

ON-LINE IMPEDANCE MEASUREMENT  
OF TRANSMISSION LINES USING A  
MICROPROCESSOR AND A FAST MULTIPLIER

by



MEERA KULKARNI, M.Tech. (Indian Institute of Technology, Bombay)

A Thesis

Submitted to the Faculty of Graduate Studies  
in Partial Fulfilment of the Requirements  
for the Degree  
Master of Engineering

McMaster University

October 1980

1  
MASTER OF ENGINEERING  
(Electrical and Computer Engineering)

McMaster University  
Hamilton, Ontario, Canada

TITLE: On-Line Impedance Measurement  
of Transmission Lines Using A Microprocessor  
and A Fast Multiplier

AUTHOR: Meera Kulkarni, M.Tech. (Indian Institute of Technology,  
Bombay)

SUPERVISOR: Dr. R. Kitai, D.Sc.

NUMBER OF PAGES: ix, 83

### ABSTRACT

A system is designed for calculating on-line digital impedance of a transmission line using a Microprocessor (Intel 8085) and a Fast Multiplier (TRW TDC 1003J). The approach taken is to use a digital filter to extract the fundamental component of voltage  $V$  and current  $I$ , then the impedance is given by  $V/I$ . The filter is of the sliding or running spectral measurement type. The total time taken for impedance calculation is 1.033 msec for 16 samples in a cycle, at the power line frequency of 60 Hz. The maximum absolute error is  $\pm 1.52\%$  of the largest possible impedance value.

## ACKNOWLEDGEMENTS

The author particularly wishes to thank Prof. R. Kitai for his helpful advice and guidance during the course of this work. The author would also like to thank Mr. Munniappan for some very useful discussions, and Mrs. Janet Schell for typing this thesis.

Finally, the author would like to thank McMaster University for financial assistance received during the course of this thesis.

## TABLE OF CONTENTS

	<u>Page</u>	
CHAPTER I	INTRODUCTION	1
CHAPTER II	DIFFERENT TECHNIQUES OF DIGITAL IMPEDANCE CALCULATION	3
	2.1 Introduction	3
	2.2 Fundamental Consideration	3
	2.3 Differential Equation Approach	4
	2.4 Steady State Approach	7
	2.5 Sliding or Running Spectral Measurement	17
CHAPTER III	SYSTEM HARDWARE	22
	3.1 Data Acquisition	22
	3.2 Selection of Microprocessor	28
	3.3 Digital Filtering	30
	3.4 Choice of the Multiplier	30
	3.5 Hardware Implementation of the Digital Filter	31
	3.6 Sequence of the Filter Operation	35
	3.7 Implementation of Division	37
CHAPTER IV	SOFTWARE	41
	4.1 Introduction	41
	4.2 Initialization	41
	4.3 Interrupt	43
	4.4 Subroutine DATAIN	44
	4.5 Calculating $[X(n)-X(n-N)]$	44
	4.6 Filter Algorithm	44

	<u>Page</u>
CHAPTER IV (cont'd)	
4.7 Division Algorithm	48
4.8 Subroutine DIVIDE16	49
4.9 Impedance Algorithm	50
4.10 Fault Routine	50
CHAPTER V DISCUSSION	51
5.1 Test Results	51
5.2 Software Simulation of Filter	51
5.3 Accuracy	54
5.4 Execution Time	56
5.5 Size	56
5.6 Extensions and Improvements	56
APPENDICES	
APPENDIX A	62
B	63
C	66
D	71
E	72
F	83
G	84
H	85

## LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
2.1	Trip characteristic with typical R,L plot	4
2.2	First sixteen Walsh functions of integral index k	11
2.3	FIR implementation of spectrum analysis	19
2.4	Recursive implementation of spectrum analysis	19
3.1	System block diagram	23
3.2	Analog multiplexer	24
3.3	Block diagram for channel select	26
3.4	Waveforms for channel select	26
3.5	Waveforms for A/D converter	27
3.6	12 bit to 16 bit conversion	27
3.7	Schematic for SDK85 single board microcomputer	29
3.8	Filter block diagram	32
3.9	Control ports	34
3.10	Block diagram for division	38
3.11	block diagram for square-root	38
4.1	Flow chart for digital impedance calculation	42
5.1	Test result	55
5.2	Photograph of impedance-calculation system	57

