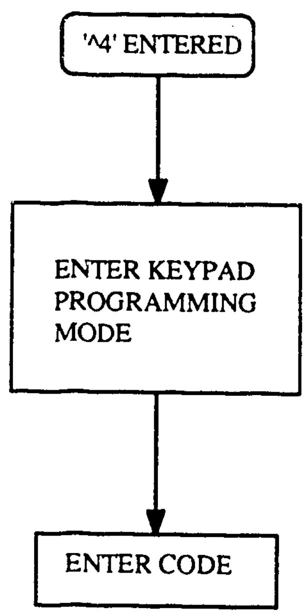

Fig. 4.26 Flowchart of module 20

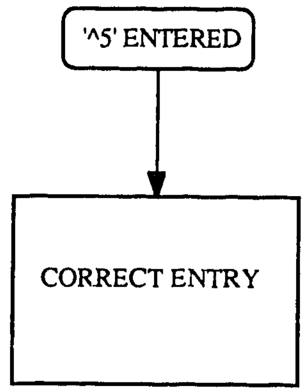

Fig. 4.27 Flowchart of module 21

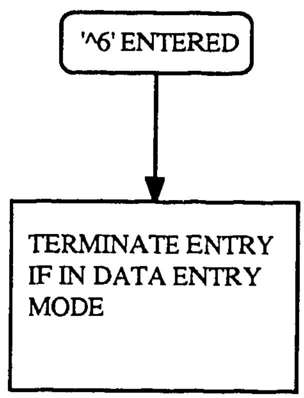

Fig. 4.28 Flowchart of module 22

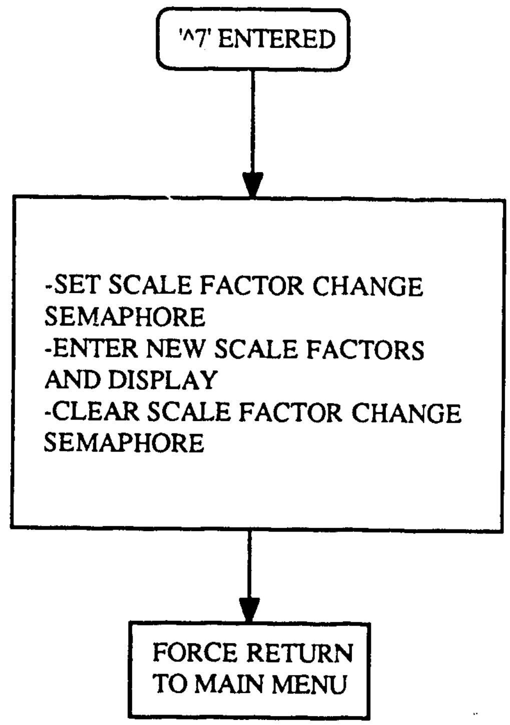

Fig. 4.29 Flowchart of module 23

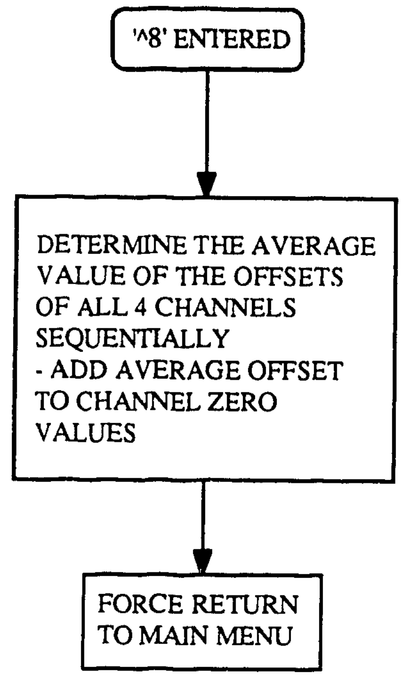

Fig. 4.30 Flowchart of module 24

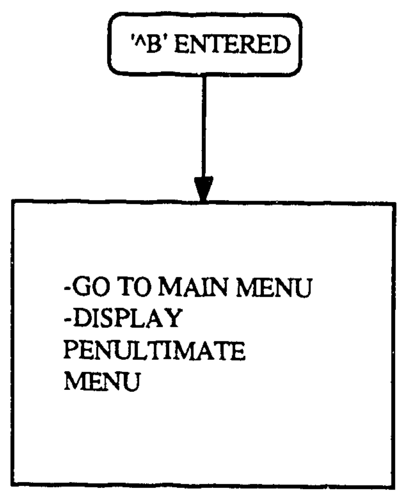

Fig. 4.31 Flowchart of module 25

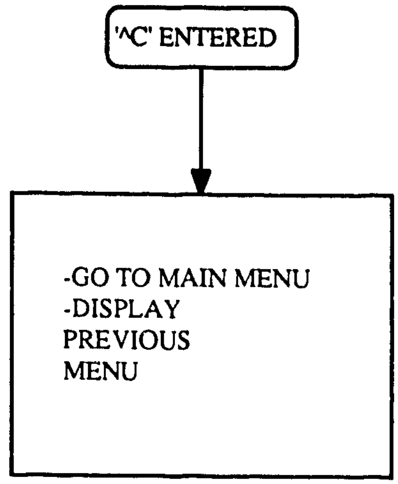

Fig. 4.32 Flowchart of module 26

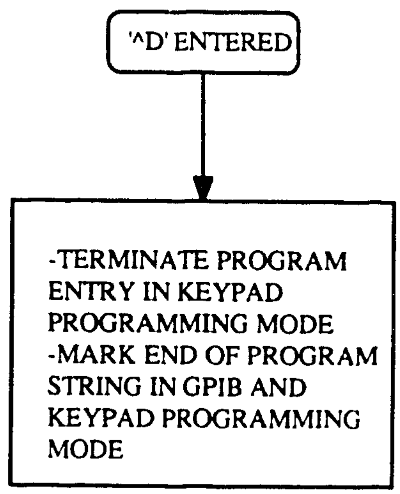

Fig. 4.33 Flowchart of module 27

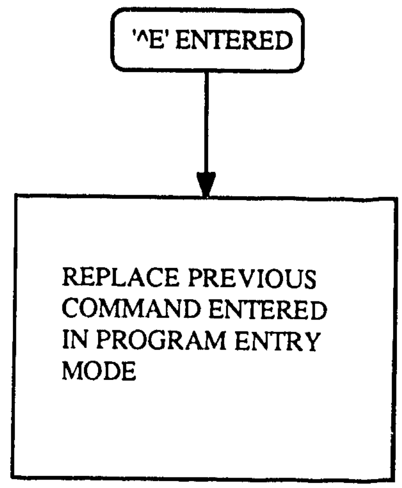

Fig. 4.34 Flowchart of module 28

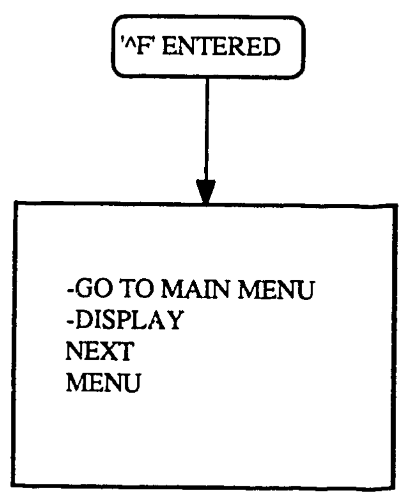

Fig. 4.35 Flowchart of module 29

5. SYSTEM ROUTINES

The program code utilizes various routines in the performance of it's tasks. These can be classified as; acquisition routines, computational routines and utility routines.

5.1 Acquisition routines

The main purpose of these routines is in transferring the digital outputs of the A/D converters to the storage media namely; the static ram ICs. During the transfer of the data, interrupts from the keypad are disabled and only re-enabled after the completion of the data transfer.

Detection of an AC or DC signal is performed using a software polling scheme of the relevant hardware lines.

In both AC and DC acquisition modes, following the issuance of a start of conversion signal, the microprocessor acquires the latched data. For each acquired 16-bit value, the respective channel 16-bit zero offset value is subtracted. The resulting two's complement value is subsequently stored in the static ram devices. All the above mentioned procedures are performed and completed well before the start of the next sampling interval. 1024 samples per channel are acquired in the DC mode. In addition to the above functions the AC mode acquisition routines determine the state of the sampling process during an AC acquisition cycle. This determination is performed