LIST OF TABLES

<u>Table</u>		<u>Page</u>
3.1	Control signals for multiplexers	34
4.1	Port assignment	43
4.2	Memory map for data storage	45
4.3	Calculation of $[X(n)-X(n-N)]$	46
5.1	Stored samples of the experiment	52
5.2	Results of the test program	53
5.3	Number of instructions in each section of impedance calculation program	58



LIST OF SYMBOLS AND ABBREVIATIONS

C <sub>lk</sub>	clock
F <sub>1</sub> , F <sub>2</sub>	Fourier coefficients
I/O	Input/Output
L	series inductance
N	Number of samples in one cycle
R	series resistance
RAM	Random Access Memory
ROM	Read Only Memory
S <sub>n</sub>	filtered output of sample X(n)
SAR	Successive Approximation Register
W <sub>k</sub>	Walsh coefficients
X(n)	current sample value
X(n-N)	Nth previous sample value
Z	magnitude of impedance

## CHAPTER I

### INTRODUCTION

The use of a digital computer for distance protection of transmission line was first suggested about a decade ago [1]. This concept requires that digital samples of line voltages and currents be obtained and analysed by a digital computer, using a program (algorithm) that calculate the series impedance of the line. Advances have since been made [2,3,4,5,6,7] for calculating impedance, off line. The need for a suitable algorithm is complicated by the nature of line voltages and currents during immediate power frequency post-fault cycles of these signals, in that the post-fault waveforms contain a power frequency fundamental component, a transient DC offset component, and transient high frequency components.

There are two approaches for calculating the series line impedance. One uses a differential approach and other a steady state approach. The basic difference lies in their implicit assumptions.

With the advent of economical microprocessors it is now possible to consider a microprocessor based digital distance protection system. The microprocessor based instrument has a number of advantages: It is cheap, small in size and consumes little power. But until very recently the microprocessor has been hampered by slow multiplication and division.

This report describes a system for digital calculation of on-line impedance of the line using a microprocessor and a fast multiplier. The approach taken here is to use a digital spectral analyser, which is

equivalent to an finite impulse response [FIR] filter, to extract the fundamental component of voltage (V) and current (I), and then the impedance (Z) is calculated by taking the ratio of V/I.

The spectral analyser used is of the sliding or running spectral measurement type. The system is built using an Intel SDK85 single board computer and a TRW TDC 1003J, a 12 bit x 12 bit, fast multiplier-accumulator. Both the microprocessor and the fast multiplier are used for digital filtering of input waveforms as well as for division.

Chapter II describes the different techniques used for digital impedance calculation, together with the method that was implemented. Chapter III describes the system hardware. Chapter IV details the digital filter and impedance algorithm and Chapter V discusses the test results, together with suggestions for further improvement.

## CHAPTER II

### DIFFERENT TECHNIQUES OF DIGITAL IMPEDANCE CALCULATION

#### 2.1 Introduction

The impact of modern development in digital technology has been felt in many disciplines of power engineering. In recent years, strong interest in applying digital methods to protective relaying has been indicated by the appearance of numerous technical articles on the subject.

The basic idea is to investigate the feasibility of utilizing digital techniques for the protection of transmission lines. The criterion for such a protection system is to equal or surpass the performance of existing relaying systems. Explicit points of interest are: operating speed, accuracy and dependability.

#### 2.2 Fundamental Considerations

For the protection of transmission lines, a well known parameter, to determine the character of a fault, is the impedance looking into the line. A small number of current and voltage samples are used to calculate the impedance.

For digital relaying, two possibilities exist for calculating the line impedance. One uses a differential approach, and the other a steady state approach.

### 2.3 Differential Equation Approach

For most applications, a power transmission line can be represented as a series resistive and inductive branch. From circuit theory the fundamental description of such a branch is the differential equation

$$V = Ri + L \frac{di}{dt} \quad 2.1$$

This relationship holds under short circuit conditions for both steady state and transient conditions and form the basis for the digital algorithm. It is necessary to note that, transmission lines deviate from the series R, L approximation due to the shunt capacitive effect. The inductances high frequency transients, but they are often short lived and die out fast. The effect can be minimized by filtering. With the differential equation approach a low pass filter is sufficient as opposed to the narrow-bandpass filter required for the impedance approach.