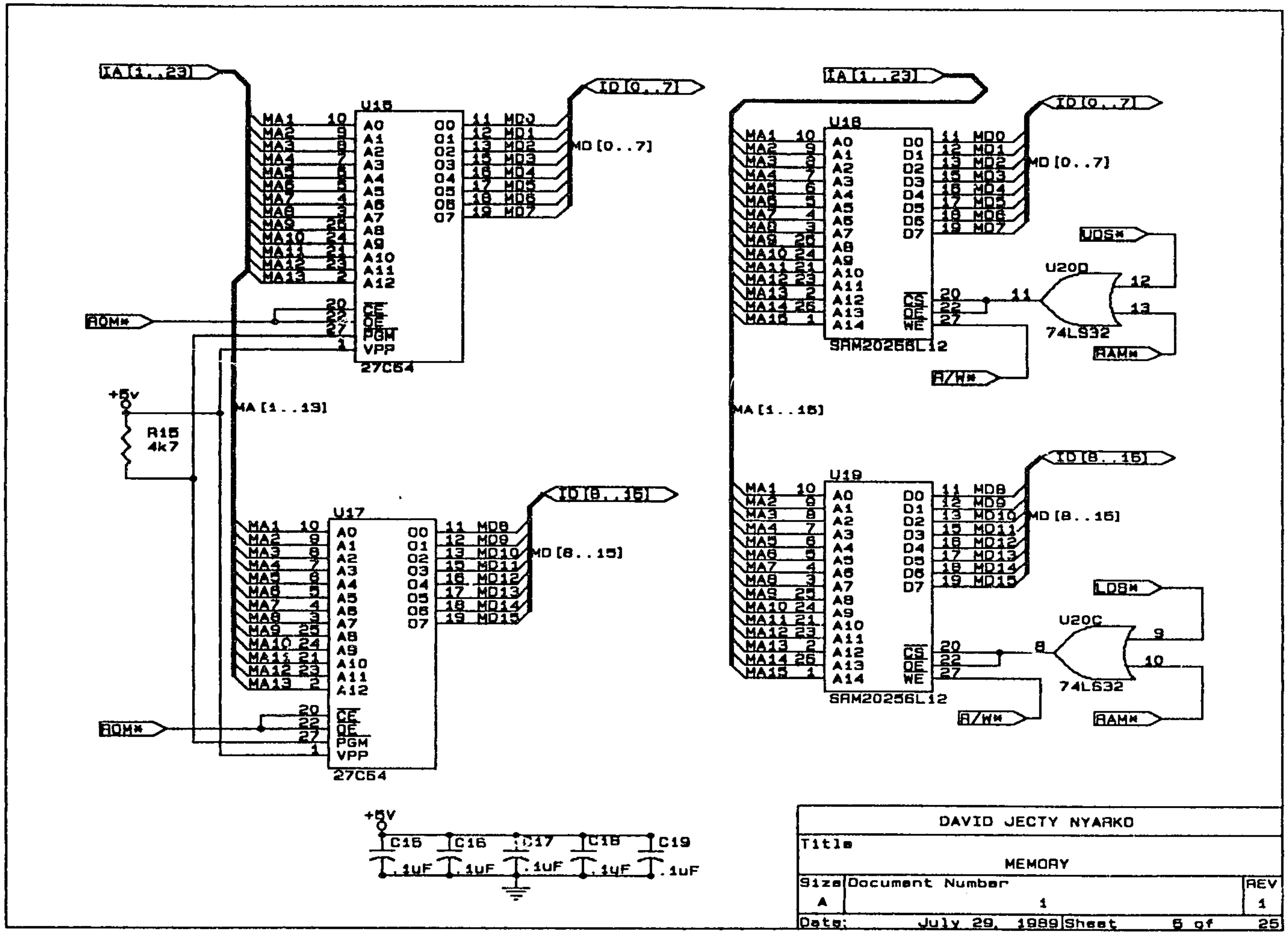

FIG. A.5 DETAILED SCHEMATIC OF MEMORY CIRCUITRY

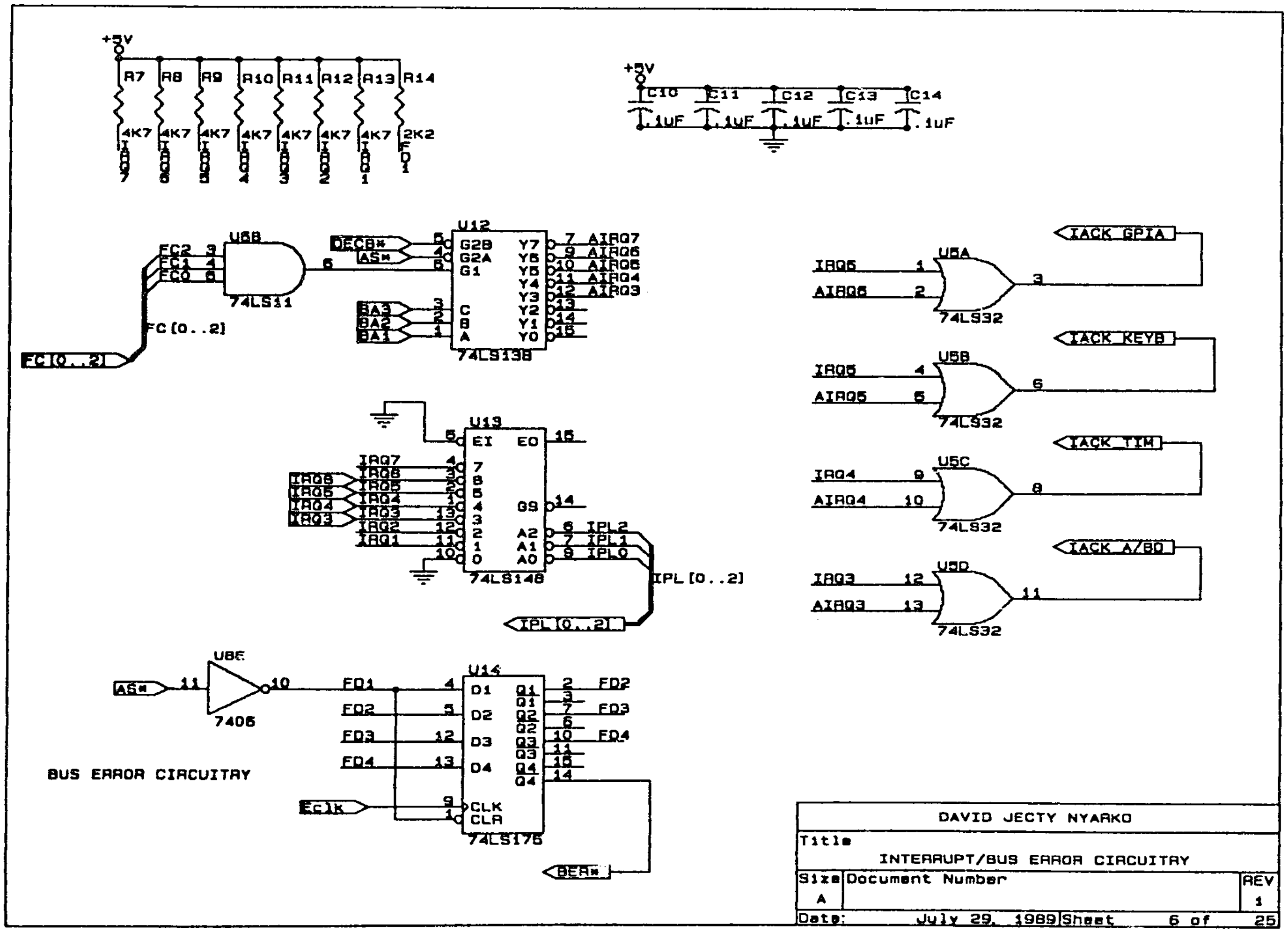

FIG. A.6 DETAILED SCHEMATIC OF INTERRUPT/BUS ERROR CIRCUITRY

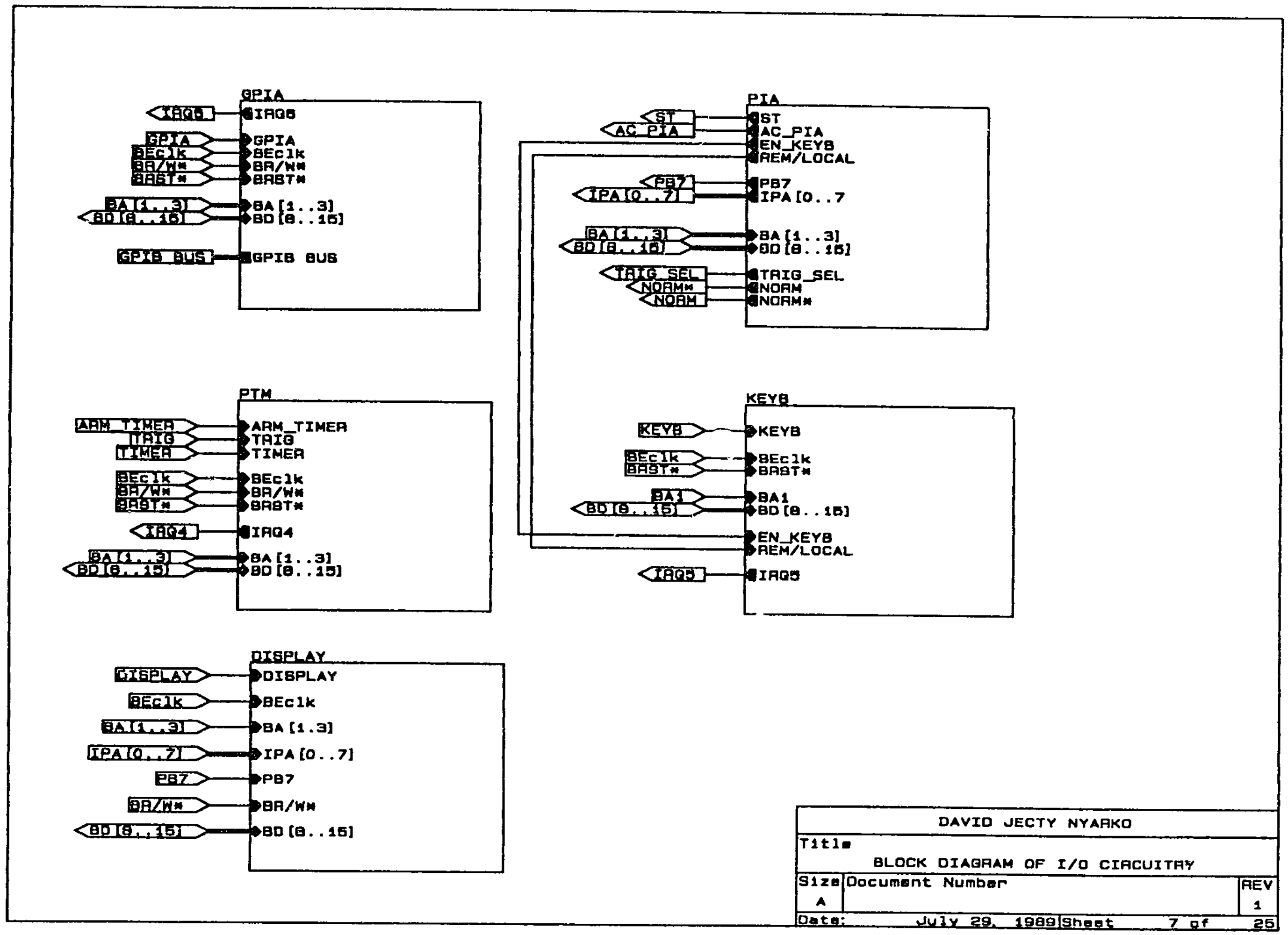

FIG. A.7 BLOCK DIAGRAM OF I/O CIRCUITRY

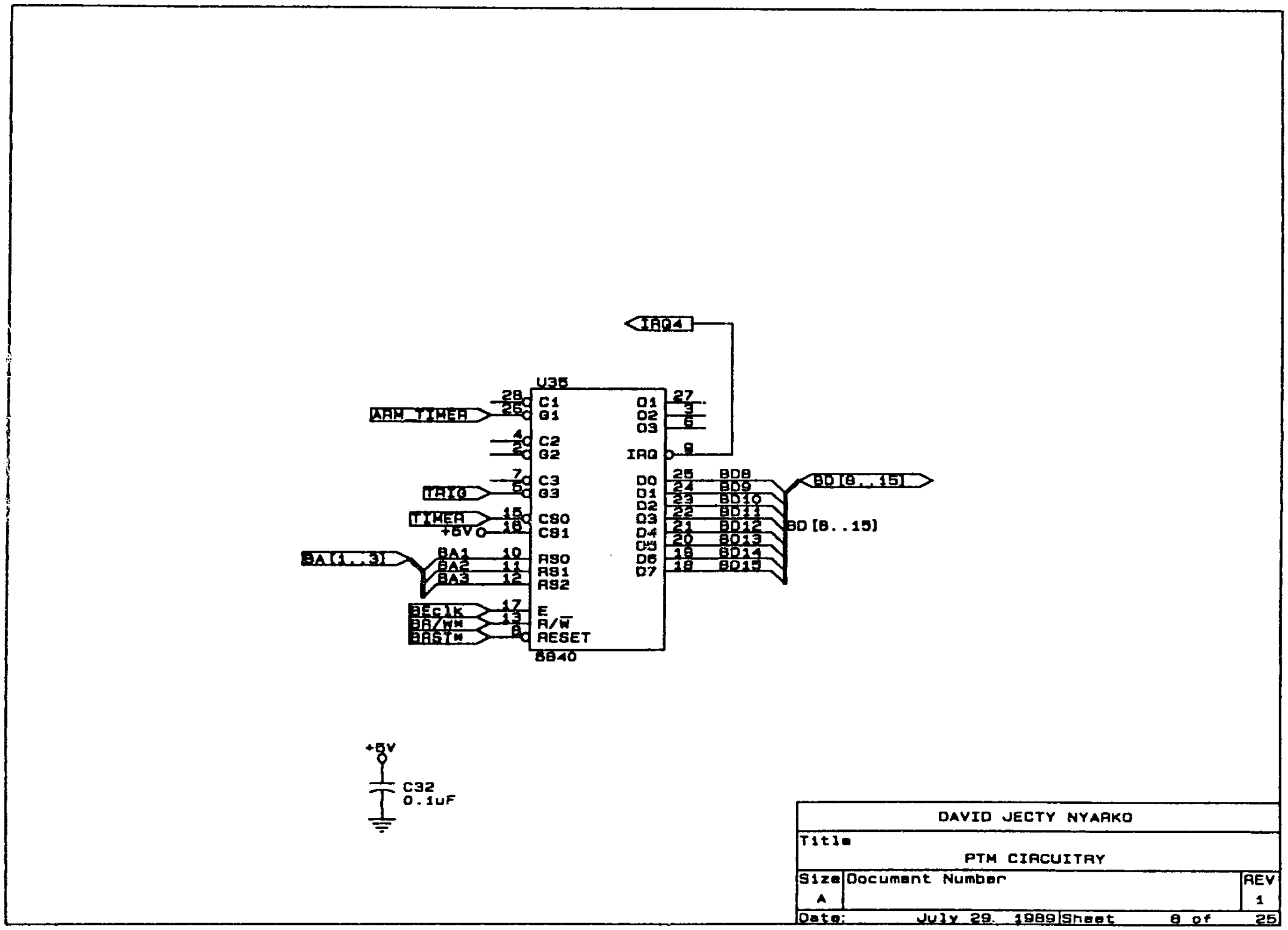

FIG. A.8 DETAILED SCHEMATIC OF PTM CIRCUITRY

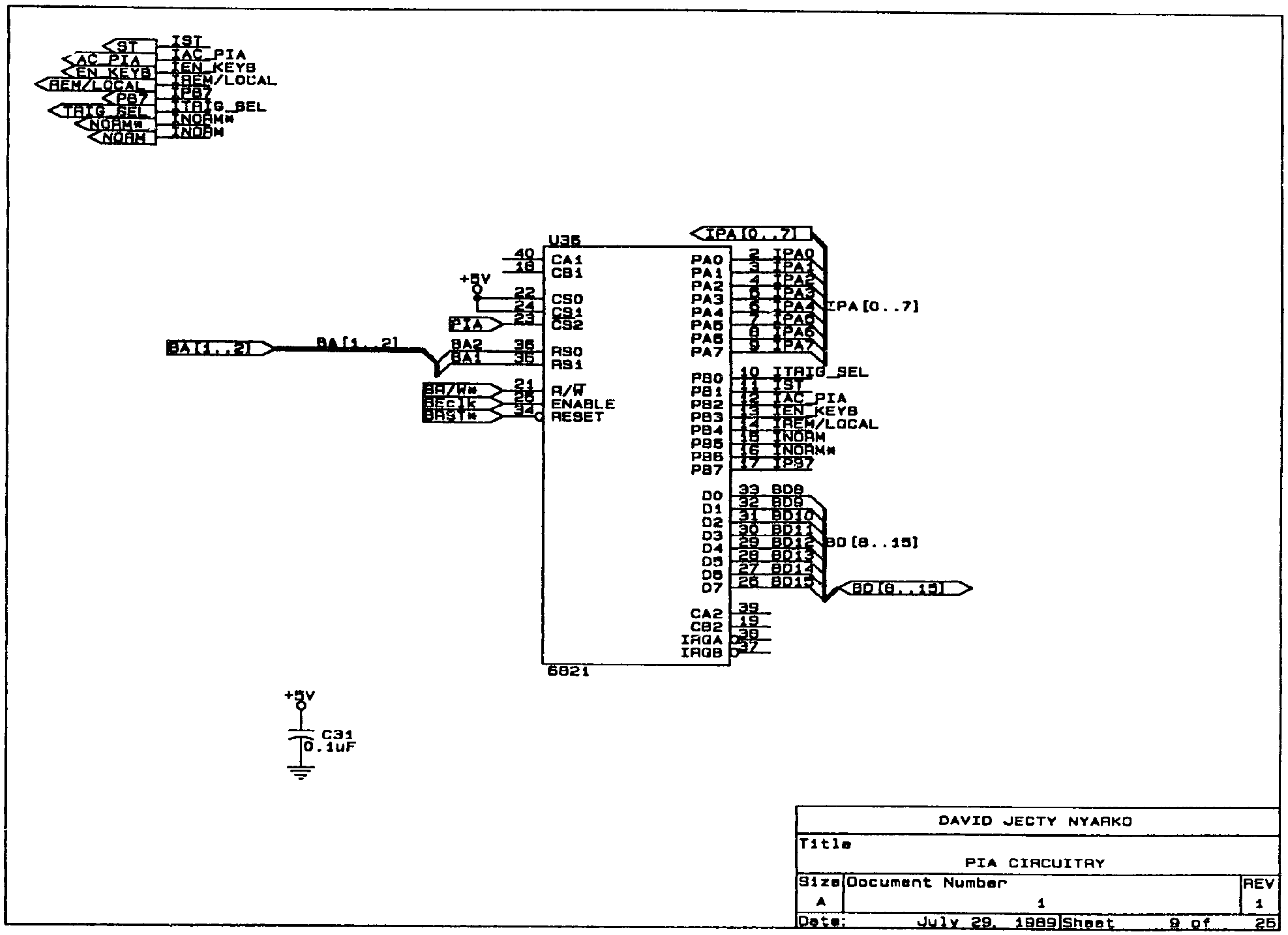

FIG. A.9 DETAILED SCHEMATIC OF PIA CIRCUITRY

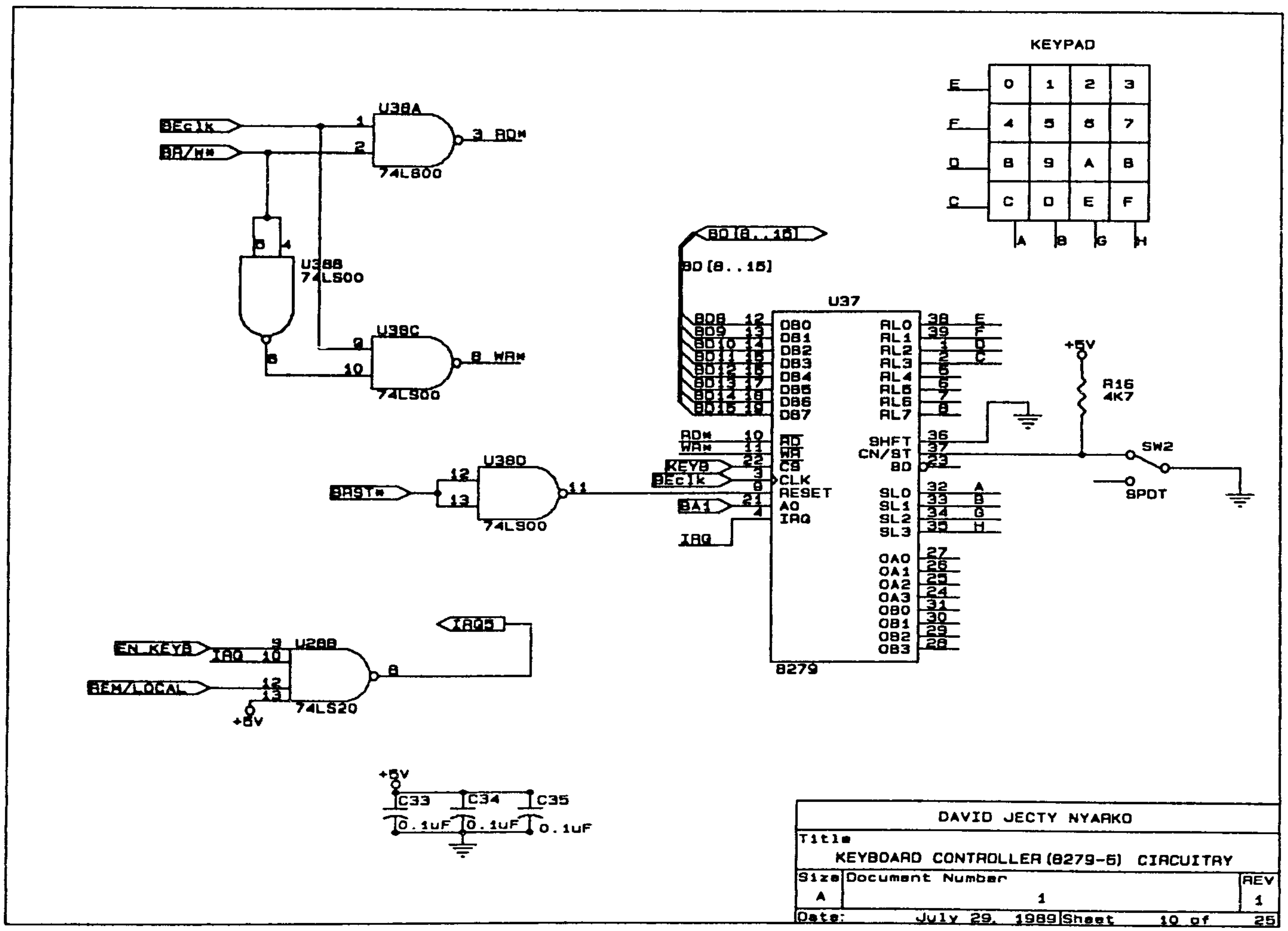

FIG. A.10 DETAILED SCHEMATIC OF KEYBOARD CONTROLLER CIRCUITRY

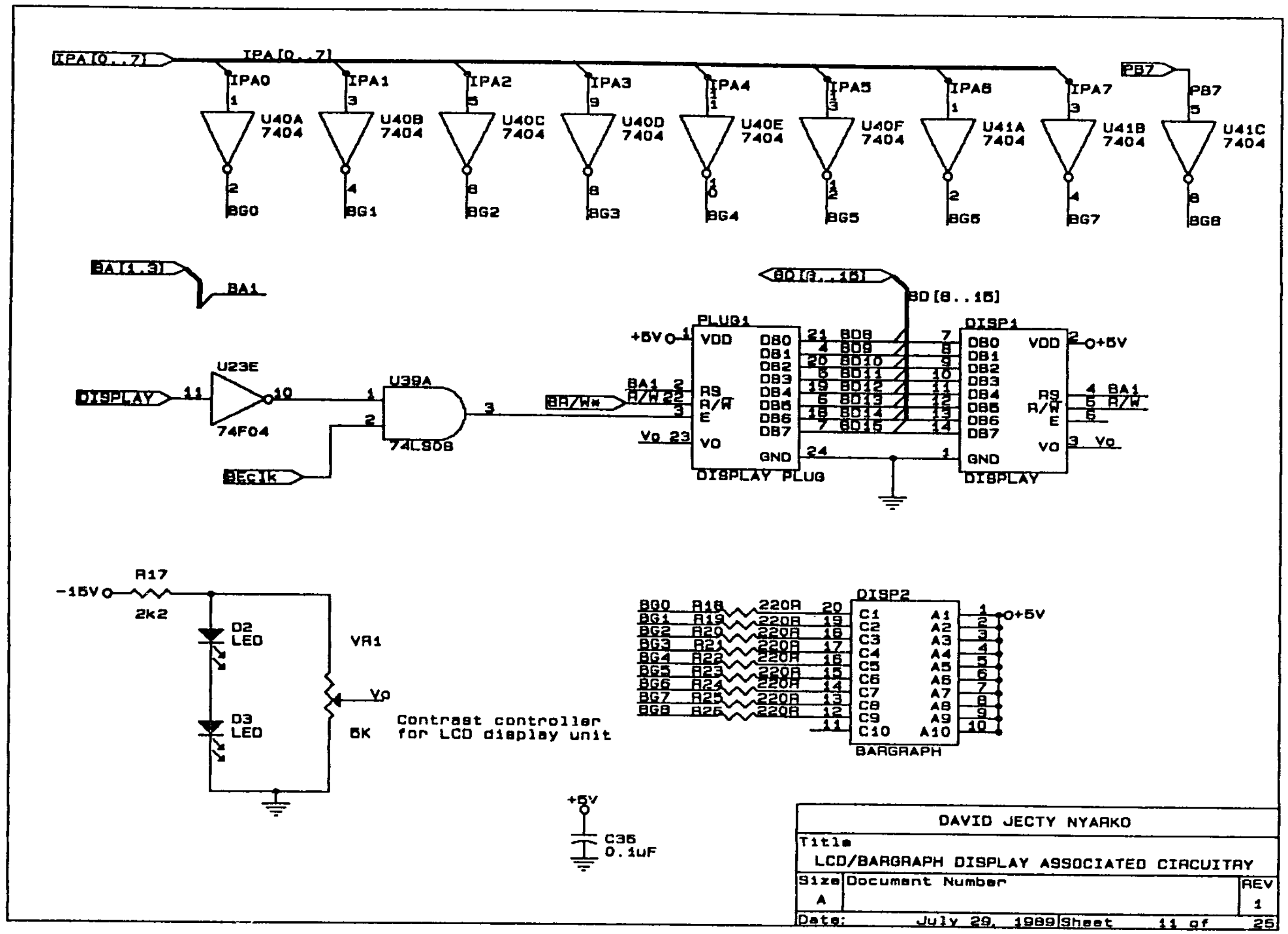

FIG. A.11 DETAILED SCHEMATIC OF LCD/BARGRAPH DISPLAY AND ASSOCIATED CIRCUITRY

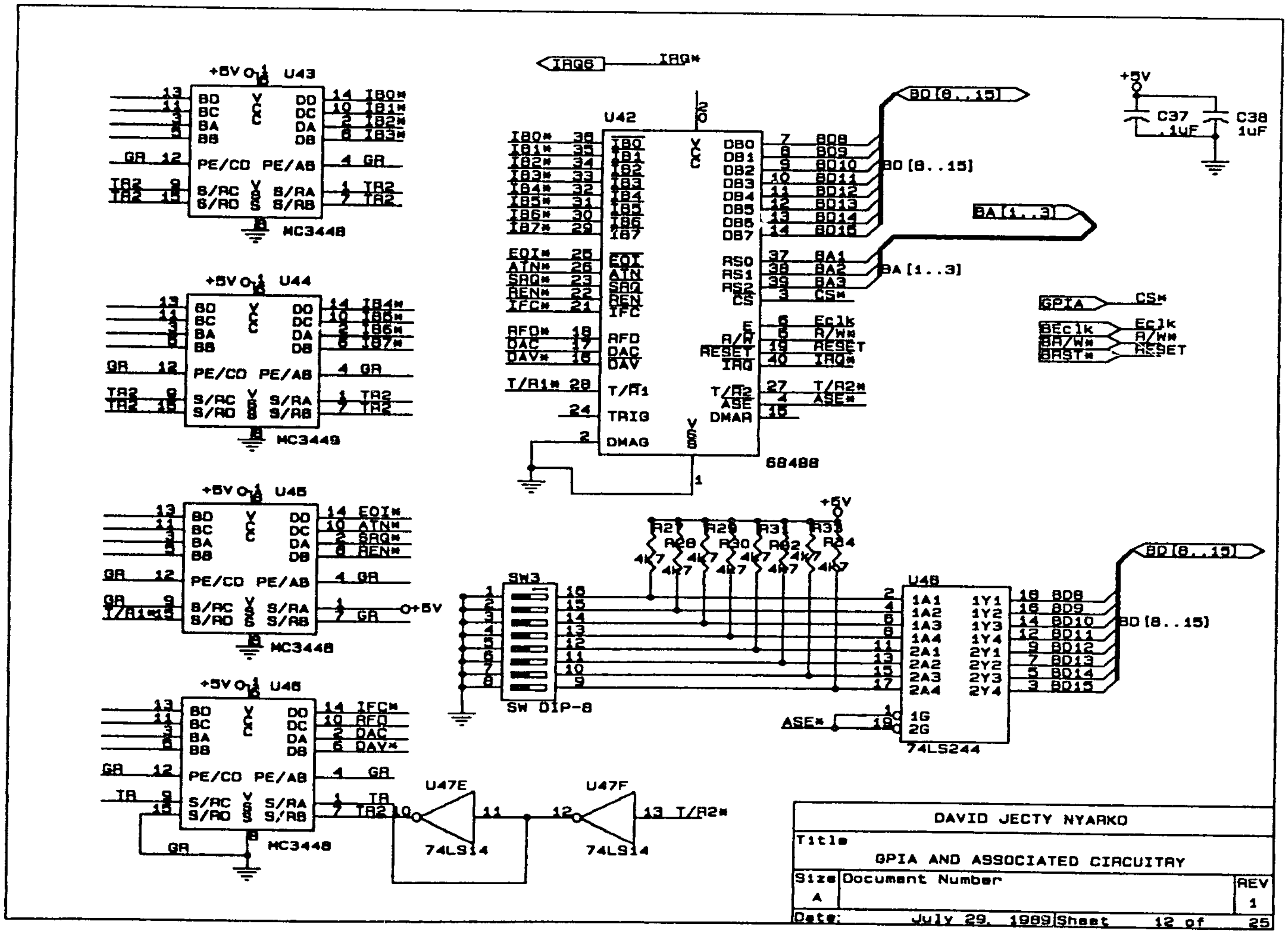

FIG. A.12 DETAILED SCHEMATIC OF GPIA AND ASSOCIATED CIRCUITRY