#### 2.3.1 Digital Algorithm For the Calculation of Circuit Parameters R,L

For a circuit, the parameter values of R and L are calculated using numerical methods. The samples are taken at a fixed rate at times say  $t_0$ ,  $t_1$  and  $t_2$ . Thus there are two sample periods, the first from  $t_0$  to  $t_1$  labelled as A, and the second from  $t_1$  to  $t_2$ , labelled as B. The average values of current and voltage during these periods are as follows:

$$i_A = \frac{i_0 + i_1}{2} ; \quad i_B = \frac{i_1 + i_2}{2} \quad 2.2$$

$$V_A = \frac{V_0 + V_1}{2} ; \quad V_B = \frac{V_1 + V_2}{2} \quad 2.3$$

taking the derivatives of Eq. 2.2 and Eq. 2.3,

$$\frac{di_A}{dt} = \frac{i_1 + i_0}{t_1 - t_0} \quad \text{and} \quad \frac{di_B}{dt} = \frac{i_2 + i_1}{t_2 - t_1}$$

The differential equations of the line for these two periods are

$$V_A = Ri_A + L \frac{di_A}{dt} \quad 2.4$$

$$V_B = Ri_B + L \frac{di_B}{dt} \quad 2.5$$

Thus by taking the samples  $i_0$ ,  $i_1$ ,  $i_2$  and  $V_0$ ,  $V_1$  and  $V_2$ , Eq. 2.4 and Eq. 2.5 can be solved for the two unknowns R and L. Eq. 2.4 and Eq. 2.5 can be programmed to determine R and L from the sampled values of current and voltage.

The characteristic for relay tripping is programmed as a generalized quadrilateral depending on number of zones to be protected, their distance and transmission line characteristic. A typical characteristic used by Breingan, Chen and Gallen [2] for a two-zone stepped distance scheme is shown in Fig. 2.1.

With three successive samples of current and voltage, the circuit parameters R and L are calculated and checked if they lie within the characteristic shown in Fig. 2.1. Normally if four values of R and L lie within the characteristic, a trip signal is sent to circuit breakers.

In this approach of solving the differential equations, the fundamental algorithm for the calculation of R and L along with the relaying requirements imposes numerous restrictions affecting the computer program. The calculation of R and L itself requires many multiplications and divisions which are time consuming operations. So in this type of approach the parameters are normally not calculated in steady state condition, and once fault is detected either by monitoring a voltage or current lines,

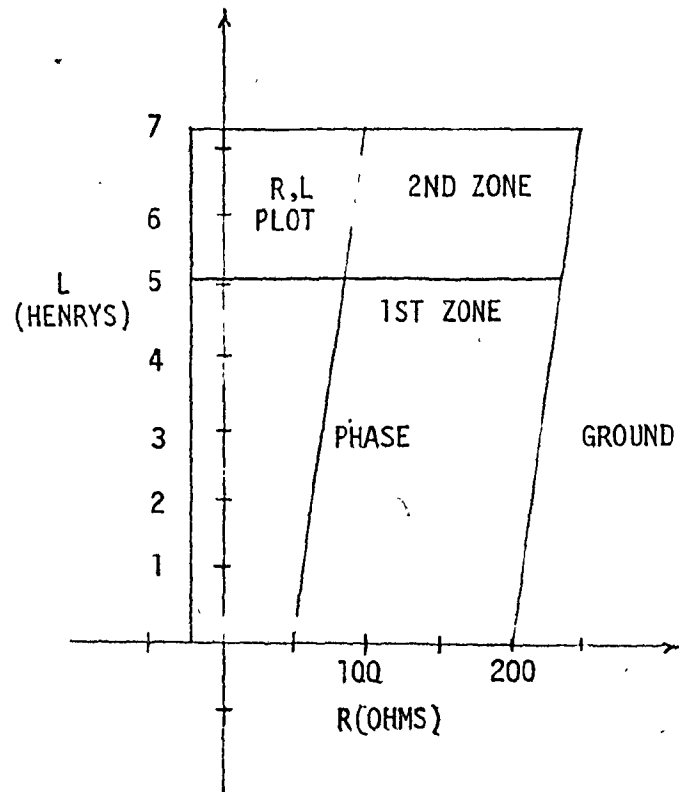


Fig. 2.1. Trip characteristics with typical R,L Plot.

the parameters R and L are determined to give a trip signal. Breingan, Chen and Gallen [2] implemented this approach on a minicomputer and achieved a tripping time of 6.12 to 7.4 msec.

#### 2.4 Steady State Approach

In this approach a line impedance is calculated from peak current and peak voltage. Different methods are suggested by different authors and we will consider each method briefly.

##### 2.4.1 Digital Impedance Calculation Using a Predictive Method

In the early stages of digital methods, Mann and Morrison [3] suggested calculation of the line impedance which involves predictive calculation of peak current and peak voltage, the impedance being determined by the division of peak voltage and peak current.

A digital computer, sampling a sinusoidal waveform, can determine the peak values as they occur. However if the peak values can be predicted before their occurrence, a faster fault-detection is achieved.

##### 2.4.2 Basic Theory of Predictive Calculation of Impedance

Consider a sinusoidal waveform, i.e.

$$v = V_{pk} \sin \omega t \quad 2.6$$

where  $V_{pk}$  is the unknown quantity and  $v$  is a typical sample value. If the sampling is not synchronized then  $\sin \omega t$  is also unknown.

By differentiating Eq. 2.6 we get



$$v' = \omega V_{pk} \cos \omega t \quad 2.7$$

If  $v'$  can be determined, then the peak value of the sinusoid is determined from Eq. 2.6 and 2.7 as

$$V_{pk}^2 = v^2 + \left[\frac{v'}{\omega}\right]^2 \quad 2.8$$

using Eq. 2.6 to Eq. 2.8 for the transmission line currents and voltages, the modulus of line impedance can be determined, given by

$$|Z| = \left[ \frac{V_{pk}^2}{I_{pk}^2} \right]^{1/2} \quad 2.9$$

where  $I_{pk}$  is a peak value of current given by

$$I_{pk}^2 = i^2 + \left[\frac{i'}{\omega}\right]^2$$

where  $i$  is the current sample value and  $i'$ , derivative of it. Furthermore the phase difference between the voltage and current waveforms can be determined from,

$$\phi = \arctan[(\omega i)/i'] - \arctan[(\omega v)/v'] \quad 2.10$$

enabling complete impedance determination.

This method has serious drawbacks: It assumes that the waveforms are pure sinusoids before and after the fault. It does not take into account the transients (including the exponentially decaying DC transients) on current and voltage waveforms. So some kind of analog filter has to be used before processing the voltage and current samples.

The method also requires the derivatives of current and voltage samples which are calculated numerically using digital algorithms, which restrict the accuracy of calculations. Mann and Morrison used three successive samples to calculate the derivatives giving an accuracy of about

+10% in impedance value.

### 2.4.3 Digital Impedance Relaying Using Fourier Analysis

Ramamoorthy [4] was the first to propose the use of Fourier analysis rather than the predictive method of Mann and Morrison. The ensemble of samples over a period of one cycle is assumed to repeat periodically and a Fourier analysis is performed on the ensemble of samples. The amplitude and phase angle of the fundamental component are obtained as follows

$$a_1 = \frac{2}{m} \left( \frac{f_0}{2} + f_1 \cos x + f_2 \cos 2x \dots + f_{m-1} \cos(m-1)x + \frac{f_m}{2} \right) \quad 2.11$$

$$b_1 = \frac{2}{m} (f_1 \sin x + f_2 \sin 2x \dots f_{m-1} \sin(m-1)x) \quad 2.12$$

where  $f_0, f_1, f_2, \dots, f_m$  are sampled values of input signal over a period of one cycle and  $x = 2\pi/m$  is the sampling interval. The factors  $\cos x, \sin x, \dots, \cos mx, \sin(m-1)x$  are constants and can be calculated and stored in the computer a priori, as weighting functions on the sample values. The fundamental quantity  $g(t)$  is given by

$$g(t) = \sqrt{a_1^2 + b_1^2} \sin[\omega t + \arctan(b_1/a_1)] \quad 2.13$$

If this computation is made for both voltage and current, the magnitude of impedance and phase angle can be calculated.

Ramamoorthy claims better results than Mann and Morrison's method for faults on a model line in the laboratory. Sampling the waveform at 20 points per cycle he calculates the impedance on an IBM/7 system in 4.6 msec. Since the sampling interval is 0.8 msec for 20 points per cycle and the impedance should be calculated within this interval, this method cannot be used for real time operation. Most of the time is consumed in

multiplication, square-rooting and division in order to get impedance  $|Z|$

as

$$|Z| = \frac{(\sqrt{a_1^2 + b_1^2}) \text{ voltage}}{(\sqrt{a_1^2 + b_1^2}) \text{ current}}$$

#### 2.4.4 Digital Impedance Relaying Using Walsh Functions

J. W. Horton (5) suggested the use of Walsh functions to extract the fundamental 60 Hz component of current and voltages. His method does not assume any sinusoidal conditions, before and after the fault.