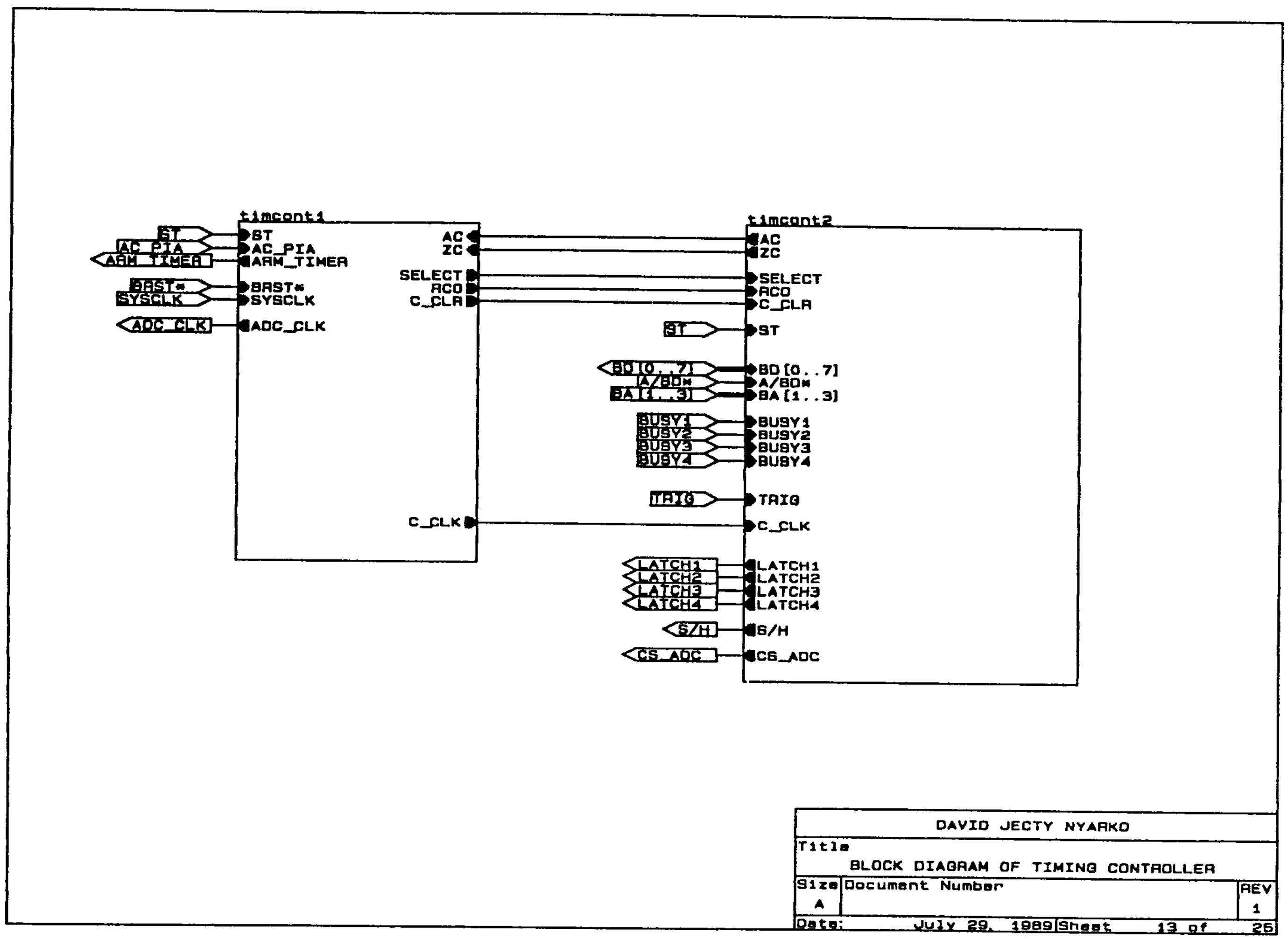

FIG. A.13 BLOCK DIAGRAM OF TIMING CONTROLLER

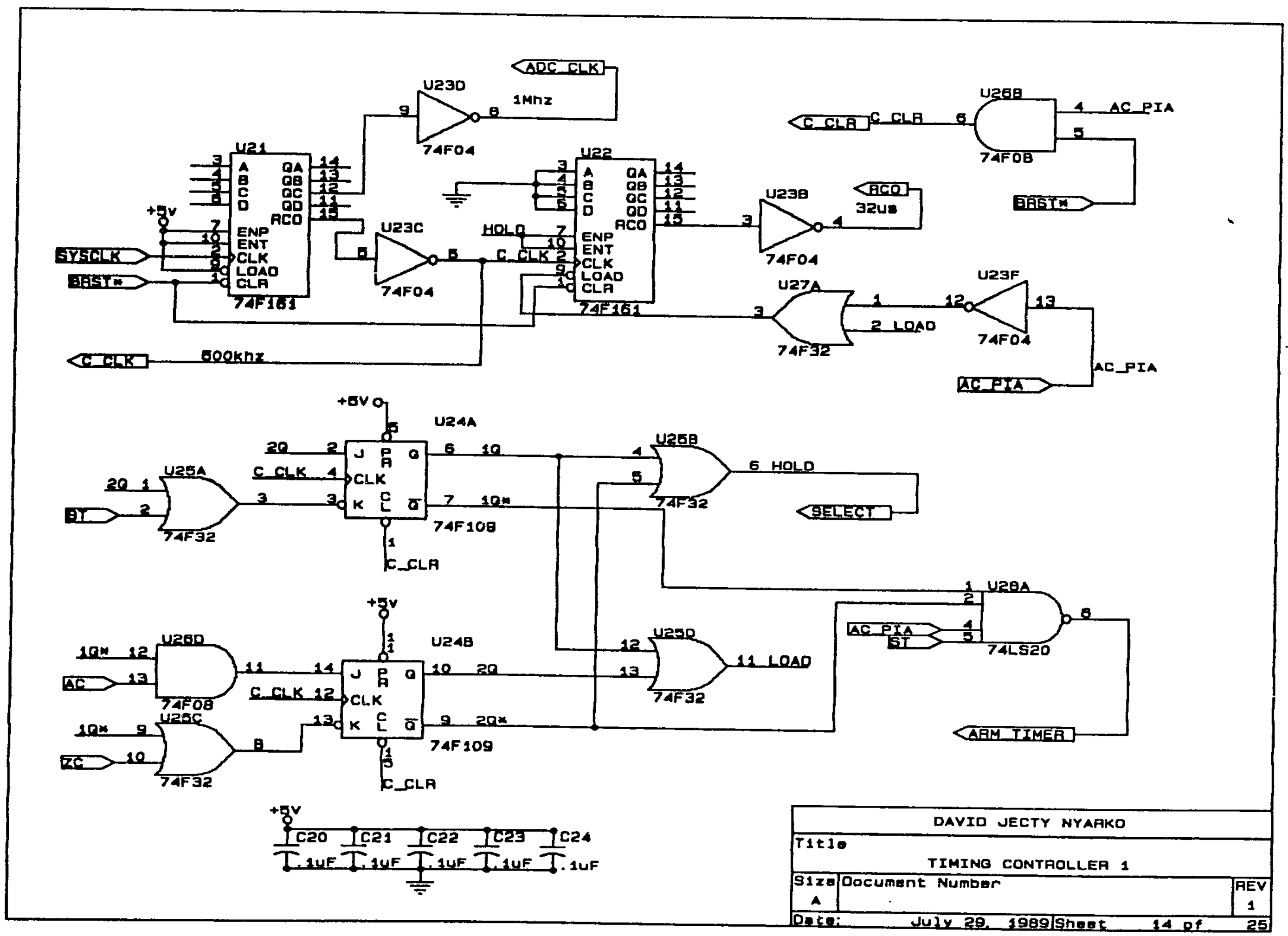

FIG. A.14 DETAILED SCHEMATIC OF TIMING CONTROLLER 1 CIRCUITRY

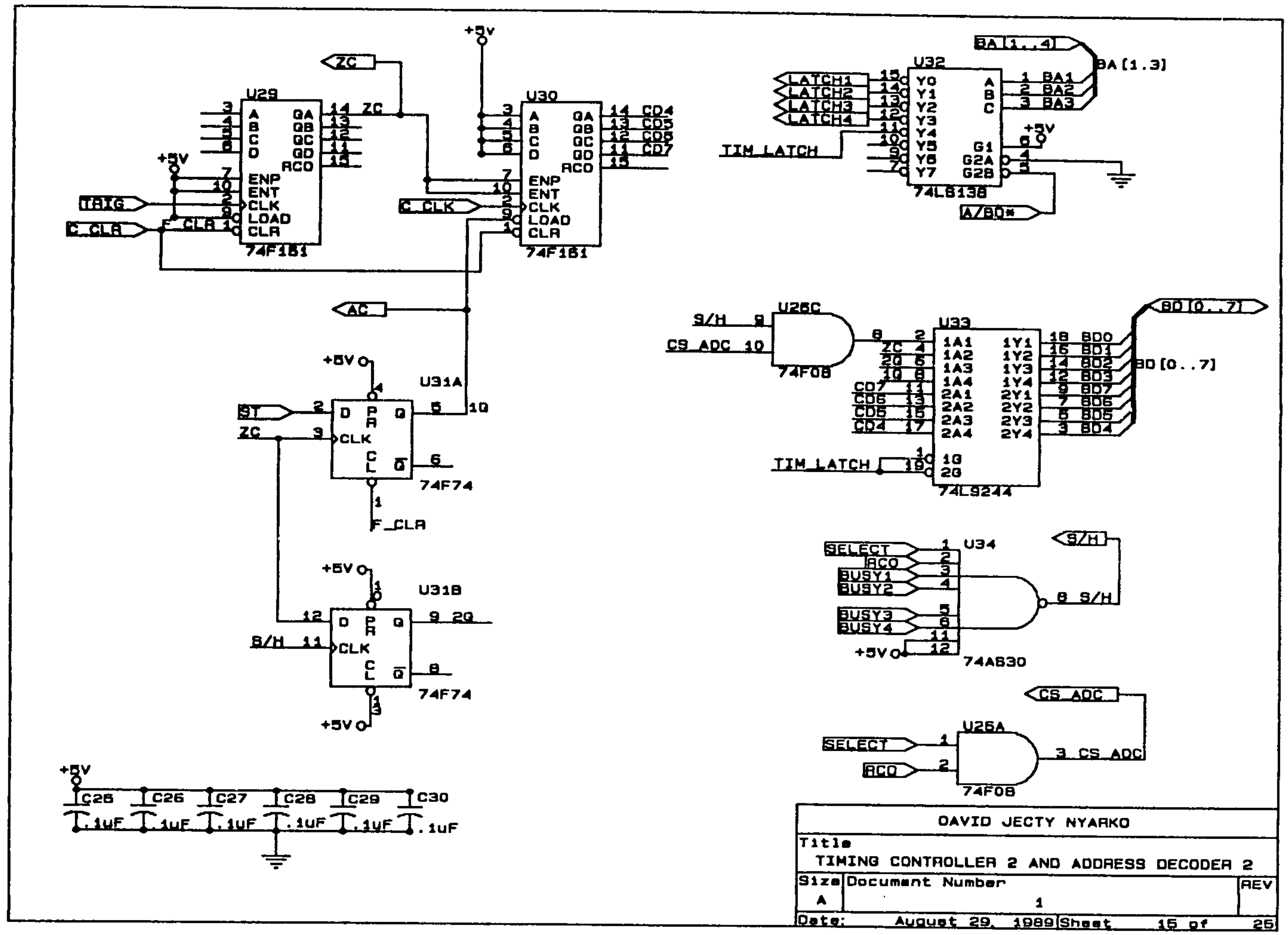

FIG. A.15 DETAILED SCHEMATIC OF TIMING CONTROLLER 2 AND ADDRESS DECODER 2 CIRCUITRY

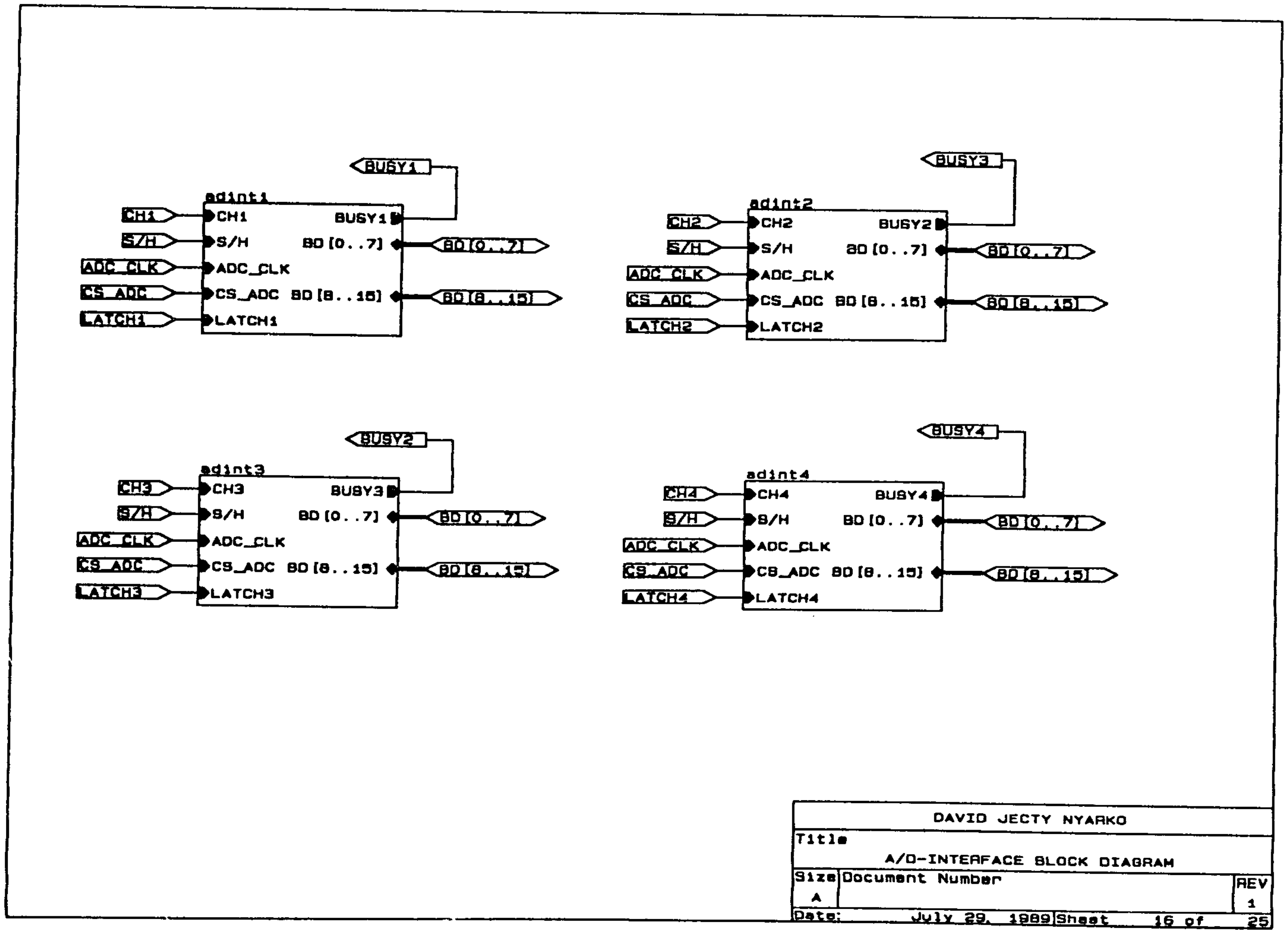

FIG. A.16 BLOCK DIAGRAM OF THE A/D INTERFACE CIRCUITRY

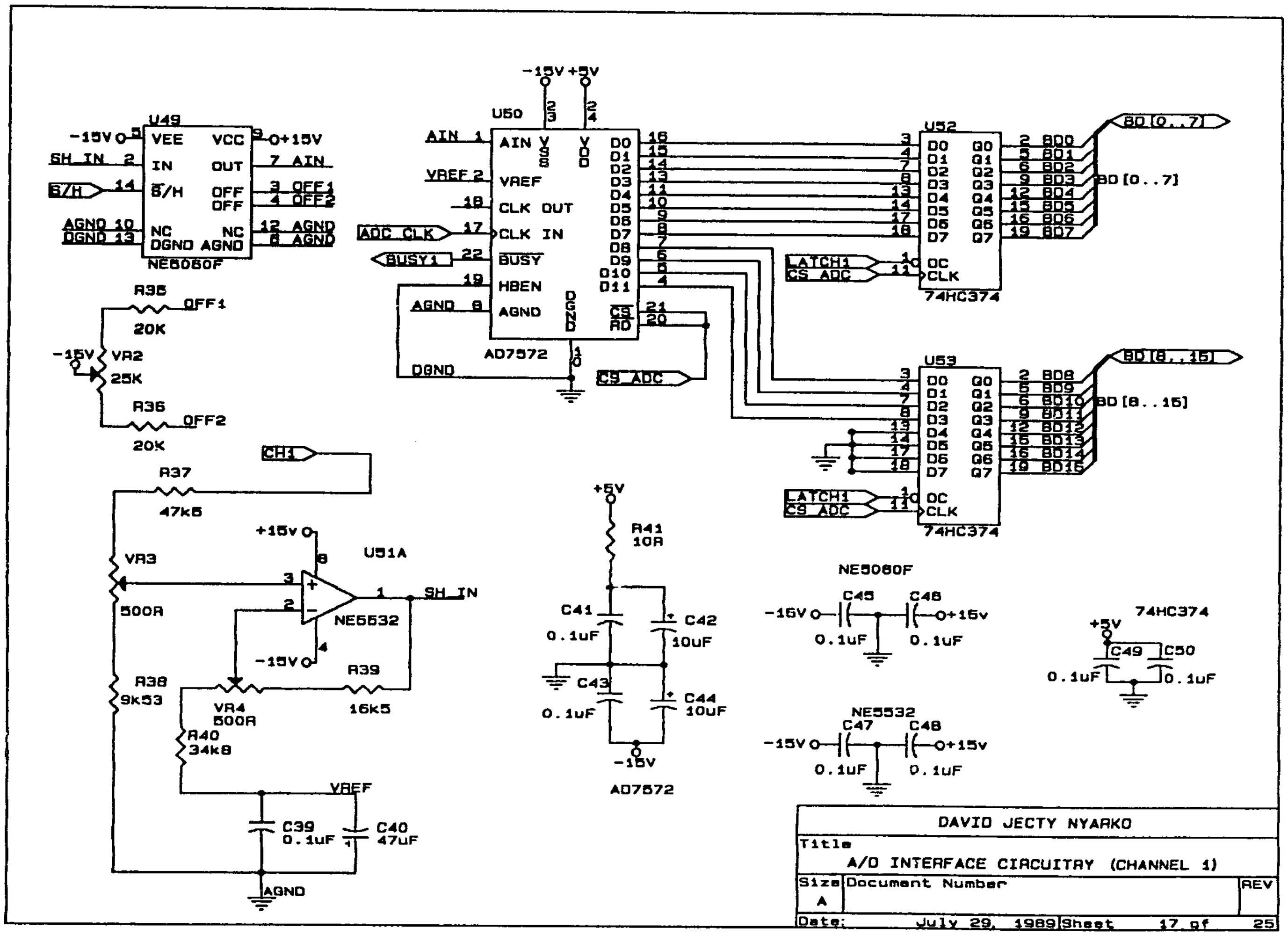

FIG. A.17 DETAILED SCHEMATIC OF THE A/D INTERFACE CIRCUITRY (channel 1)

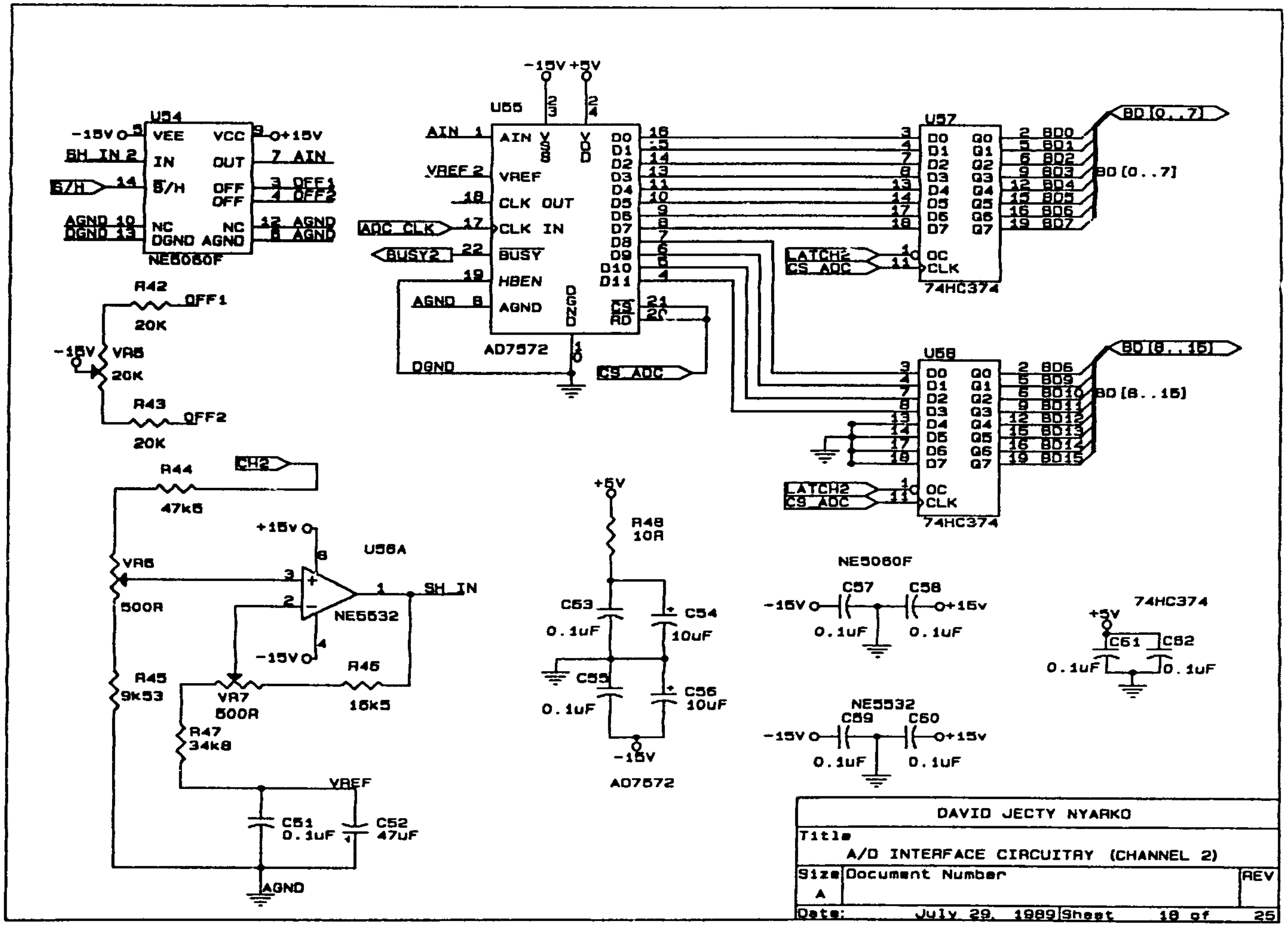

FIG. A.18 DETAILED SCHEMAATIC OF THE A/D INTERFACE CIRCUITRY (channel 2)

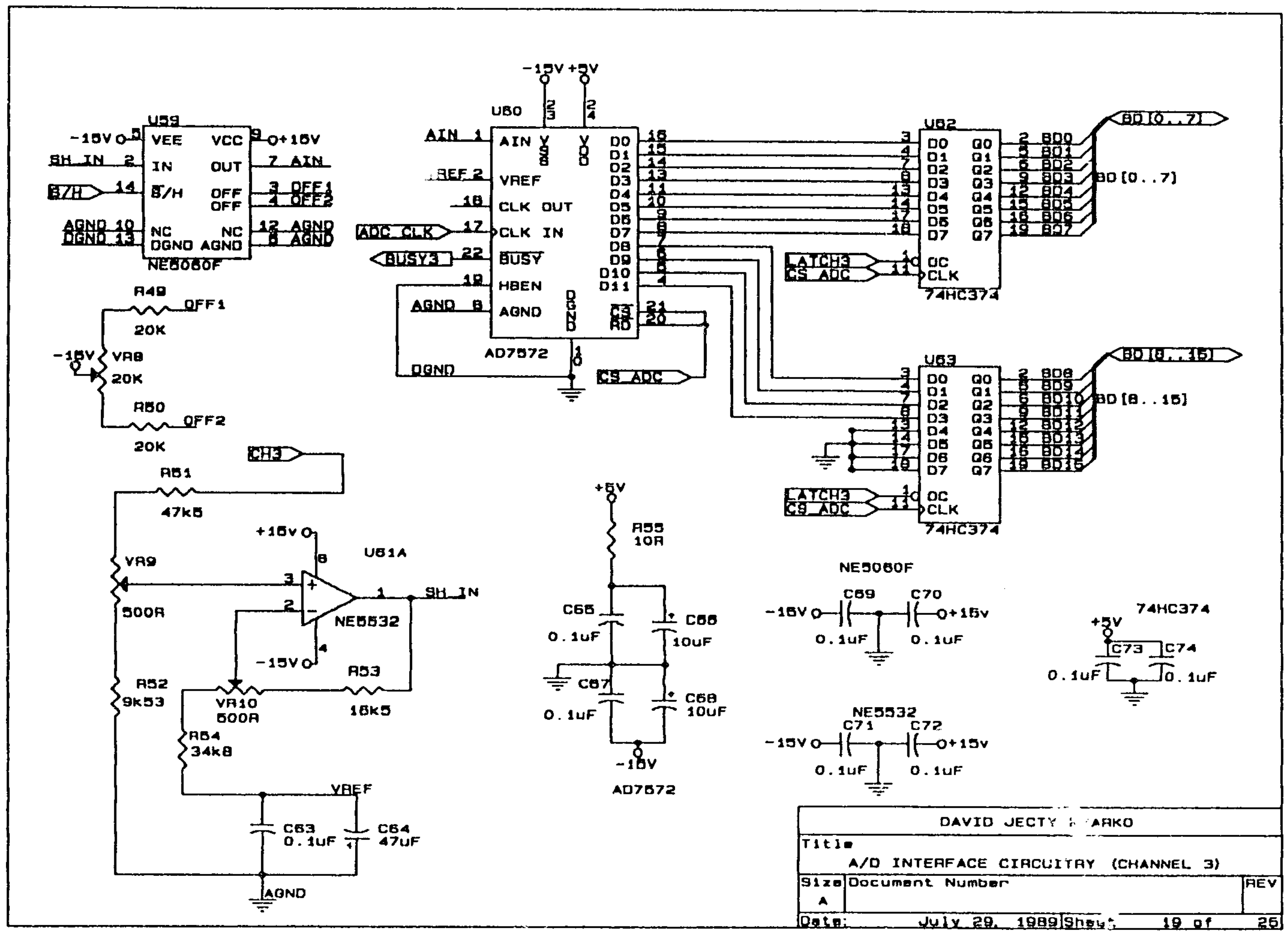

FIG. A.19 DETAILED SCHEMATIC OF THE A/D INTERFACE CIRCUITRY (channel 3)