The time consuming multiplications and even the squaring and square-rooting operations, in the Fourier analysis algorithm used by Ramamoorthy can be almost entirely eliminated if the function  $g(t)$  is analysed into its Walsh functions  $Wal(k,t)$ . This is possible because  $Wal(k,t)$  has only two values,  $+1$  and  $-1$  and so Walsh analysis can be performed by the operations of additions and subtractions.

The first 16 of these Walsh functions are shown in fig. 2.2. They resemble "squaring-up" sine and cosine functions and form a complete orthonormal set. Walsh functions are undefined at the points at which they change from  $+1$  to  $-1$ , but as these points are a set of measure zero, this is of no consequence.

Let  $t' = \frac{t}{T}$  and let the Fourier expansion of  $g(t)$  in the interval  $(0,T)$  be defined as:

$$g(t) = F_0 + \sqrt{2} F_1 \sin \frac{2\pi t}{T} + \sqrt{2} F_2 \cos \frac{2\pi t}{T}$$

and the Walsh function is defined as

$$g(t) = \sum_{k=0}^{\infty} W_k Wal(k,t/T)$$

Wal(0,t')	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	
Wal(1,t')	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	
Wal(2,t')	+	+	+	+	-	-	-	-	-	-	-	+	+	+	+	
Wal(3,t')	+	+	+	+	-	-	-	-	+	+	+	+	-	-	-	
Wal(4,t')	+	+	-	-	-	-	+	+	+	+	-	-	-	-	+	+
Wal(5,t')	+	+	-	-	-	-	+	+	-	-	+	+	+	+	-	-
Wal(6,t')	+	+	-	-	+	+	-	-	-	-	+	+	-	-	+	+
Wal(7,t')	+	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-
Wal(8,t')	+	-	-	+	+	-	-	+	+	-	-	+	+	-	-	+
Wal(9,t')	+	-	-	+	+	-	-	+	-	+	+	-	-	+	+	-
Wal(10,t')	+	-	-	+	-	+	+	-	-	+	+	-	+	-	-	+
Wal(11,t')	+	-	-	+	-	+	+	-	+	-	-	+	-	+	+	-
Wal(12,t')	+	-	+	-	-	+	-	+	+	-	+	-	-	+	-	+
Wal(13,t')	+	-	+	-	-	+	-	+	-	+	-	+	+	-	+	-
Wal(14,t')	+	-	+	-	+	-	+	-	-	+	-	+	-	+	-	+
Wal(15,t')	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-

Fig. 2.2. The first sixteen Walsh function of integral index  $k$ .

then,

$$F_0 = \frac{1}{T} \int_0^T g(t) dt \quad 2.14$$

$$F_1 = \frac{\sqrt{2}}{T} \int_0^T g(t) \sin \frac{2\pi t}{T} dt \quad 2.15$$

$$F_2 = \frac{\sqrt{2}}{T} \int_0^T g(t) \cos \frac{2\pi t}{T} dt \quad 2.16$$

and

$$W_k = \frac{1}{T} \int_0^T g(t) \text{Wal}(k, t') dt \quad 2.17$$

The set of components  $F_k$  form a vector in Hilbert space and so does the set  $W_k$ . The two vectors are related by the orthogonal matrix  $A$ . Thus,

$$W = AF$$

Since

$$A^{-1} = A'$$

where  $A'$  is the transpose of  $A$ , we also have

$$F = A'W \quad 2.18$$

The matrix  $A$ , in part is given by,

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & .90 & 0 & 0 & 0 & .30 & 0 & 0 & 0 \\ 0 & 0 & .90 & 0 & 0 & 0 & -.300 & 0 & 0 \\ 0 & 0 & 0 & .90 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .90 & 0 & 0 & 0 & 0 \\ 0 & -.373 & 0 & 0 & 0 & .724 & 0 & 0 & 0 \\ 0 & 0 & .373 & 0 & 0 & 0 & .724 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.90 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.90 \end{bmatrix}$$