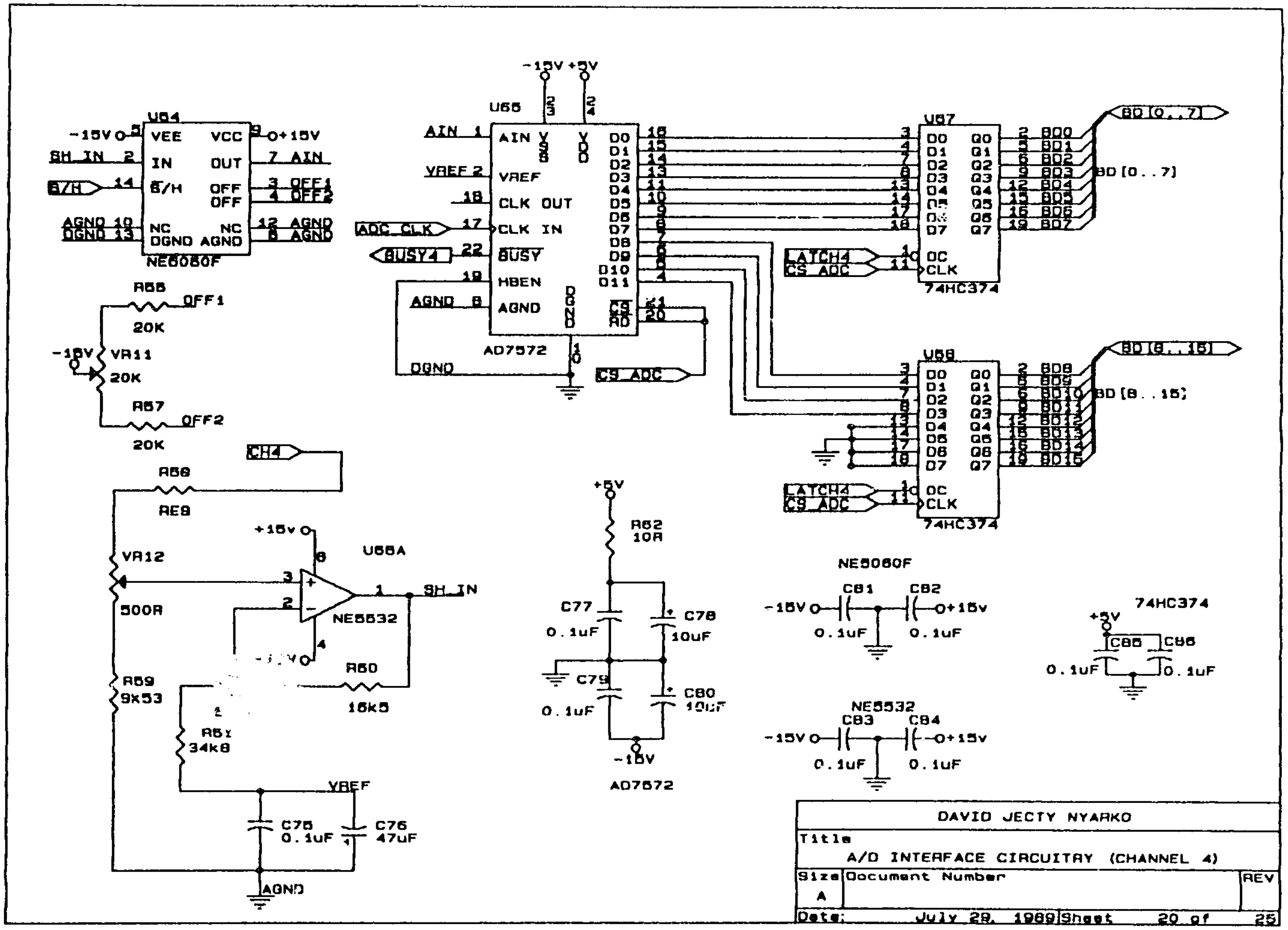

FIG. A.20 DETAILED SCHEMATIC OF THE A/D INTERFACE CIRCUITRY (channel 4)

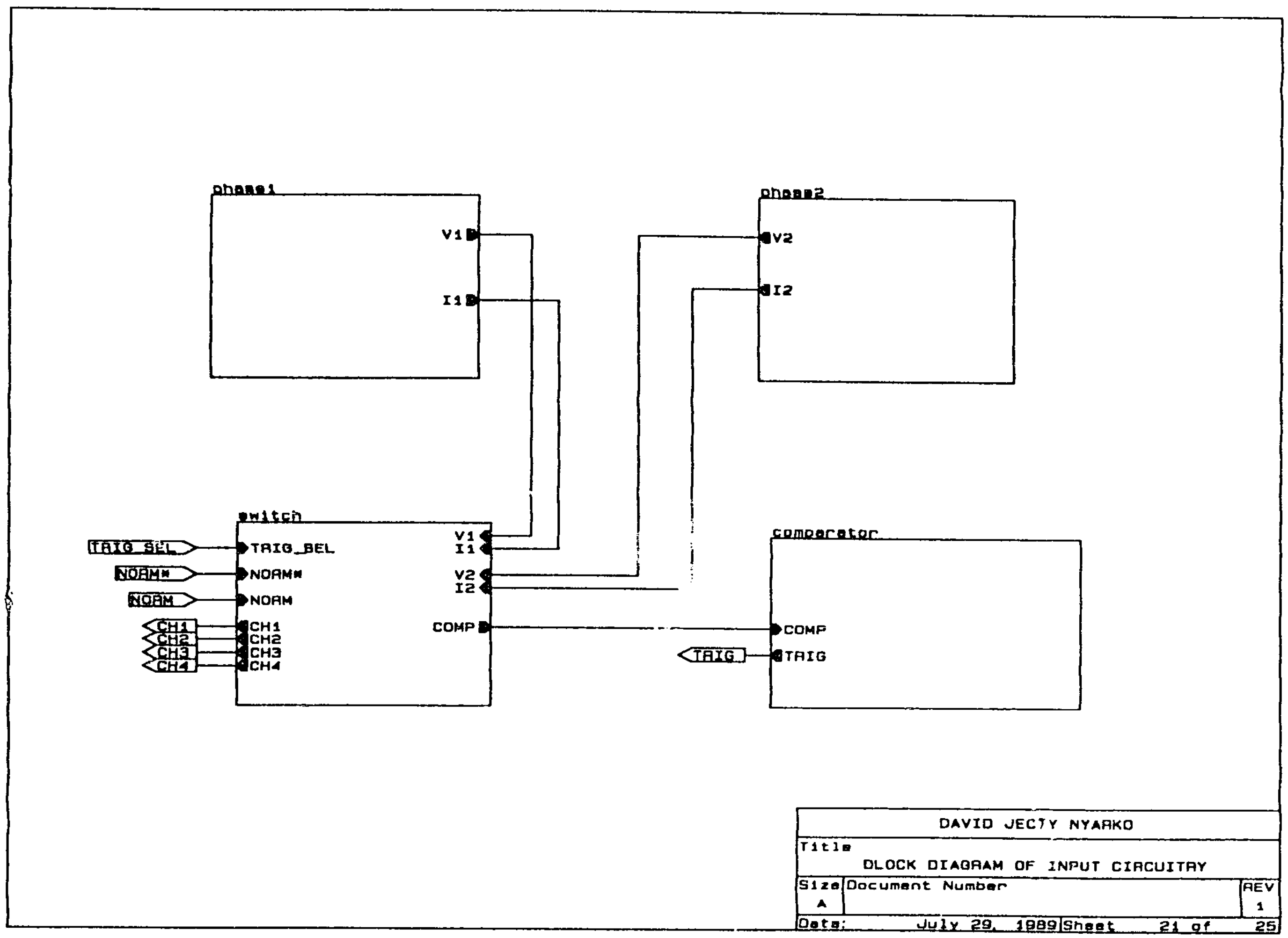

FIG. A.21 BLOCK DIAGRAM OF THE INPUT-CONDITIONING CIRCUITRY

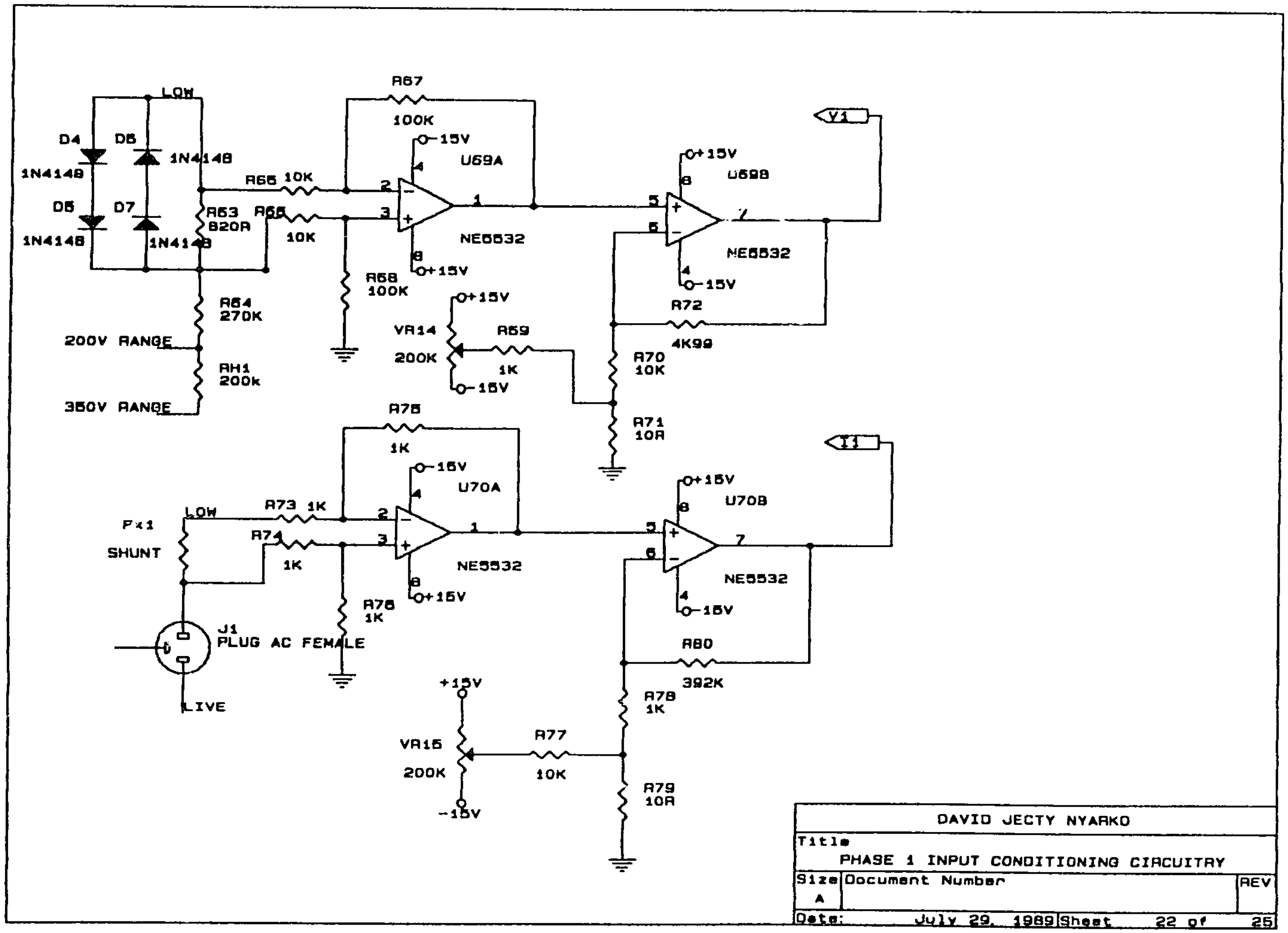

FIG. A.22 DETAILED SCHEMATIC OF THE PHASE 1 INPUT CONDITIONING CIRCUITRY

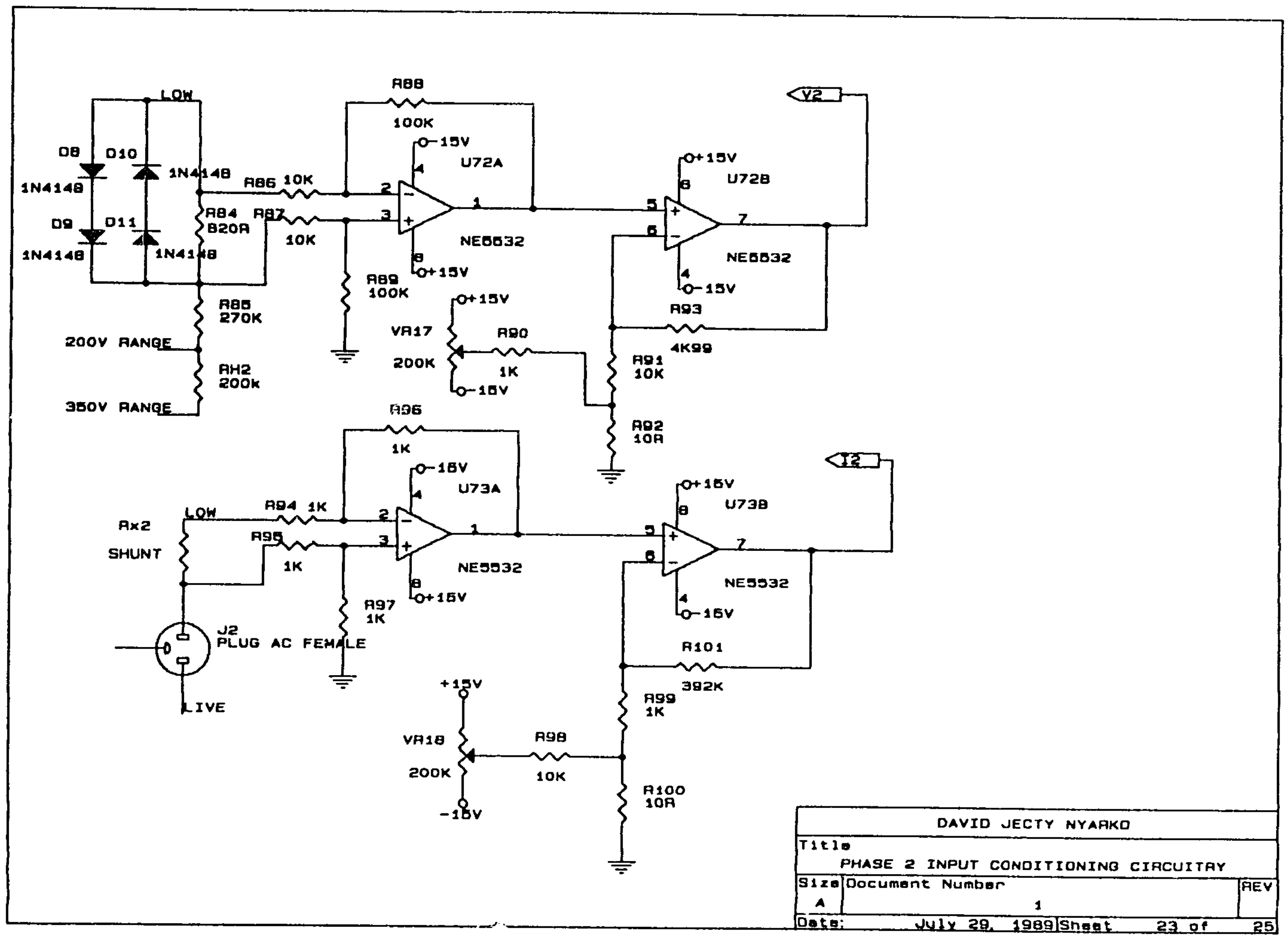

FIG. A.23 DETAILED SCHEMATIC OF THE PHASE 2 INPUT CONDITIONING CIRCUITRY

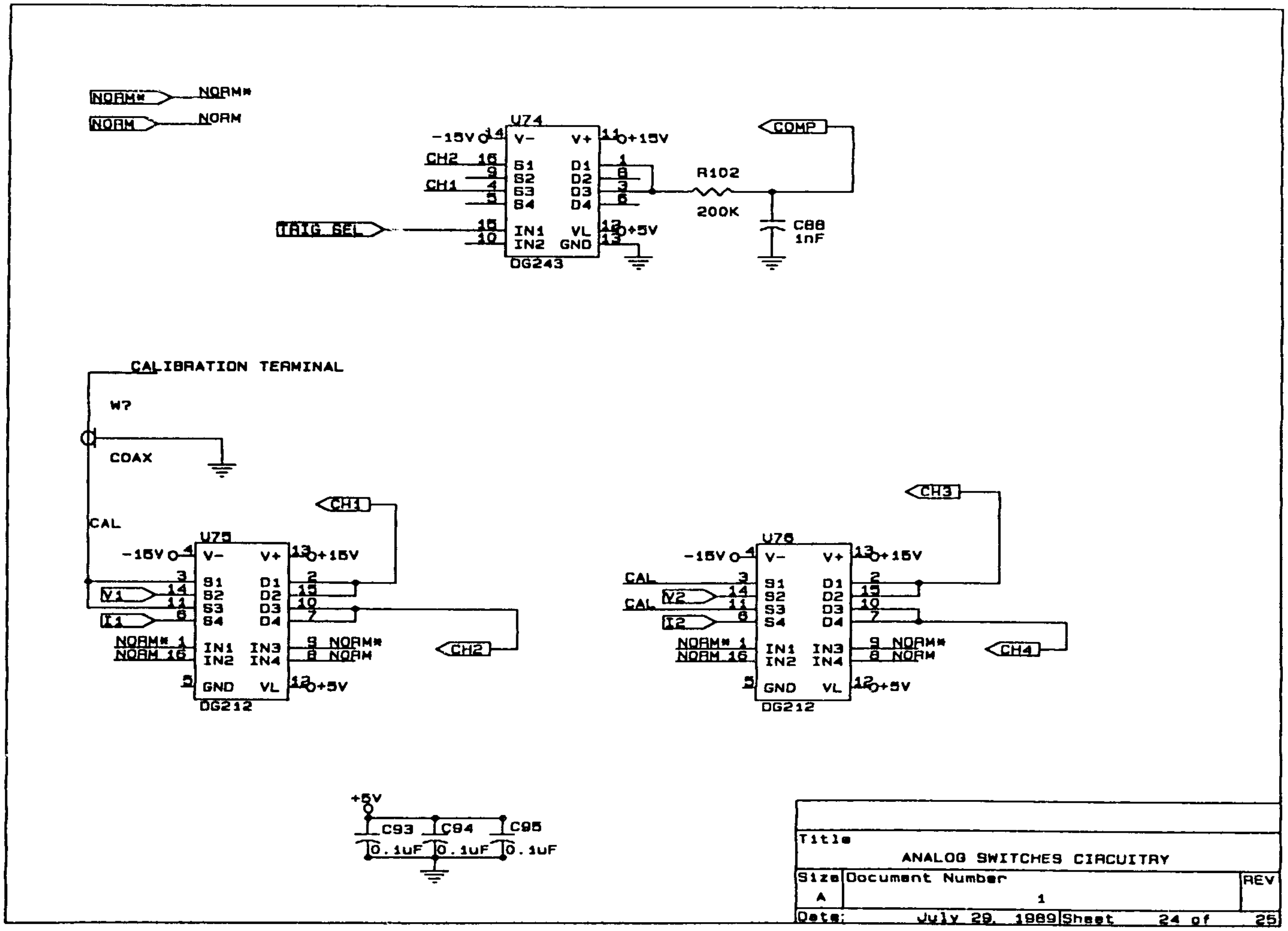

FIG. A.24 DETAILED SCHEMATIC OF THE ANALOG SWITCHES CIRCUITRY

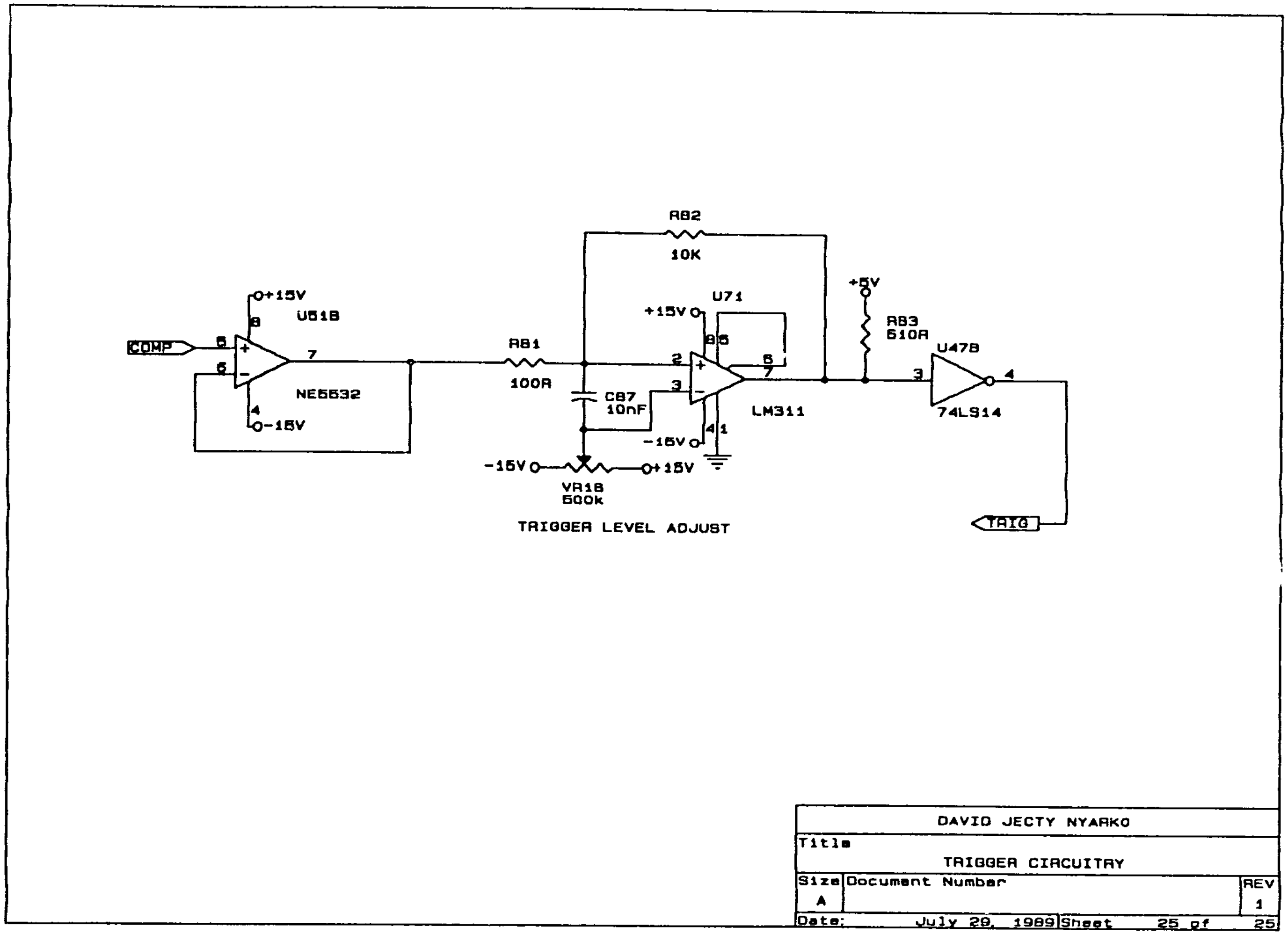

FIG. A.25 DETAILED SCHEMATIC OF THE TRIGGER CIRCUITRY

APPENDIX B SOFTWARE LISTING

```
;
;
        section  0
;
ramstart equu      $200000
ramend   equ       $20ffff
;
; lcd display registers
;
disp     equ       $6c0000        ;display
;
;  gpib registers
;
gpia     equ     $640000       ;gpia base register
instatr  equ     0             ;interrupt status  (r0r)
inmskr   equ     instatr       ;interrupt mask  (r0w)
cstatr   equ     2             ;command status (r1r)
adstatr  equ     4             ;address status (r2r)
admodr   equ     adstatr       ;address mode  (r2w)
auxcr    equ     6             ;auxiliary command (r3r)
adsw     equ     8             ;address switch (r4r)
adrg     equ     adsw          ;address register (r4w)
spolr    equ     10            ;serial poll (r5)
cpastr   equ     12            ;command pass through (r6)
ppolr    equ     cpastr        ;parallel poll (r6w)
dtinr    equ     14            ;data in (r7r)
dtoutr   equ     dtinr         ;data out (r7w)
;
; pia registers
;
piada    equ       $680000        ;pia base register
piadb    equ       piada+2        ;pia data reg. b
piacra   equ       piada+4        ;pia control reg. a
piacrb   equ       piada+6        ;pia control reg. b
;
; programmable keyboard/display
; interface register
;
keyb     equ       $700000
;
; ptm registers
;
time     equ     $600000
ctl13    equ     0         ;control register 1,3
ctl2     equ     2         ;control register 2
status   equ     2         ;status register
lth1     equ     4         ;timer 1 latch   (msb)
cnt1h    equ     4         ;timer 1 counter (msb)
lth2     equ     8         ;timer 2 latch   (msb)
cnt2h    equ     8         ;timer 2 counter (msb)
```

```
lth3      equ     12        ;timer 3 latch   (msb)
cnt3h     equ     12        ;timer 3 counter (msb)
;
; other equates
;
intsoff equ     $0f00
intson  equ     $f8ff
;
false_key equ 5
local    equ     %00010000
remote   equ     %11101111
key_dis  equ     %11110111
key_en   equ     %00001000
stac_dis   equ   %11111001
stac_en  equ     %00000110
ac_en     equ     %00000100
calib_on   equ   %01000000
calib_off  equ   %10111111
norm_on  equ     %00100000
norm_off   equ   %11011111
;
samp_st     equ  $1ffffe
;
line1    equ     0
line2    equ     $40
line3    equ     $14
line4    equ     $54
;
pt       equ     $a5
;
blink_on   equ   $f
blink_off  equ   $e
;
adc1 equ         $400000
adc2 equ         $400002
adc3 equ         $400004
adc4 equ         $400006
;
; data backgrounds
;
dbgnd_1  equ   0
dbgnd_2  equ   $5555
dbgnd_3  equ   $3333
dbgnd_4  equ   $0f0f
dbgnd_5  equ   $00ff
;
;
tim_latch equ    $400009
;
stack_top equ    ramend-1     ;stack pointer
stack_base equ      stack_top-320
;
```

```
;   scale factors
;
ch1low_num    equ  2020
ch1high_num   equ  3638
ch2_num       equ  201
ch3low_num    equ  2020
ch3high_num   equ  3638
ch4_num       equ  201
;
ch1_denom     equ  11
ch2_denom     equ  10
ch3_denom     equ  11
ch4_denom     equ  10
w1_denom equ  17
w2_denom equ  17
;
;*************
; main program
;*************
;
        dc.l    stack_top    ;reset stack pointer
        dc.l    start       ;reset program counter
        dc.l    b_excpt     ;bus error
        dc.l    a_excpt     ;address error
        dc.l    i_excpt     ;illegal instruction
        dc.l    d_excpt     ;divide by zero
        dc.l    c_excpt     ;chk
        dc.l    o_excpt     ;trapv
        dc.l    p_excpt     ;privilege
i_trace dc.l    t_excpt     ;trace
        dc.l    x_excpt     ;11010
        dc.l    y_excpt     ;11111
        ds.l    12          ;unassigned as yet
        dc.l    s_excpt     ;spurious iterrupt
;
;       autovectored interrupts
;
;
        dc.l    int         ;level 1
        dc.l    int         ;level 2
        dc.l    int         ;level 3
        dc.l    int_tim     ;level 4 autovector (timer)
        dc.l    int_key     ;level 5 autovector  (8279)
        dc.l    int_gpia    ;level 6 autovector (gpia)
        dc.l    int7        ;level 7
;
        ds.l    224
;
;*******************
; program start
;*******************
;
```

```
;
start:
        reset                          ;reset all devices
        ori.w    #intsoff,sr           ;disable interrupts
;
pia:
    lea      piada,a0
    moveq    #0,d0
    movep.w d0,4(a0)
    not.w    d0
    movep.w d0,0(a0)              ;ports a and b as outputs
    move.w   #$404,d0             ;point to dra and drb
    movep.w d0,4(a0)
    not.w      d0
    movep.w d0,0(a0)
key:
        lea        keyb,a0
        move.b     #%00000011,2(a0)
        move.b     #%00101000,2(a0)
        move.b     #%01000000,2(a0)
        move.b     #%11000001,2(a0)
timer:
    lea        time,a0
    move.w     #$ffff,d0
    movep.w    d0,lth1(a0)   ;initialize timer counter 1
    movep.w    d0,lth2(a0)   ;initialize timer counter 2
    movep.w    d0,lth3(a0)   ;initialize timer counter 3
;
    move.b  #%00001010,ctl13(a0) ;timer 3 external inputs
;                                 freq mode ints. disabled
lcd_display:
;
        lea      disp,a0
;
        moveq.l #-2,d0
        moveq.l #-2,d1
        moveq.l #-2,d2
        moveq.l #-2,d3
;
        move.b   #$30,(a0)      ;internal reset
lcd1    dbra     d0,lcd1
        move.b   #$30,(a0)
lcd2    dbra     d1,lcd2
        move.b   #$30,(a0)
lcd3    dbra     d2,lcd3
;
        move.b   #$38,(a0)     ;2 line mode 8
;                               bit data transfer
busy_1  dbra     d3,busy_1
;
        move.b   #$d,(a0)
busy_2  move.b   (a0),d0
```

```
        bmi.s   busy_2
;
        move.b  #6,(a0)       ;set entry mode
busy_3  move.b  (a0),d0
        bmi.s   busy_3
;
gpia_:
        lea     gpia,a1
;
        move.b  adsw(a1),d0   ;read address switch
        move.b  d0,adrg(a1)   ;write to address reg.
        move.b  #0,auxcr(a1)    ;clear gpib reset state
        move.b  #0,inmskr(a1)   ;clear interrupts
;
    bsr     clear
;
    move.b  #blink_on,(a0)    ;set the cursor blinking
    move.b  #blink_on,d6     ;save the state of the display
blon1   move.b  (a0),d0
    bmi.s   blon1
;
    moveq   #line2,d3
    lea     messg1(pc),a5
    bsr     display
;
; memory(13N) test for static rams
;
; a2 holds failing address
;
    moveq.l   #-14,d5
;
    lea.l     ramstart,a0
    lea.l     ramend-1,a1
;
    move.l    #dbgnd_1,d0
    move.b    #1,piada
memtest:
    move.w    d0,d1
    move.w    d0,d2
    not.w     d2
    clr.w     d3
    clr.w     d4
;
    move.l    a0,a2
;
mem_init:           ;Initialization Wr(0)
    move.w    d1,(a2)+
    cmpa.l    a2,a1
    bcc.s     mem_init
;
m1234:
    tst.w     d4
```

```
        beq.s     m12_init
        movea.l   a1,a2
        bra.s     m1234a
;
m12_init:
        movea.l   a0,a2
;
m1234a:
        cmp.w     (a2),d1
        bne.s     ramerr
        move.w    d2,(a2)
        cmp.w     (a2),d2
        bne.s     ramerr
        tst.w     d4
        beq.s     m1234b
        subq.l    #2,a2
        cmpa.l    a0,a2
        bra.s     m1234c
m1234b:
        addq.l    #2,a2
        cmpa.l    a2,a1
m1234c:
        bcc.s     m1234a
        exg.l     d1,d2
        not.w     d3
        bne.s     m1234
;
        not.w     d4
        bne.s     m1234
;
        cmpi.w    #42,d5
        beq  ram_ok     ;test successful
        addi.l    #14,d5
        jmp  dbg_table(pc,d5.w)
;
dbg_table:
        move.w    #dbgnd_2,d0
        move.b    #2,piada
        bra.s     memtest
        move.w    #dbgnd_3,d0
        move.b    #4,piada
        bra.s     memtest
        move.w    #dbgnd_4,d0
        move.b    #8,piada
        bra.s     memtest
        move.w    #dbgnd_5,d0
        move.b    #16,piada
        bra  memtest
;
ramerr  moveq     #line1,d3
        lea       messg2(pc),a5
        bsr       display
```

```
;
        lea.l   disp_buff1,a5
    move.l    #'A2= ',(a5)+
        move.l  a2,(a5)
        bsr     hex_ascii
    subq.l    #4,a5
        moveq   #line2,d3
        bsr     display
        lea.l   disp_buff2,a5
    move.l    #'D1= ',(a5)+
        move.l  d1,(a5)
    moveq.l   #2,d5
        bsr     hex_ascii
    subq.l    #4,a5
        moveq   #line3,d3
        bsr     display
;
        lea.l   disp_buff3,a5
    move.l    #'D2= ',(a5)+
        move.l  d2,(a5)
        bsr     hex_ascii
    subq.l    #4,a5
        moveq   #line4,d3          ;display failing
        bsr     display            ;address
;
here2   bra.s   here2
ram_ok:
    bclr.l    #0,d6
    move.b    d6,disp_state
    bsr  disp_ctrl
        move.b  #$ff,key_buff       ;initialize key buffer
        move.b  #0,pia_a
        move.b  pia_a,piada     ;turn off all leds
        move.b  #0,pia_b
        ori.b      #local,pia_b  ;keyboard on local control
;                  triger source = channel 1
        ori.b   #key_en,pia_b
        ori.b   #norm_on,pia_b
        andi.b  #calib_off,pia_b
    bclr.b    #7,pia_b  ;turn off remote led (no 8)
        move.b  pia_b,piadb     ;enable keyboard interrupts
        clr.b   prog_state        ;assume normal mode
    clr.b     state1
    move.w    #100,shunt_val1
    move.w    #100,shunt_val2
;
    move.w    #ch1low_num,tnumer_v1l
    move.w    #ch1high_num,tnumer_v1h
    move.w    #ch2_num,tnumer_i1
    move.w    #ch3low_num,tnumer_v2l
    move.w    #ch3high_num,tnumer_v2h
    move.w    #ch4_num,tnumer_i2
```

immediately after the results of the previously converted analog input signals of all 4 A/D converters have been stored in the system RAM. An AC acquisition cycle is the time interval during which the crossing signal (ZC) is at a logic high state as shown in Fig. 5.1 (page 117). Fig. 5.2 (page 117) shows the relationship between the crossing signal and the software tests for the end of an AC acquisition cycle. The above-mentioned software test also provides information regarding the state of the AC data acquisition process as obtained from the logic state of lines BD0 - BD4 of IC U33 in Fig. A.15. Based on the information obtained, a look-up table is used to determine the next stage of program execution as indicated below.

	Logic state of the relevant lines of IC U33				Action taken by routine
	BD3 (AC)	BD2 (2Q)	BD1 (ZC)	BD0	
a)	0	0	0	*	Start sampling again
b)	0	0	1	*	Start sampling again
c)	0	1	0	*	Start sampling again
d)	0	1	1	*	Start sampling again
e)	1	0	0	*	Stage 1 (An additional sample required)
f)	1	0	1	*	Continue (1st conversion underway)

g) 1 1 0 * Stage 2 (2 more samples required)

h) 1 1 1 * Continue (ZC high)

* - A don't care condition

Restarting the sampling procedure involves reinitializing the state controller flip-flops and counters as well as the PTM.

5.2 Computational routines

These routines are used in the determination of the various system parameters. The routines include floating point and fixed point routines. A C programming language floating point routines for a 68000 microprocessor target machine are used. In addition a conversion routine for a 64-bit fixed point to floating point notation and vice-versa is employed. Most of the computations involve fixed point routines resulting in a better computational time as opposed to the use of floating point routines. The major mathematical computations required in determining the parameters are those of multiplication, summation and square-root extraction. To speed up the computations extensive use is made of the microprocessor registers. In the scale factor routines, since the denominators consist of a mantissa of 2 and an exponent, determining the appropriate parameter values in their SI units, simply involves multiplications and shift operations.

The sign of the phase angle for the selected phase is determined in software. This routine simply determines the sample number of the voltage and current samples which result in the respective signal maximum value. A higher sample number for the current signal indicates a lagging (-) power factor. The converse indicates a leading power factor.

5.3 Utility routines

These routines include the input and display subroutines, system state indication routines and a number of house-keeping routines.

The input routine comprises the keypad input routine and the GPIB input routine. The self-explanatory flow chart for the former routine is shown in Fig. 5.3 (page 118). This routine interacts with the keypad on an interrupt basis. The GPIB routine employs the GPIA device operating in an interrupt mode. This routine which employs interrupt service routine int_gpia is described below.

```
int_gpia:
    save all registers
    do {
    read character;
    switch;
    case 1 ( invalid character);
         break;
    Case 2( character = U )
```

```
            if character previously entered
            { decrement program counter;
            decrement program pointer; }
            break;
        Case 3 ;
            convert character to keypad code;
            store code in program buffer;
            break;
        }while ( character != 'T' or program counter < 128 )
        enddo
        restore all registers
    end int_gpia
```

The main display routines are the LCD and the bargraph display routines. There are two main LCD display routines; an ASCII character display routine and an ASCII string display routine. The former routine simply uses the LCD display unit instruction set and internal ASCII font table. The flow diagram of the string display routine is shown in Fig. 5.4 (page 119). The latter routine, requires an input ASCII string pointed to by register A5. The string should terminate with a zero. In addition, the line number of the display where the result should appear is passed to the routine through register D3 of the microprocessor. This routine serves all the microprocessor registers.

The bargraph display routine the corresponding LED status correspond to the bits in 2 memory locations. LEDS

0 to 7 correspond to one memory location and LED 8 corresponds to a bit in the second memory location. Any required change in an LED status, is thus performed by initially changing the corresponding memory location bit, and finally writing the contents to the corresponding PIA port. The state of the bargraph display LEDS are summarized below.

LED number	Status when on
0	Display off
1	Overload Test off
2	Trigger Source = Channel 2
3	350v range selected on channel 1
4	350v range selected on channel 3
5	Storage of displayed data enabled
6	System in program mode
7	Storage space full
8	Keypad in remote mode

The data entry mode routine employs the input and LCD display routines. This routine is employed in the entry of numerals. This routine is centered around sub-routine acq. The latter subroutine which preserves all registers is described below.

acq:

Save working registers;

Clear display and digit counter;

Display module message;

```
    do {
    Read keycode;
    Switch;
    Case 1( Invalid keycode)
              break;
    Case 2( keycode corresponds to ^5 )
              if numeral previously entered
              { move display cursor back one space };
              break;
    Case 3:   -Store keycode in program buffer
              -Display numeral entered
              -Update digit counter
    }while ( keycode != ^6 and digit counter != 4 )
    enddo
  restore working registers;
  return from subroutine;
end acq:
```

The program entry mode routines comprise the keypad programming routines and the GPIB routines that have already been described. The first-mentioned routine uses the input and LCD display routines. This routine is centered around subroutine automode. The latter subroutine which preserves all registers, is described below.

```
automode:
    Save working registers;
    Initialize program mode variables;
```

```
    Clear display;
    do {
    Read keycode;
    Switch;
    Case 1( Invalid keycode)
              break;
    Case 2( keycode corresponds to ^D )
              if character previously entered
              { decrement program counter;
              decrement program pointer; }
              break;
    Case 3:   -Store keycode in program buffer
              -Display corresponding GPIB command
              -Update program size counter
    }while ( keycode != ^E or program size < 128 bytes )
    enddo
 restore working registers;
 force return to main routine;
end auto_mode:
```

The system state indication routines are simply routines geared towards the ease of maintenance of the system. These routines utilize the microprocessor exception vectors excluding those allocated to the system device interrupts. Following an exception, the address causing the exception as well as the type of exception is displayed. In this mode only a system reset can restart the system.

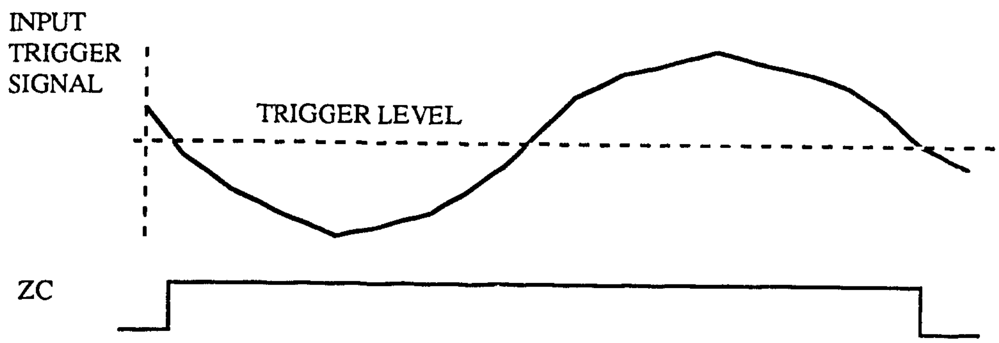

Fig. 5.1 Crossing detector timing diagram 1

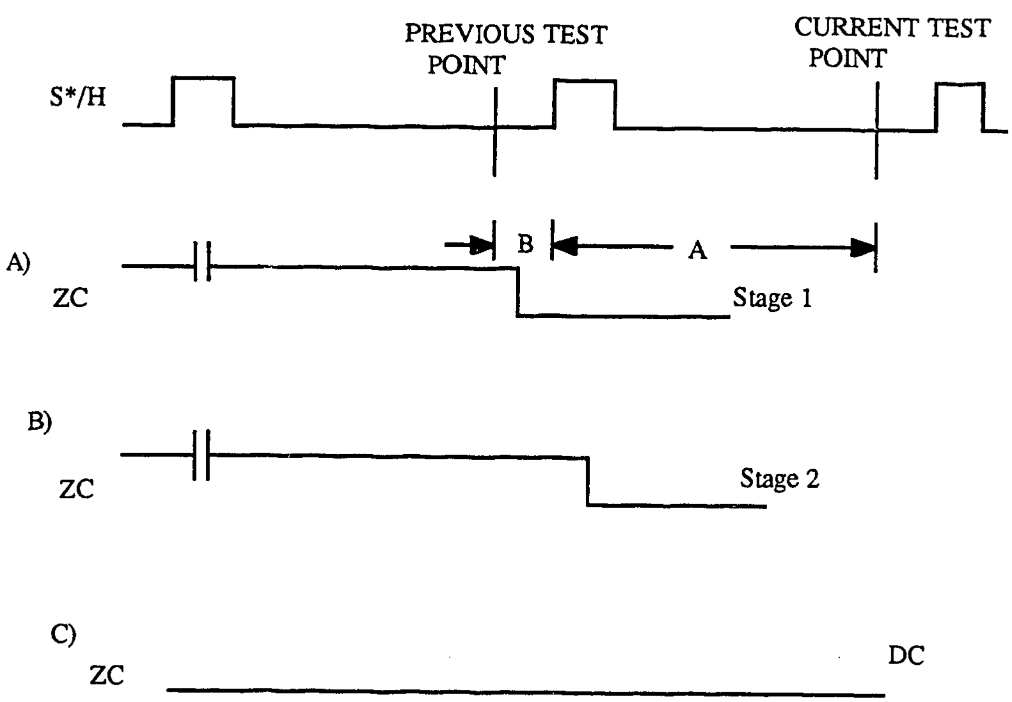

Fig. 5.2 Crossing detector timing diagram 2

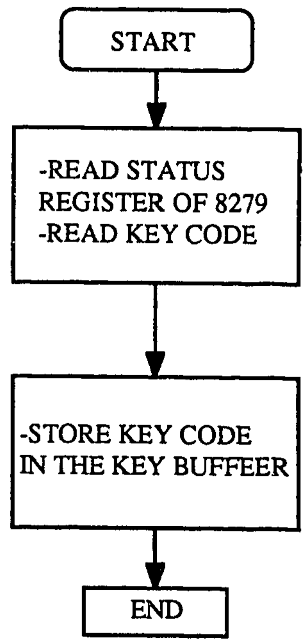

Fig. 5.3 Flowchart of the input routine

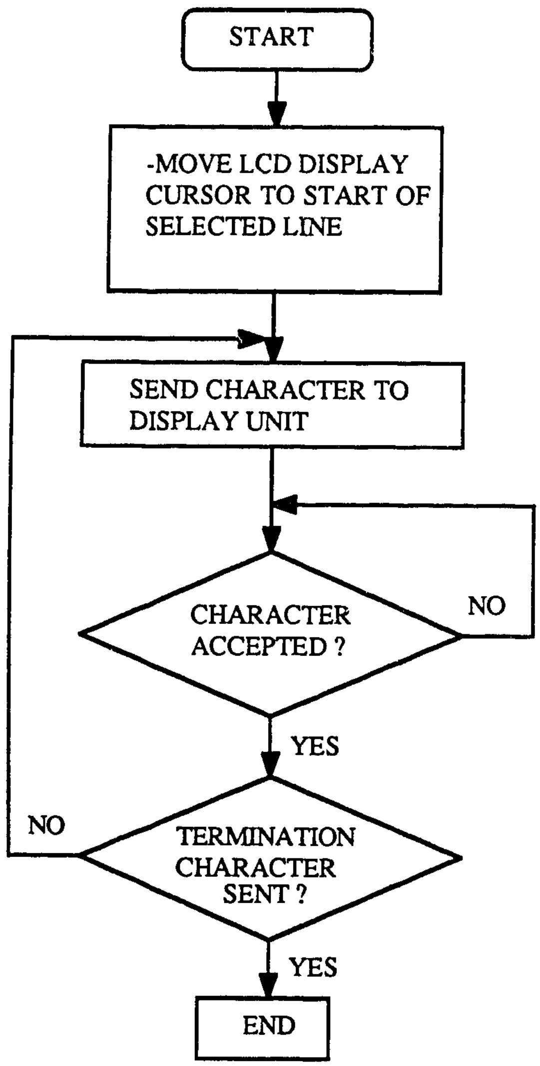

Fig. 5.4 Flowchart of the LCD display routine

6. SYSTEM OPERATION, PERFORMANCE AND SPECIFICATIONS

The system commands and their effects on the wattmeter operations have already been described. A convenient order of operation is however detailed. Following the power-up of the system, when the first menu is displayed, the shunt values can be entered. This can be followed by entering the appropriate voltage ranges. The command B is next entered and the variable resistors VR2, VR5, VR8 and VR11 are adjusted until the displayed parameters V1, I1, V2 and I2 are zero. Following this, the appropriate input terminals can be connected to the signal sources. The required measured value can be obtained by selecting the corresponding command.

6.1 Calibration

It is recommended that the system be up and running for at least a couple of hours prior to undertaking this procedure. This ensures that the system would have settled down to it's quiescent operating state. A standard voltmeter and ammeter, a variable voltage source 0 - 250V ac and a variable alternating current source are required for this procedure.

The command '4' is entered to disable the overload test. The command 'B' is entered, thus placing the system in the normal mode. The respective inputs of each channel are connected together. The corresponding offset adjustment resistors VR14, VR15, VR17 and VR18 are adjusted until a zero volt reading is recorded on the respective outputs CH1-CH4 of the analog switches.

Calibration of channels 1 and 3 are performed as follows. The system is set to the calibration mode by entering the command 'C'. Channel 1 is selected and the calibration terminal grounded. Variable resistors VR2 and VR4 indicated in Fig. A.17 are adjusted until a reading of 000800 is obtained. Next, the system is switched to channel 3. Variable resistors VR8 and VR10 indicated in Fig. A.20 are adjusted until a reading of 000800 is obtained. The command 'B' is next entered and various input voltages applied to the respective voltage inputs using the variable voltage source. The corresponding gain setting variable resistors VR3 and VR9 are adjusted until the meter reading corresponds to that of the standard voltmeter.

The system is set to the calibration mode by entering the command 'C'. Channel 2 is selected and with the calibration terminal still grounded, variable resistors VR5 and VR7 indicated in Fig. A.18 are adjusted until a reading of 000800 is obtained. Next, the system is switched to channel 4. Variable resistors VR11 and VR13 indicated in Fig. A.21 are adjusted until a reading of 000800 is obtained. The command 'B' is next entered and various input currents applied to the respective shunts using the variable current source. The corresponding gain setting variable resistors VR6 and VR12 are adjusted until the meter reading corresponds to the reading of the standard ammeter.

After calibrating all channels, the command '4' is entered to enable the overload test.

6.2 Scale factor calculations

The scale factors required for each channel where calculated by determining the ratio between the corresponding A/D input and the input voltage for various input voltages and a fixed load. A fine adjustment of the values obtained was performed in software to produce the following values.

Channel 1 (low range) - 2020/2^11 = 0.986

Channel 1 (high range) - 3638/2^11 = 1.776

Channel 2 (10 A shunt) - 201/2^10 = 0.196

Channel 3 (low range) - 2020/2^11 = 0.986

Channel 3 (high range) - 3638/2^11 = 1.776

Channel 2 (10 A shunt) - 201/2^10 = 0.196

The current and voltage scale factors are employed in obtaining the required power scale factors. The voltage scale factors when multiplied by the 2's complement of the corresponding A/D outputs result in a voltage reading 10 times the actual value. A shift in the decimal point one place to the left thus produces the required value to one decimal place. The voltage ratios for the low range indicate that a 1 LSB change in the channel 1 or 3 A/D converter, results in approximately a 0.1v change in the output value. This thus places a limit on the resolution of the displayed value. Similarly, the corresponding output value change for the high ranges is approximately 0.18v for a 1 LSB change in the output of the A/D converters.

Similarly the current ratios using a 10 amp shunt, indicate that a 1 LSB change in the channel 2 or 4 A/D converter, results in approximately a 0.002A change in the output value. This results because the current reading is computed to two decimal places.

6.3 Performance checks

The meter was compared with a voltmeter, ammeter and digital wattmeter. As indicated in Fig. 6.1, a linear relationship between the computed rms value and input signal amplitude occured when a sinusoidal, triangular and square waveforms at 60hz were applied to the calibration input. The near constant (difference between maximum and minimum values = 5) rms values over a frequency range of 20hz to 1100hz for a sine and square wave input are indicated in Fig. 6.2. The linear relation between the wattmeter reading and the calibrated wattmeter reading are indicated for two loads in Figs. 6.3 and 6.4.

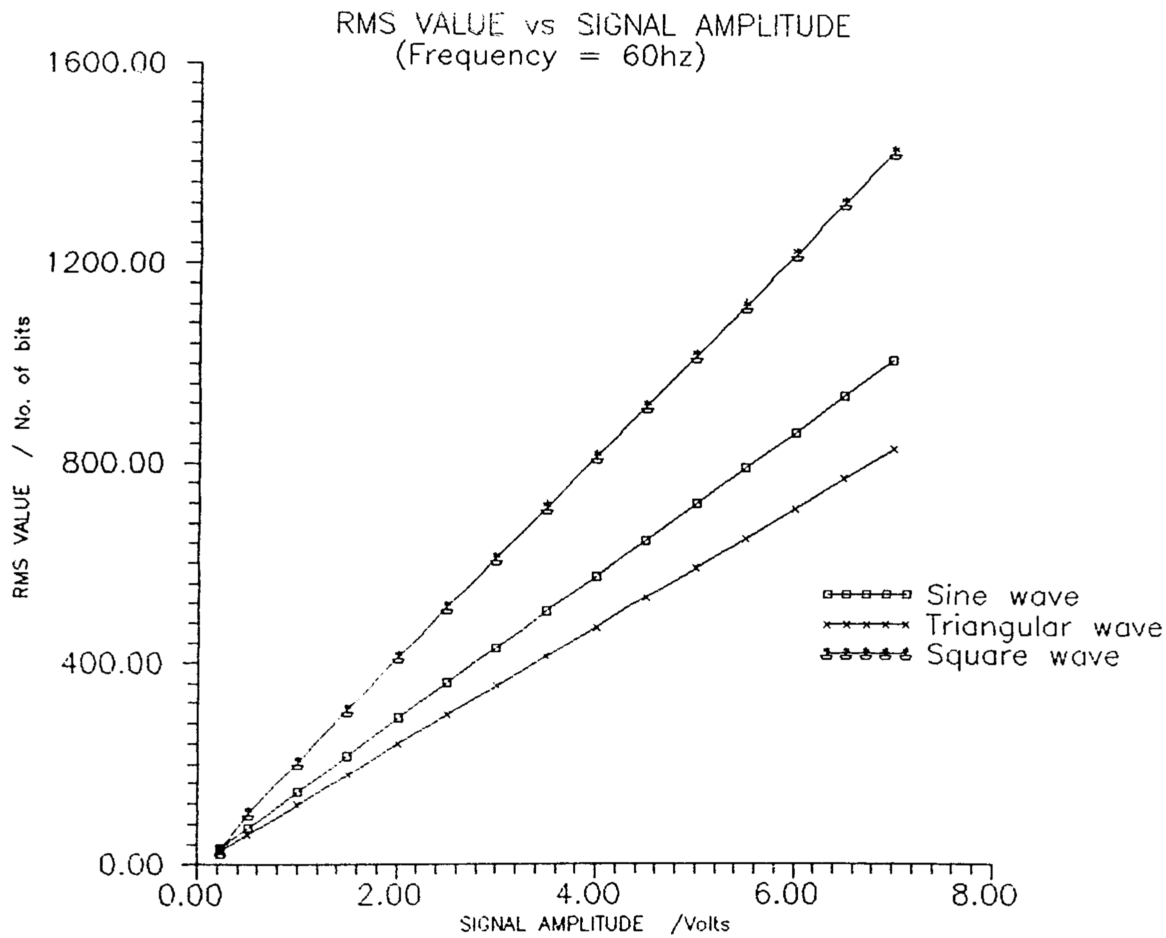

FIG. 6.1 GRAPH OF RMS VALUE vs SIGNAL FREQUENCY (60 Hz)

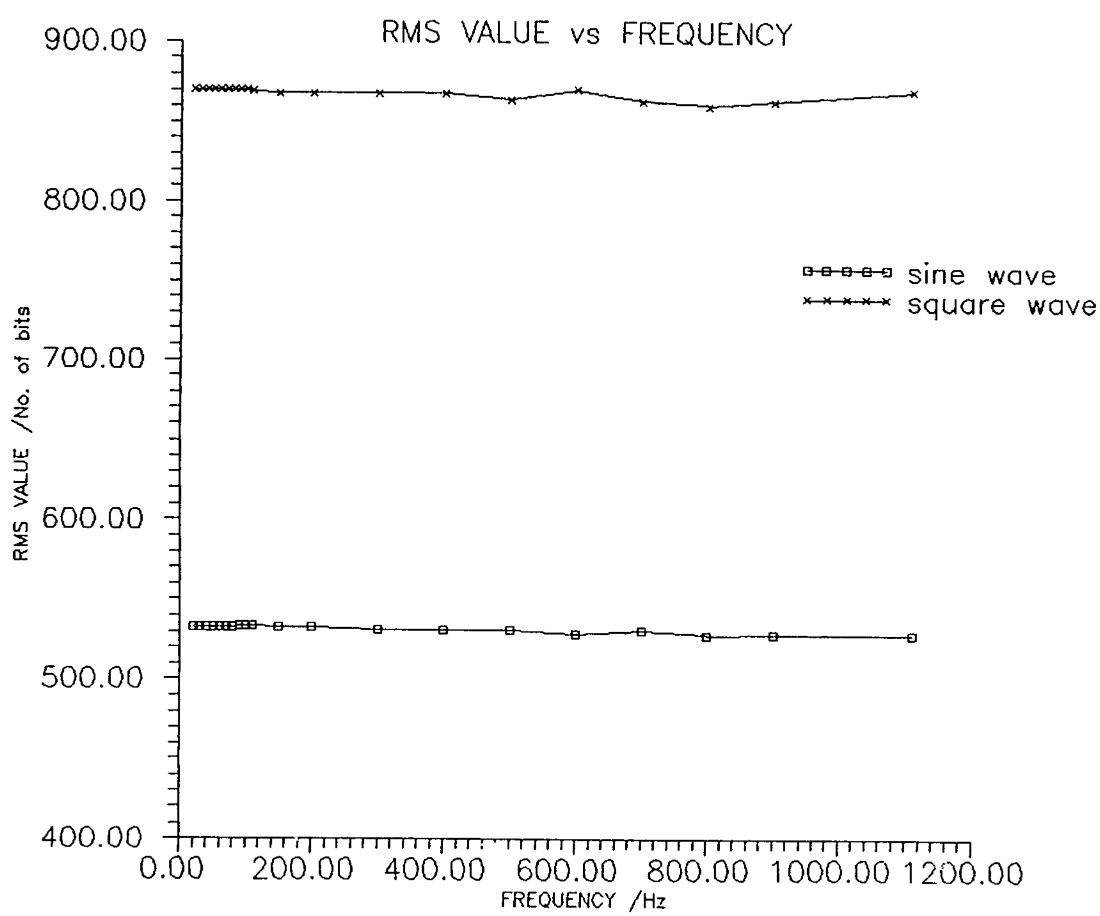

FIG. 6.2 GRAPH OF RMS VALUE vs FREQUENCY

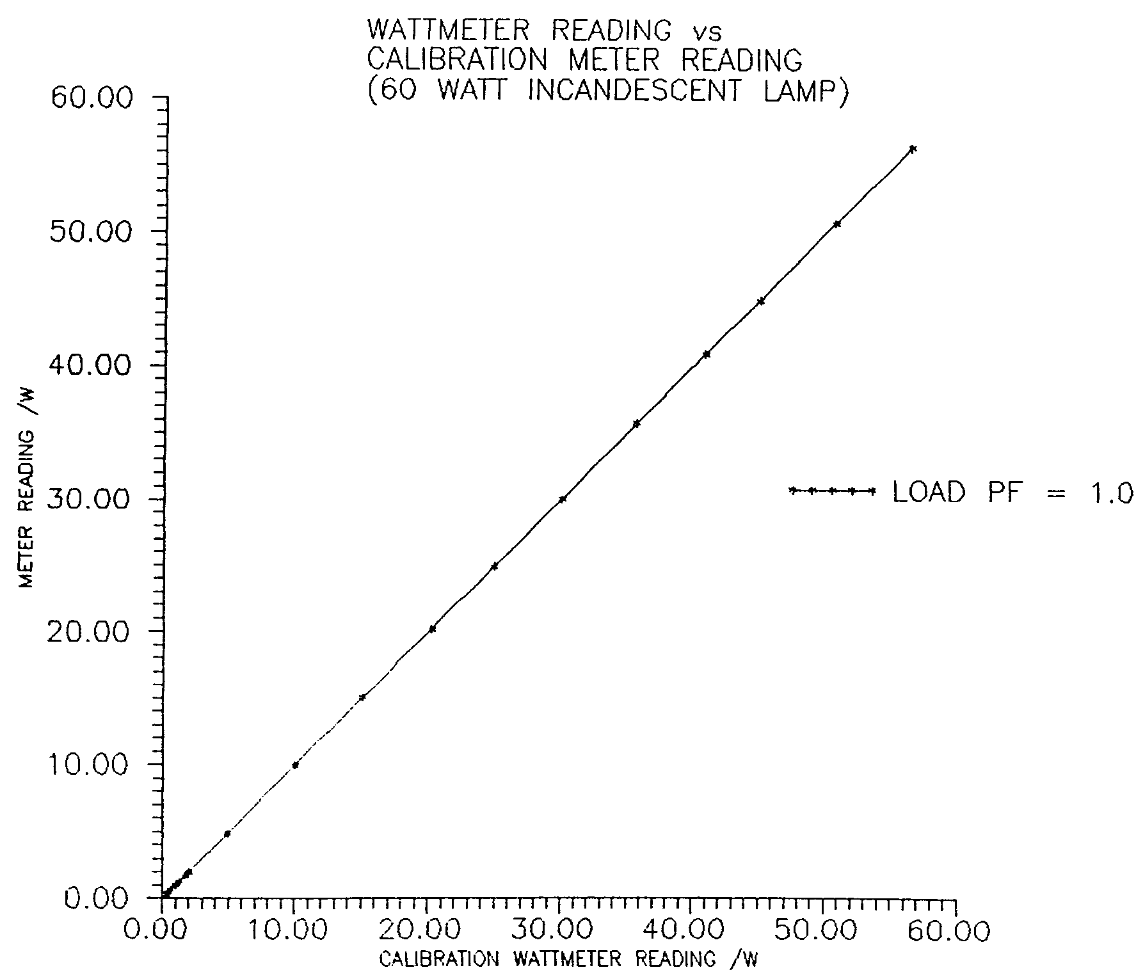

FIG. 6.3 GRAPH OF WATTMETER READING vs CALIBRATION METER READING (60 WATT INCANDESCENT LAMP)

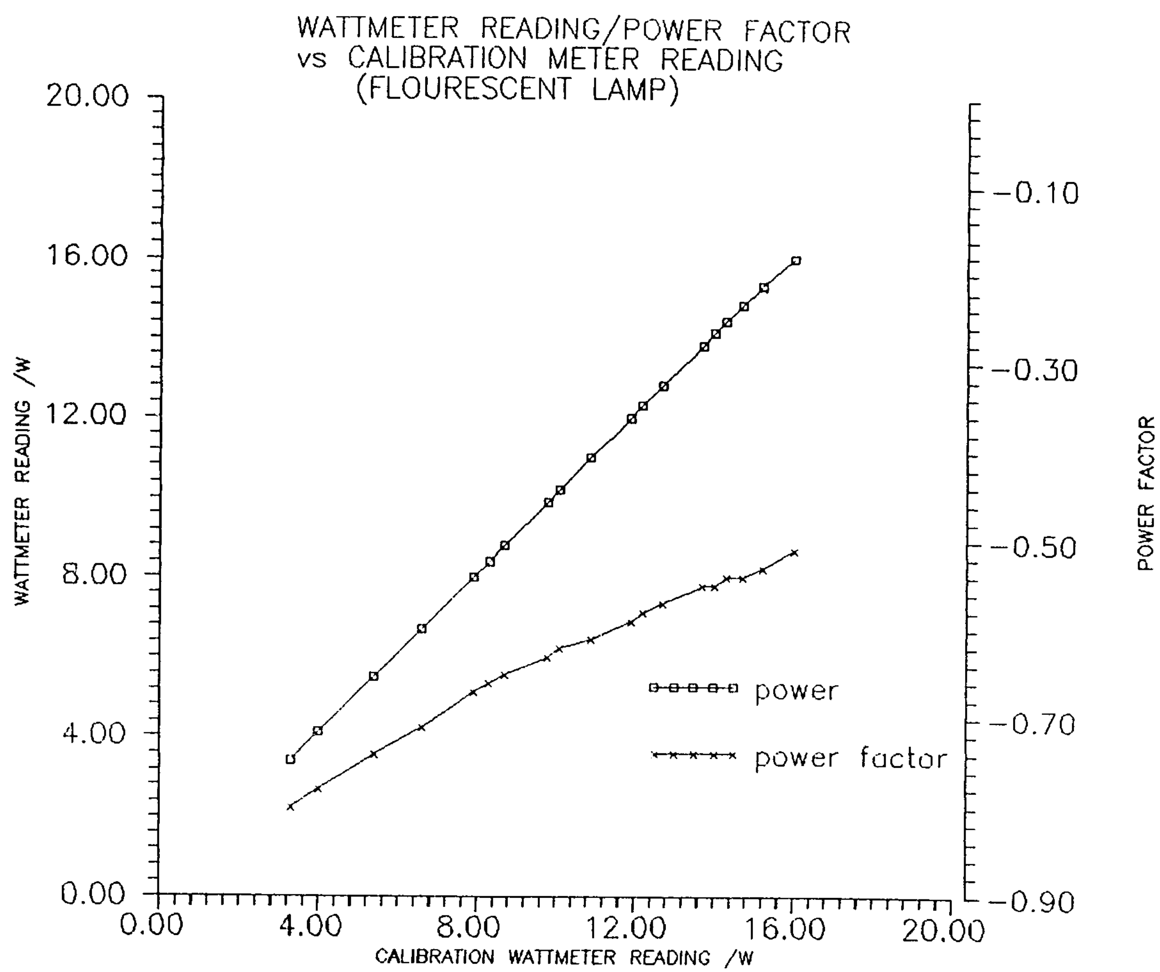

FIG. 6.4 GRAPH OF WATTMETER READING vs CALIBRATION METER READING (FLUORESCENT LAMP)

6.4 Sources of error

The significant sources of error are:

a) Noise primarily due to the switching circuits used.

b) Rounding errors in the arithmetic computations used.

c) Component tolerances especially in the analog section of the system.

The use of 1% tolerance resistors results in worst case common-mode rejection ratios of 275 and 50 for the voltage and current input conditioning circuits respectively.

6.5 Specifications

	Minimum	Maximum
Input voltage /V	-350	+350
Load current /A	-10	+10
Power /W		3500
Frequency /hz	12	1000

Time span for the acquisition, computation and display.

Mode	Maximum time
Autocalibrate(All channels sequentially)	162.1ms
DC (2 channels)	185ms
AC (1-phase)	83.5ms
AC power (3-phase)	67ms
DC power (4 channels)	187ms
Frequency (60 hz signal)	33ms

Power requirements

+5volts	1.2A

+15volts	120mA
-15volts	150mA

Voltages and power readings are read to one decimal place. Current and power factor readings can be read to two decimal places.

6.6 Suggestions for improvements

Although the system met the design specifications, finance and the available components, placed a limitation on some desireable features which could be included. A few improvements that can be made include the incorporation of devices to support a higher sampling rate. Including components with higher frequency bandwidths will enable the upper frequency limit of the instrument to be extended. In addition, an autoranging feature could also be incorporated.

7. CONCLUSION

A digital wattmeter has been designed and constructed. which compared favourably with a wattmeter with a worst case accuracy of 0.3% . Voltage measurements agreed with those obtained on a 6-digit voltmeter. An extension of the trapezoidal rule is employed. This technique, theoretically produces the best accuracy among current numerical integration techniques employed in digital sampling wattmeters. In addition to the electrical power, the voltage current and power factor are also computed. The instrument is also capable off of frequency measurement.

The microprocessor-based instrument can measure the parameters of two loads simultaneously. The hardware uses off-the-shelf components. A GPIB port is provided for integrating the instrument into an automated test setup.

The system software consists of a number of subprograms which are used in acquiring, storing, computing and displaying various parameters and the system status.

The outstanding features of this thesis are:

(a) the first practical implementation of an instrument employing the modified trapezoidal method.

(b) the instrument is easy to use

(c) software is optimized for speed and size

(d) reasonably priced ($500)

REFERENCES

1. CLARKE F. J. J., STOCKTON J. R. "Principles and theory of wattmeters operating on the basis of regularly spaced sample pairs", **Journal of Physics. E, Scientific Instruments**, Vol. 15. 1982, pp. 645-652.

2. CORNEY A.C., PULLMAN R.T. "Digital Sampling Laboratory Wattmeter", **IEEE Transactions on Instrumentation and Measurement, Vol. IM-36, No. 1**, March 1987, pp. 54-59.

3. DEKKER R., BEENKER F., THIJSSEN L. "Fault Modeling and Test Algorithm Development for Static Random Access Memories" **IEEE 1988 International Test Conference,** pages 343-352.

4. FRANCO S., Design with Operational Amplifiers and Analog Integrated Circuits, **McGraw-Hill Book Company**,1988.

5. GREGORY B.A. An Introduction to Electrical Instrumentation, **Macmillan**, 1973.

6. HILL J. J., ANDERSON W. E., "Design of a microprocessor-based digital wattmeter", **IEEE Transactions on Industrial Electronics and Control Instrumentation, Vol. IECI-28, No. 3**, August 1981, pp. 180-184.

7. INTEL MICROPROCESSOR AND PERIPHERAL HANDBOOK, VOL. II, PERIPHERAL , 1988.

8. MATOUKA M.F., "A wide-range digital power/energy meter for systems with nonsinusoidal waveforms." **IEEE**

Transactions on Industrial Electronics, Vol. IE-36, No. 1, February 1982, pp. 18-31.

9. MOTOROLA MICROPROCESSORS DATA MANUAL ,1981.

10. RATHORE T. S. "Theorems on power, mean and RMS values of uniformly sampled periodic signals" **IEE PROCEEDINGS, Vol. 131, Pt. A. No. 8,** November 1984, pp. 598-600.

11. SAY M.G., Electrotechnology Basic Theory and Circuit Calculations for Electrical Engineers, **Newnes-Butterworth,1974.**

12. STENBAKKEN G. N., "A wideband sampling wattmeter," **IEEE Trans. Power App. Syst.,** Vol PAS-103, no. 10 Oct. 1984, pp. 2919-2926.

13. TURGEL R. S. "Digital Wattmeter Using a Sampling Method," **IEEE Trans. on Instrumentation and Measurement**, Vol. IM-23, No. 4, Dec. 1974, pp. 337-341.

14. ZU-LIANG L., "An Error Estimate for Quasi-Integer-Period Sampling and an Approach for Improving its Accuracy", **IEEE Trans. on Instrumentation and Measurement,** Vol. 37. No. 2. June 1988, pp. 219-222.

APPENDIX A WATTMETER CIRCUIT DIAGRAMS

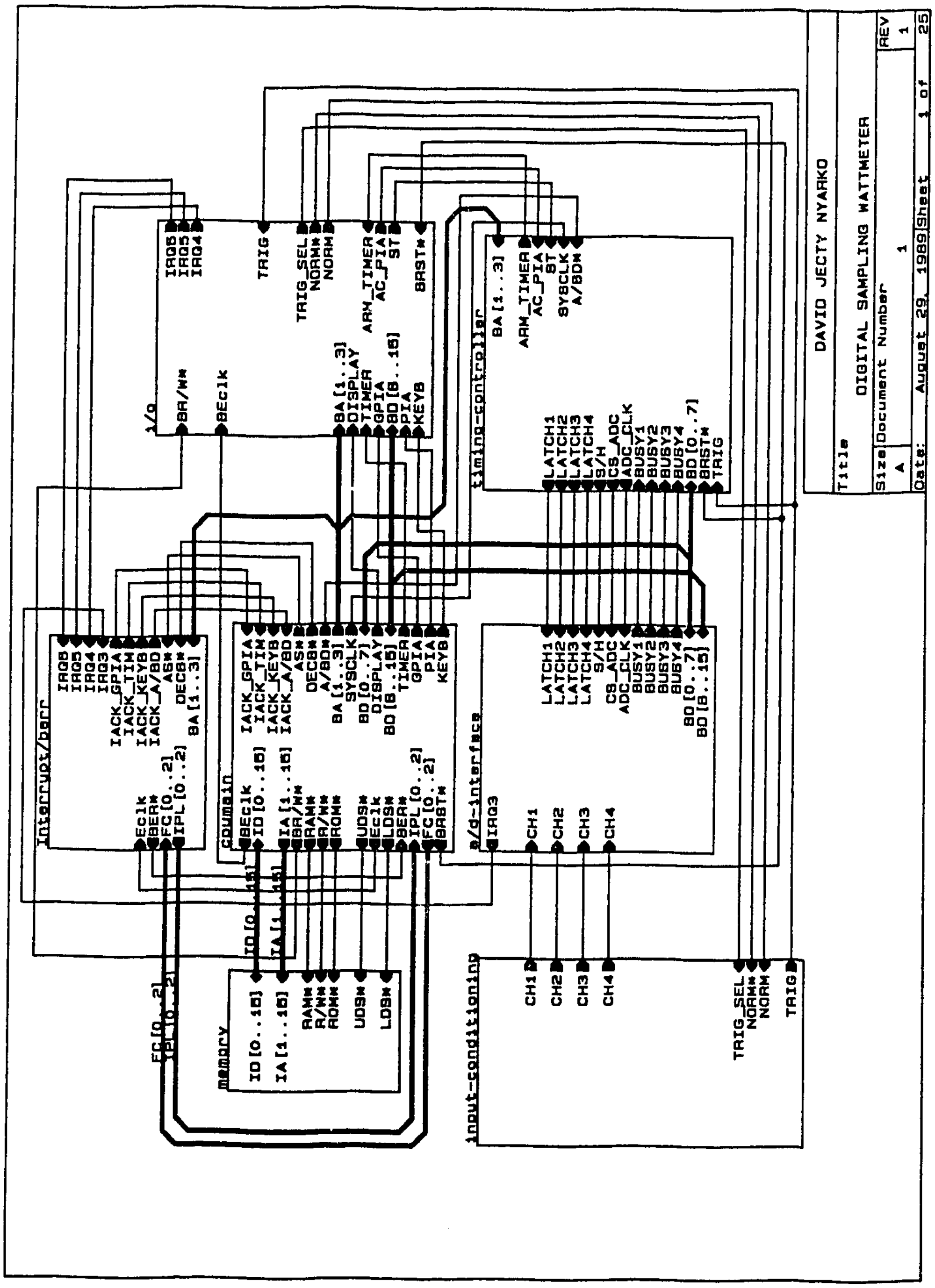

FIG. A.1 BLOCK DIAGRAM OF DIGITAL SAMPLING WATTMETER

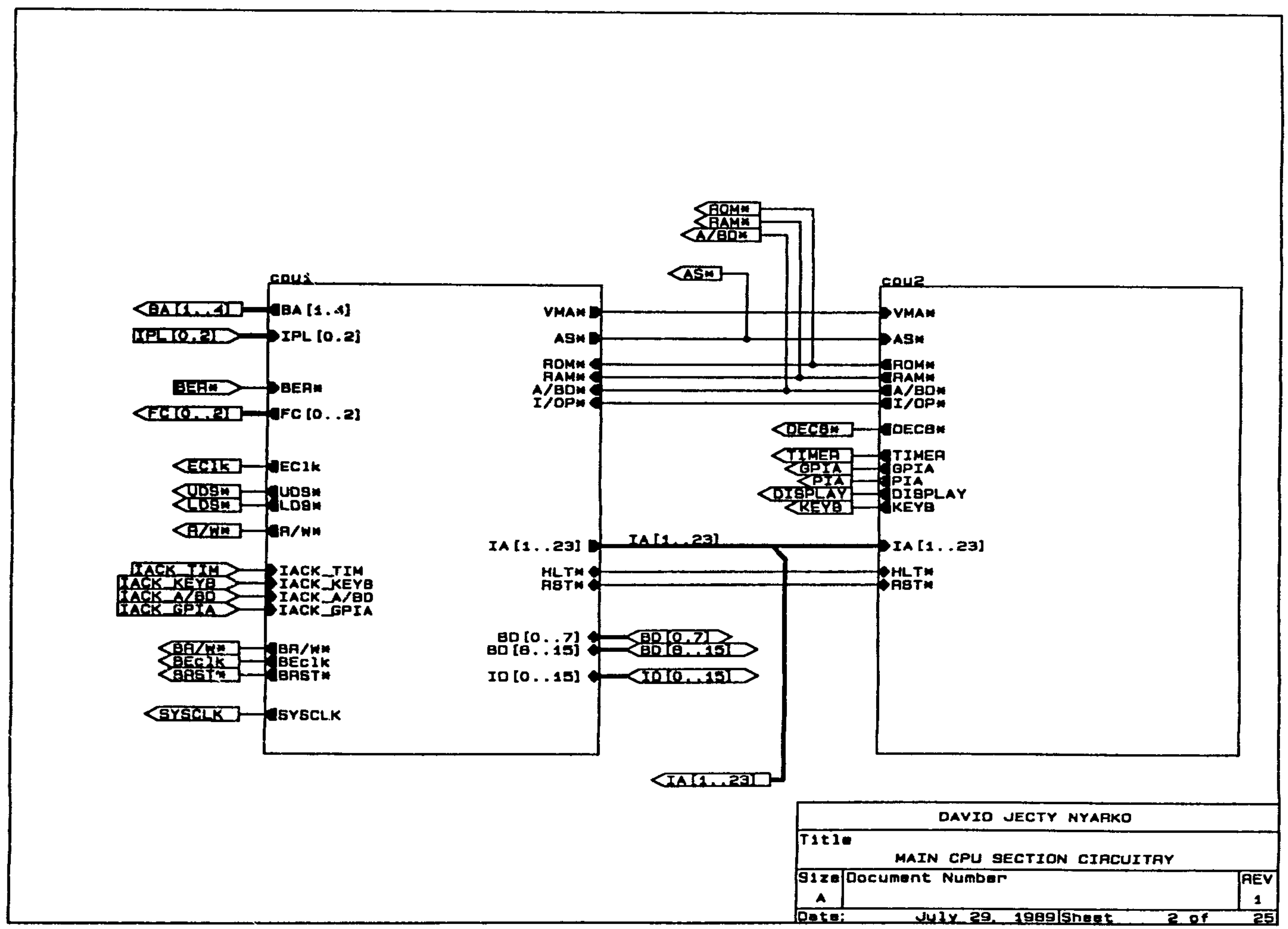

FIG. A.2 BLOCK DIAGRAM OF MAIN CPU CIRCUITRY

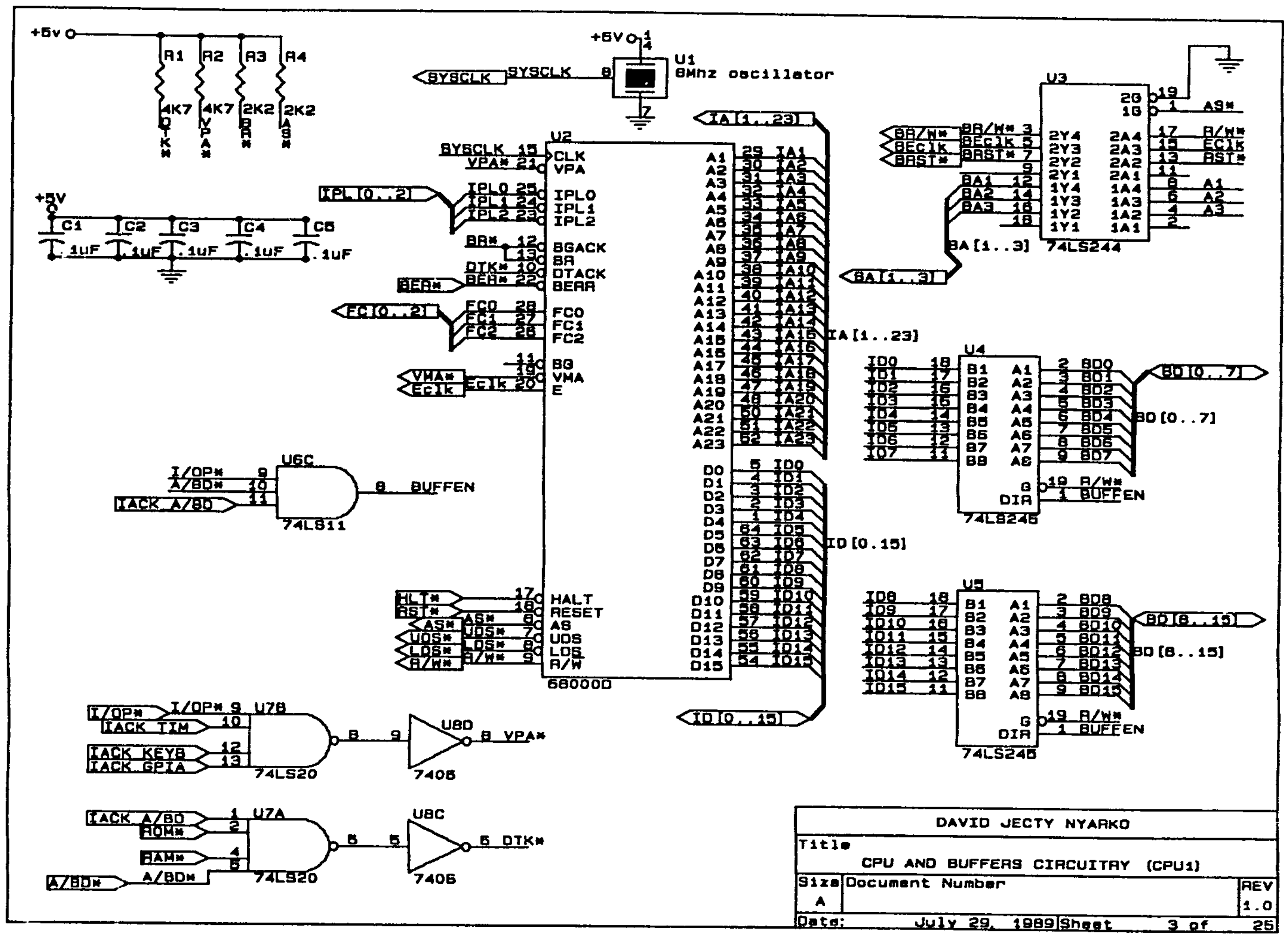

FIG. A.3 DETAILED SCHEMATIC OF CPU AND BUFFERS CIRCUITRY

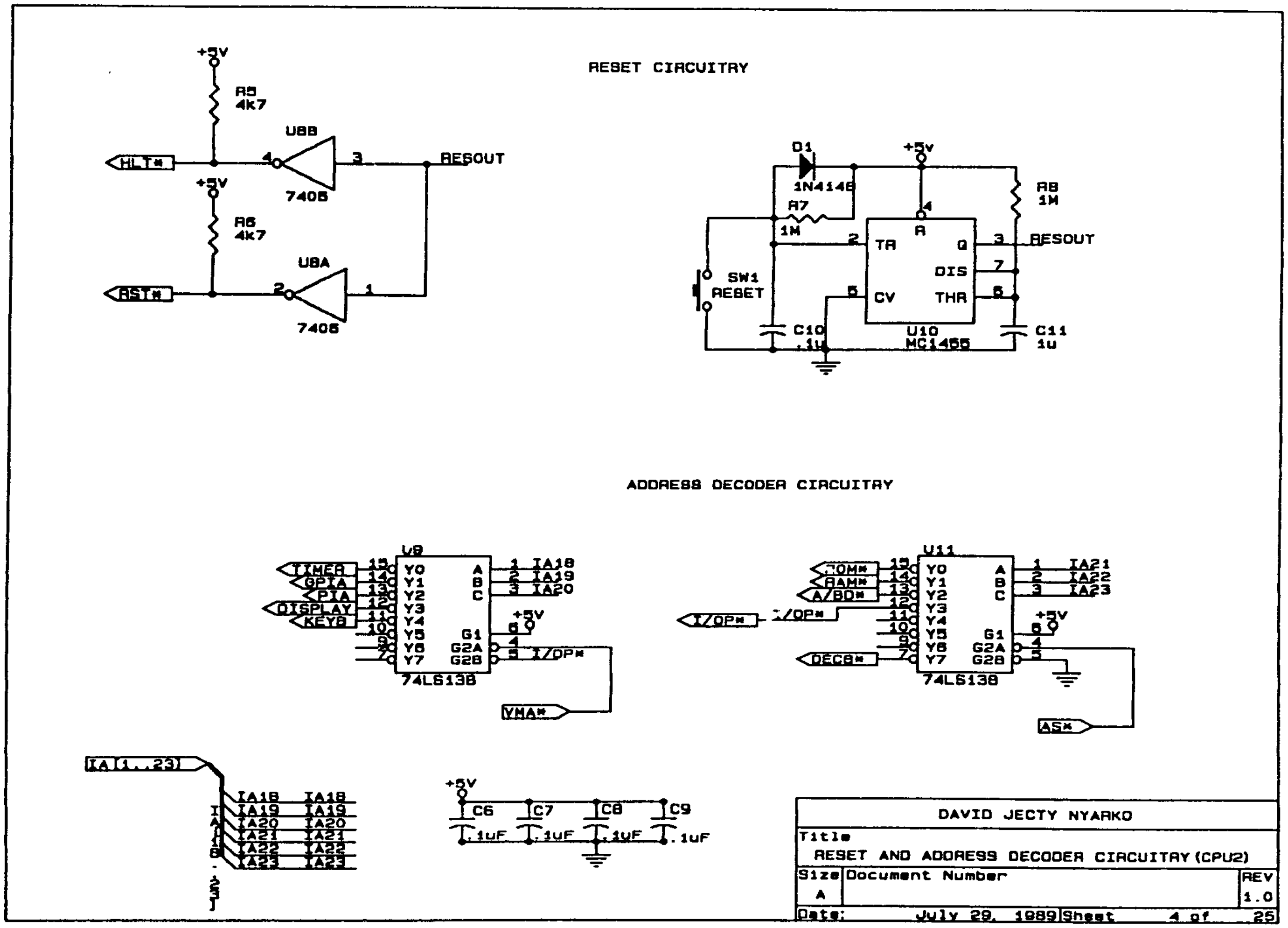

FIG. A.4 DETAILED SCHEMATIC OF RESET AND ADDRESS DECODER CIRCUITRY

```
;
    move.w     #ch1_denom,denom_v1
    move.w     #ch2_denom,denom_i1
    move.w     #ch3_denom,denom_v2
    move.w     #ch4_denom,denom_i2
    move.w     #w1_denom,denom_w1
    move.w     #w2_denom,denom_w2
;
    bsr  watt_fact
    bsr  store_fact
;
; initialize  program mode variables
;
    move.l     #prog_buff,prog_max
    move.l     #prog_buff,prog_pointer
    clr.w      store_size
    move.l     #data_pointer+4,data_pointer
;
        move.b  #%11111111,gpia
        andi.w  #intson,sr        ;enable interrupts
;
    move.w     #$800,cal_buff1
    move.w     #$800,cal_buff2
    move.w     #$800,cal_buff3
    move.w     #$800,cal_buff4
;
    clr.b      menu_state
;
begin   clr.b   state
;
begin_0:
;
    lea.l      menu_state,a0
;
    tst.b      (a0)
    bne.s      begin_1
    bsr  menu0
    bra.s      wait
begin_1:
    cmpi.b     #1,(a0)
    bne.s      begin_2
    bsr  menu1
    bra.s      wait
begin_2:
    cmpi.b     #2,(a0)
    bne.s      begin_3
    bsr  menu2
    bra.s      wait
begin_3:
    cmpi.b     #3,(a0)
    bne.s      begin_4
    bsr  menu3
```

```
    bra.s     wait
begin_4:
    cmpi.b    #4,(a0)
    bne.s     begin_5
    bsr  menu4
    bra.s     wait
begin_5:
    cmpi.b    #5,(a0)
    bne.s     begin_6
    bsr  menu5
    bra.s     wait
begin_6:
    cmpi.b    #6,(a0)
    bne.s     begin_7
    bsr  menu6
    bra.s     wait
begin_7:
    cmpi.b    #7,(a0)
    bne.s     begin_0
    bsr  menu7
wait    andi.b  #%00110000,state
        tst.b   prog_state
        beq.s   wait_norm
    bsr  program
wait_norm:
        lea     key_buff,a0
        tst.b   (a0)            ;0 to start
        beq     start
    cmpi.b    #$12,(a0) ;a for recal (dc only)
    bne.s     waitb
    bset.b    #7,state
    bra.s     wait_3
waitb    cmpi.b  #$13,(a0)        ;b to sample
        beq     cont
    cmpi.b    #$11,(a0) ;9 to calibrate and compute
    bne.s     waitc           ;rms value
    bset.b    #6,state
    bra.s     wait_2
waitc    cmpi.b  #$18,(a0)        ;c to calibrate
        bne.s   waitd
wait_2  ori.b   #calib_on,pia_b
        andi.b  #norm_off,pia_b
        move.b  pia_b,piadb
wait_3   bsr     cal_display
waita0   btst.b    #0,state1
    beq.s     waita1
    move.b    auto_chan,(a0)
    bra.s     waita2
waita1  tst.b   prog_state
        beq.s   waita2
    bsr  program
waita2   tst.b    (a0)
```

```
        beq.s   waita2
        cmpi.b  #8,(a0)
        bne.s   waita3
        move.b  #4,(a0)
waita3  cmpi.b  #4,(a0)
        bhi.s   waita2
        move.b  (a0),cal_chan
        bra   cont
waitd   cmpi.b  #$19,(a0)       ;d for no. of acquisitions
        bne.s   waite
    bsr  clear
    moveq     #line3,d3
    lea  messg24(pc),a5 ;display ' SAMPLE SIZE'
    bsr  display
        bsr     acq_disp
        lea     acq_count,a1
        bsr     acq
        move.w  (a1),acq_count
        bra     begin_0
waite   cmpi.b  #$1a,(a0)       ;e for delay
        bne.s   wait1
    bsr  clear
    moveq     #line3,d3
    lea  messg25(pc),a5 ;display ' DELAY'
    bsr  display
        bsr     acq_disp
        lea     delay_value,a1
        bsr     acq
        move.w  (a1),delay_count
        bra     begin_0
wait1   cmpi.b    #1,(a0)   ;1 for menu display on/off
    bne.s      wait4
    bchg.b    #0,pia_a
    move.b    pia_a,piada    ;turn led 0 on/off
    bchg.b    #2,disp_state
    bsr  disp_ctrl
    move.b    #$ff,(a0) ;force return to menu
;
wait4   cmpi.b    #8,(a0)   ;4 for overload test on/off
    bne.s      wait7
    bchg.b    #2,state1
    bchg.b    #1,pia_a
    move.b    pia_a,piada
    move.b    #$ff,(a0)
wait7   cmpi.b    #$b,(a0)  ;7 pressed
    bne.s     wait2
    bchg.b    #2,pia_a
    bchg.b    #0,pia_b
    move.b    pia_a,piada
    move.b    pia_b,piadb
    move.b    #$ff,(a0)
;
```

```
wait2    cmpi.b    #2,(a0)         ;channel 1 high/low
    bne.s     wait3
    bchg.b    #3,pia_a
    move.b    pia_a,piada
    btst.b    #3,pia_a
    beq.s     wait2a
    bset.b    #6,state1 ;high range
    bra.s     wait2b
wait2a   bclr.b    #6,state1 ;low range
;
wait2b   bsr  store_fact
    move.b    #$ff,(a0) ;force return to menu
;
wait3    cmpi.b    #3,(a0)    ;channel 3 high/low
    bne.s     wait5
    bchg.b    #4,pia_a
    move.b    pia_a,piada
    btst.b    #4,pia_a
    beq.s     wait3a
    bset.b    #7,state1
    bra.s     wait3b
wait3a   bclr.b    #7,state1
wait3b   bsr  store_fact
    move.b    #$ff,(a0)
    bra  begin_0
wait5    cmpi.b    #9,(a0)    ;5 for scale factor numerators
    bne.s     wait6
    move.b    #$ff,(a0)
    move.w    #'N0',d2
    lea  numer_v1-4,a1
    bsr  factor
    bra  begin_0
;
wait6  cmpi.b #$a,(a0)  ;6 for scale factor denominators
    bne.s     waitc_0
    move.b    #$ff,(a0)
    move.w    #'D0',d2
    lea  denom_v1-4,a1
    bsr  factor
    bra  begin_0
waitc_0:
    cmpi.b    #$80,(a0) ;^0 for entering shunt values
    bne.s     waitc_1
    move.b    #$ff,(a0)
    bsr  shunt
    bra  begin_0         ;return to main menu
waitc_1:
    cmpi.b    #$81,(a0) ;^1 for lines remaining
    bne.s     waitc_2
    bsr  lines_left
    move.b    #$ff,(a0)
    bra  begin_0
```

```
waitc_2:
    cmpi.b    #$82,(a0) ;^2 display data
    bne.s     waitc_3
    bsr  disp_data
    bra  begin_0        ;return to main menu
waitc_3:
    cmpi.b    #$83,(a0) ;^3 storage on/off
    bne.s     waitc_4
    move.b    #$ff,(a0)
    bchg.b    #5,pia_a
    move.b    pia_a,piada
    btst.b    #5,pia_a
    beq.s     waitc_3a
    bset.b    #3,state1
    bra.s     waitc_4
waitc_3a:
    bclr.b    #3,state1
waitc_4:
    cmpi.b    #$88,(a0) ;^4 for keypad program
    bne.s     waitc_7
    bsr  automode
    bra  begin_0
waitc_7:
    cmpi.b    #$8b,(a0) ;^7 change scale factors
    bne.s     waitc_8
    bsr  sc_fact
    bra  begin_0
waitc_8:
    cmpi.b    #$90,(a0) ;a to perform 2nd calibration
;                on all channels (dc only)
    bne.s     waitc_8a
    move.l    #$01020308,auto_chan
    bra.s     waitc_8b
waitc_8a:
    cmpi.b    #$9c,(a0)
    bne.s     wait8
waitc_8b:
    bset.b    #7,state
    bset.b    #0,state1
    bset.b    #0,state
    clr.w     recal_point
    bra  waita0
wait8    cmpi.b    #$10,(a0) ;8 to clear data
    bne.s     waitf
    clr.w     store_size
    move.l    #data_pointer+4,data_pointer  ;initialize
data pointer
    move.b    #$ff,(a0)
waitf   cmpi.b  #$1b,(a0)       ;f for frequency
    bne.s     waitc_f
    move.b    #14,meas_state ;in frequency mode
        bra     freq
```

```
waitc_f  cmpi.b    #$9b,(a0) ;^F next menu
    bne.s     waitc_b
    addq.b    #1,menu_state
    bclr.b    #3,menu_state
    move.b    #false_key,(a0)
    bra  begin_0
waitc_b
    cmpi.b    #$93,(a0) ;^B previous menu
    bne  wait
    move.b    #false_key,(a0)
    subq.b    #1,menu_state
    bpl.s     waitend
    andi.b    #%00000111,menu_state
waitend:
    bra  begin_0
cont     btst.b  #0,state
         bne.s   sample_0
         tst.b   prog_state
         beq.s   cont0
;
    bsr  program
;
cont0     bsr     clear
         moveq   #line1,d3
         lea     messg17(pc),a5
         bsr     display
         moveq   #line2,d3
         lea     messg18(pc),a5
         bsr     display
         moveq   #line3,d3
         lea     messg19(pc),a5
         bsr     display
         moveq   #line4,d3
         lea     messg19a(pc),a5
         bsr     display
waitc0    cmpi.b  #1,(a0)
         bne.s   waitc1
         bset.b  #1,state
         bra.s   sample_0
waitc1   cmpi.b  #2,(a0)
         bne.s   waitc2
         bset.b  #2,state
         bra.s   sample_0
waitc2   cmpi.b  #3,(a0)
         bne.s   waitc0
         bset.b  #3,state
sample_0:
         bsr     clear
         move.b  #$ff,key_buff
sample:
         bsr     init_samp
    bsr  init_sampac
```

```
         btst.b  #0,state
         beq.s   sample_a
     btst.b    #6,state  ;if rms value is
     bne.s     sample_a  required, continue
     btst.b    #7,state
     bne.s     sample_a  ;if in recal mode,skip
         clr.l   d2
         clr.l   d3
         clr.l   d4
         clr.l   d5
         bra.s   samp0
sample_a:
         move.w  cal_buff1,d2
         move.w  cal_buff2,d3
         move.w  cal_buff3,d4
         move.w  cal_buff4,d5
samp0    addq.l  #8,a5
samp1    tst.b   (a6)        ;if low freq, use dc mode
         beq     dc_samp     ;else, use ac mode
         btst.b  #0,(a0)      ;data stored in latches?
         beq.s   samp1
         move.w  (a4),(a5)
         move.w  (a3),-(a5)
         move.w  (a2),-(a5)
         move.w  (a1),-(a5)
         sub.w   d2,(a5)+
         sub.w   d3,(a5)+
         sub.w   d4,(a5)+
         sub.w   d5,(a5)
         move.b  (a0),d1
     move.b    d1,d7
     and.w     d0,d1
         jmp     zc_tab(pc,d1.w)
zc_tab:
         bra.s   sample
         bra.s   sample
         bra.s   sample
         bra.s   sample
         bra.s   stg_1a  ;1 more sample required
         bra.s   samp0    ;just started sampling,continue
         bra.s   stg_2a  ;2 more samples required
         bra.s   samp0   ;continue sampling
stg_2a   addq.l  #8,a5
stg_2    btst.b  #0,(a0)
         beq.s   stg_2
         move.w  (a4),(a5)
         move.w  (a3),-(a5)
         move.w  (a2),-(a5)
         move.w  (a1),-(a5)
         sub.w   d2,(a5)+
         sub.w   d3,(a5)+
         sub.w   d4,(a5)+
```

```
        sub.w   d5,(a5)
stg_1a  addq.l  #8,a5
stg_1   btst.b  #0,(a0)
        beq.s   stg_1
        move.w  (a4),(a5)
        move.w  (a3),-(a5)
        move.w  (a2),-(a5)
        move.w  (a1),-(a5)
;
        sub.w   d2,(a5)+
        sub.w   d3,(a5)+
        sub.w   d4,(a5)+
        sub.w   d5,(a5)
;
;
    andi.b  #stac_dis,pia_b  ;start' and ac'(to clear ffs)
from pia
    ori.b   #key_en,pia_b  ;enable keyboard interrupts
    move.b  pia_b,piadb
;
    asr.b     #4,d7
    bmi.s     stg_3
    subq.l    #8,a5           ;adjust a5
;
stg_3   ext.w    d7
    move.w    d7,alpha16_store    ;store 16 * delta
    btst.b    #2,state1
    bne.s     samp1a          ;neglect overload test
    bsr  ov_load
    btst.b    #1,state1
    beq.s     samp1a          ;no overload
    bset.b    #0,disp_state
    bsr  disp_ctrl ;set cursor blinking
    bra  samp4b1a
samp1a   bclr.b    #0,disp_state
    bsr  disp_ctrl ;set blinking cursor off
    bsr  clear_regs
        move.l  a5,samp_end
        move.l  a5,d5
        sub.l   #$20000e,d5
        asr.l   #3,d5
        asl.w   #4,d5    ;multiply by 16 (16n)
        lea     $200008,a1  ;point to start of samples (Y0
chan 1)
    bsr  clear_regs
        btst.b  #0,state
        beq.s   samp3
    btst.b    #6,state
    bne.s     samp2b
      bsr     ac_calibrate
    move.b    #0,meas_state  ;in ac_calibrate state
    bsr  sign
```

```
samp2a  lea     disp_buff,a5
    move.w    #'R0',d1
    add.b     cal_chan,d1
    move.w    d1,(a5)+
    bsr  update
        bra     samp4
samp2b  bsr   ac_rms
    move.b    #4,meas_state  ;in ac_rms state
    bra.s     samp2a
samp3    btst.b    #3,state  ;3-phase mode selected?
    beq.s     samp3_0
    move.b    #10,meas_state ;in 3-phase ac mode
    bsr  phase3        ;compute power of both phases
    bsr  pf3_100       ;calculate 100 * pf(power factor)
    lea  disp_buff,a5
    move.l    #'PF =',(a5)+
    move.w    pf_store,d0
    move.w    #'  ',(a5)+
    move.w    Vmax,d1
    cmp.w     Imax,d1
    bcc.s     lead0
    move.w    #' -',(a5)+    ;lagging power factor
    bra  samp3_2
lead0    move.w    #' +',(a5)+    ;leading power factor
samp3_2:
    move.w    d0,(a5)
    moveq.l   #2,d5          ;two bytes
    bsr  hex_ascii ;convert to ascii
    move.b    1(a5),d0
    move.b    #'.',1(a5)     ;fix decimal point
    move.b    d0,(a5)
    bra  samp4
samp3_0:
    btst.b  #1,state
        beq.s   samp3a
        addq.l  #8,a1   ;point to Y1 chanl
        movea.l a1,a4
        movea.l a5,a2
        movea.l a5,a3
;
; for Vrms
;
        subq.l  #6,a2
        subq.l  #6,a3
        bra.s   samp3b
samp3a  btst.b  #2,state
        bne.s   samp3a_0
samp3a_0:
        adda.l  #12,a1    ;point to Y1( chan 3)
        movea.l a1,a4
        movea.l a5,a2
        movea.l a5,a3
```

```
        subq.l  #2,a2
        subq.l  #2,a3
samp3b  move.b    #6,meas_state  ;in normal ac mode
    move.l    a0,-(sp)
    lea  Vmax,a0
    bsr  max_sample
    move.l     (sp)+,a0
    bsr      correct
    bsr      evaluate
        move.l  4(a0),d0
        bsr     sqrt
    movem.l  d1/d2/a5/a6,-(sp)
;
    clr.l      d2
    btst.b     #1,state
    beq.s      fac_v1
    move.w     denom_v1,d2
    muls.w     numer_v1,d0
    bra.s      fac_v3
;
fac_v1  btst.b    #2,state
    bne.s      fac_v2
    movea.l    #6,a2          ;TEST
    bra  ramerr
fac_v2  move.w    denom_v2,d2
    muls.w     numer_v2,d0
fac_v3  asr.l      d2,d0      ;for 10 * result( to 1dp)
    bcc.s      fac_v4
fac_v4  move.l     d0,d1
    clr.l      d0          ;prepare for routine
    bsr  bcd1        ;convert to bcd
        lea.l   disp_buff1,a5
        move.b  #'V',(a5)+
        bsr     update
    bsr  blank
    move.b     (a6),2(a6)      ;move termination character
    move.b     #'V',1(a6)
    move.b     -1(a6),(a6)
    move.b     #'.',-1(a6)     ;insert decimal point
;
        movem.l  (sp)+,d1/d2/a5/a6     ;restore d1/d2/a5/a6
;
; I rms
;
        addq.l  #2,a1        ;initialize to Y1
        addq.l  #2,a2        ;initialize to Yn
        addq.l  #2,a3        ;initialize to Yn
        addq.l  #2,a4        ;initialize to Y1
    bsr  clear_regs
    move.l     a0,-(sp)
    lea  Imax,a0
    bsr  max_sample
```

```
    move.l    (sp)+,a0
    bsr     correct
        bsr     evaluate
        move.l  4(a0),d0
    move.l    d0,Imsqr
        bsr     sqrt
    movem.l  d1/d2/a5/a6,-(sp)
    clr.l     d2
    btst.b    #1,state
    beq.s     fac_i1
    move.w    denom_i1,d2
    muls.w    numer_i1,d0
    bra.s     fac_i3
fac_i1  btst.b    #2,state
    bne.s     fac_i2
    movea.l   #7,a2          ;TEST
    bra  ramerr
fac_i2  move.w    denom_i2,d2
    muls.w    numer_i2,d0
fac_i3  asr.l     d2,d0     ;for 100 * result (to 2 dp)
    bcc.s     fac_i4
    addq.l    #1,d0          ;round up
fac_i4  move.l    d0,d1
    clr.l     d0        ;prepare for routine
    bsr  bcd1      ;convert to bcd
        lea.l   disp_buff2,a5
        move.b  #'I',(a5)+
        bsr     update
    bsr  blank
    move.b    (a6),2(a6)     ;move termination character
    move.b    #'A',1(a6)
    move.b    -1(a6),(a6)
    move.b    -2(a6),-1(a6)
    move.b    #'.',-2(a6)    ;insert decimal point
        movem.l  (sp)+,d1/d2/a5/a6      ;restore d1/d2/a5
;
;  W (power)
;
        subq.l  #2,a1
        subq.l  #2,a2
    bsr  clear_regs
        bsr     correct
        bsr     evaluate
        move.l  4(a0),d0        ;save a5
    move.l    d0,Watt
    bsr  watt_bcd
    move.l    a5,-(sp)
        lea.l   disp_buff3,a5
        move.b  #'W',(a5)+
        bsr     update
    bsr  blank
    move.b    (a6),2(a6)
```

```
    move.b    #'W',1(a6)
    move.b    -1(a6),(a6)
    move.b    #'.',-1(a6)
    move.l    (sp)+,a5
;
    bsr  pf_100          ;calculate 100 * pf(power factor)
;
    lea  disp_buff,a5
    move.l    #'PF =',(a5)+
    move.w    pf_store,d0
    move.w    #'  ',(a5)+
    move.w    Vmax,d1
    cmp.w     Imax,d1
    bcc.s     lead
    move.w    #' -',(a5)+     ;lagging power factor
    bra.s     samp3c_0
lead     move.w    #' +',(a5)+     ;leading power factor
;
    move.w    d0,(a5)
    moveq.l   #4,d5
    bsr  hex_ascii
    move.b    1(a5),d0
    move.b    #'.',1(a5)
    move.b    d0,(a5)
;
samp4    btst.b    #5,state
    beq.s     samp4_0
    subq.w    #1,delay_count
    bne  samp4b1a
;
samp4_0 move.w  delay_value,delay_count
        moveq   #line1,d3
        btst.b  #0,state
        beq.s   samp4a
        lea     messg9(pc),a5
        bsr     display
        moveq   #line3,d3
        lea     disp_buff,a5
        bsr     display
;
        bra.s samp4a4
samp4a   moveq     #line1,d3
    lea  disp_buff,a5   ;display PF(power factor)
    bsr  display
    moveq   #line2,d3
    lea     disp_buff1,a5   ;display VOLTAGE/POWER 1
    bsr     display
;
    moveq   #line3,d3
    lea     disp_buff2,a5   ;display CURRENT/POWER 2
    bsr     display
    moveq     #line4,d3
```

```
    btst.b    #3,state
    beq.s     samp4a2
    lea      messg26(pc),a5
    bra.s     samp4a3
samp4a2:
    lea      disp_buff3,a5    ;display POWER
samp4a3:
    bsr      display
samp4a4:
    btst.b    #3,state1
    beq.s     samp4b
;
    bsr  store_data
;
samp4b  btst.b  #4,state
        beq.s   samp4b1a
    subq.w    #1,acq_count
    beq.s     samp4b2
samp4b1a:
    cmpi.b  #$ff,key_buff
        beq     sample
samp4b2 btst.b  #0,state
        beq     begin_0
        ori.b   #norm_on,pia_b
        andi.b  #calib_off,pia_b
        move.b  pia_b,piadb
        bra     begin_0
;
dc_samp:
        bsr     init_samp
;
        andi.b  #stac_dis,pia_b ;start' and ac' from pia
        andi.b  #key_dis,pia_b
        move.b  pia_b,piadb
        move.l  #1025,d6
        btst.b  #0,state
        beq.s   dc_sampa
        btst.b  #7,state
        bne.s   dc_sampa     ;skip if in recal mode
        clr.l   d2
        clr.l   d3
        clr.l   d4
        clr.l   d5
        bra.s   dc_samp0
dc_sampa:
        move.w  cal_buff1,d2
        move.w  cal_buff2,d3
        move.w  cal_buff3,d4
        move.w  cal_buff4,d5
;
dc_samp0:
        addq.l  #8,a5
```

```
dc_samp1:
        btst.b  #0,(a0)
        beq.s   dc_samp1
        move.w  (a4),(a5)
        move.w  (a3),-(a5)
        move.w  (a2),-(a5)
        move.w  (a1),-(a5)
        sub.w   d2,(a5)+
        sub.w   d3,(a5)+
        sub.w   d4,(a5)+
        sub.w   d5,(a5)
;
        dbra    d7,dc_samp0
;
        ori.b   #key_en,pia_b  ;enable keyboard interrupts
        move.b  pia_b,piadb
;
    btst.b    #2,state1
    bne.s     dc_samp0a ;neglect overload test
    bsr  ov_load
    btst.b    #1,state1
    beq.s     dc_samp0a
    bset.b    #0,disp_state
    bsr  disp_ctrl ;set cursor blinking
    bra  dc_samp2b1
;
dc_samp0a:
    bclr.b    #0,disp_state
    bsr  disp_ctrl ;set cursor blinking
    bsr  clear_regs
    move.l  a5,samp_end
    lea     $200008,a1  ;point to start of samples( Y0)
;                                  (channel 1)
    btst.b  #0,state
    beq.s   dc_samp1a
    move.b    #2,meas_state  ;in calibrate dc mode
        bsr     dc_calib
        bsr     sign
        lea     disp_buff,a5
    move.w    #'R0',d1
    add.b     cal_chan,d1
    move.w    d1,(a5)+
        bsr     update
        bra        dc_samp1c
dc_samp1a:
    btst.b    #4,state
    beq  dc_samp1a0
    adda.l    #12,a1          ;point to Y1 chan 3
    movea.l   a5,a2
    movea.l   a5,a3           ;point to Yn chan 4
    subq.l    #2,a2           ;point to Yn chan 3
;
```