9x9

Now let  $g(t)$  be a given function. Then using Walsh functions, we first find Walsh coefficients  $W_k$  from Eq. 2.17. In actual computation the waveform is sampled  $(m+1)$  times in interval  $(0,t)$ . From Eq. 2.17 we must perform the numerical integration which closely approximate the integral. Once Walsh coefficients  $W_k$  are known, then using Eq. 2.18, the Fourier coefficients  $F_1$  and  $F_2$  are calculated. The amplitude of a sinusoid is given by  $\sqrt{F_1^2 + F_2^2}$ . Thus finding out the Fourier coefficients  $F_1$  and  $F_2$  for both voltage and current waveforms, the impedance can be calculated. The impedance calculation requires four multiplications, one division and one square rooting. To avoid multiplications, Horton has suggested an amplitude and phase theorem [6] which gives the empirical formula for impedance  $|Z|$  as

$$|Z| = \frac{\{1.0822(|W_1| + |W_2|) + 0.414 (|W_2| - |W_1|)\} \text{voltage}}{\{\text{same}\} \text{current}} \quad 2.19$$

and he has calculated impedance using Eq. 2.19 for 17 samples per cycle in 1 msec, with an accuracy of  $\pm 8\%$ .

Most of the digital impedance algorithms cited before, were developed prior to the advent of microprocessors and envisaged the use of dedicated minicomputers or larger computers for the implementation of digital distance protection of transmission lines. These implementations generally include programmed logic to first determine the most probable type of fault that exists and then calculate the apparent impedance of the faulty phase or phases off-line. This was done in order to avoid large amount of calculations.

It is now possible to consider a microprocessor based system, which would retain the essential parallelism of existing electromechanical

and solid state, phase and ground distance relays. The main problem in using microprocessors is slow multiplication and division. Advances in integrated circuit technology have provided fast array multipliers.

Using a fast multiplier and other digital hardware, any of the earlier-mentioned steady-state techniques can be used for impedance calculation. One can use Walsh analysis or Fourier analysis to get Fourier coefficients  $F_1$  and  $F_2$  and then a fast multiplier together with additional digital circuit can be used to implement the impedance calculation  $|Z|$ .

#### 2.4.4 Digital Impedance Calculation Using Digital Spectral Analyser

The method of the present work is to use a digital spectral analyser to extract the fundamental component of voltage and current. The basic principle of digital filtering is explained in the following sections.

#### 2.4.5 Digital Filtering of Fundamental Component [7]

Spectral analysis can be considered as a problem of evaluating the z transform of a modified version of signal over a region of the z plane.

##### z transform of a given signal

Given a sequence  $X(n)$  defined for all  $n$ , its z transform is defined as,

$$X(z) = \sum_{n=-\infty}^{\infty} X(n) z^{-n} \quad 2.20$$

where  $z$  is a complex variable.

The  $z$  transform of a sequence may be viewed as a unique representation of that sequence in the complex  $z$  plane. Eq. 2.20 indicates that if the  $z$  transform is evaluated on a circle of unit radius, i.e.  $z = e^{j\omega}$  then

$$X(z) \Big|_{z=e^{j\omega}} = X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} X(n) e^{-j\omega n}$$

which is the Fourier transform of the sequence  $X(n)$ .

### The Discrete Fourier Transform

Consider a case where the sequence to be represented is periodic; then the sequence can be represented in a discrete Fourier series. For a periodic sequence  $X_p(n)$  with a period of  $N$  samples,

$$X_p(n) = \sum_{k=-\infty}^{\infty} X_p(k) e^{-j2\pi nk/N} \quad 2.21$$

where the only possible frequencies of which  $X_p(n)$  can be composed of are

$$\omega_k = \frac{2\pi k}{N} \quad ; \quad -\infty < k < \infty$$

Since  $\omega_k$  are the only frequencies whose periods are integrally related to  $N$ . The quantity  $X_p(k)$  in Eq. 2.21 represents the amplitude of the sinusoid at frequency  $\omega_k$ . Because of the periodicity of the function:

$$e^{j2\pi nk/N} = e^{j2\pi n(k \pm mN)/N} \quad ; \quad 0 < m < \infty$$

so Eq. 2.21 reduces to

$$X_p(n) = \sum_{k=0}^{N-1} X_p(k) e^{j2\pi nk/N}$$

In a more familiar representation,  $X_p(n)$  is expressed as,



$$X_p(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_p(k) e^{j2\pi nk/N} \quad 2.22$$

Taking the inverse transform

$$X_p(k) = \sum_{n=0}^{N-1} X_p(n) e^{-j2\pi nk/N} \quad 2.23$$

Eq. 2.23 is called the Discrete Fourier transform and  $X_p(k)$  represents the amplitude of the sinusoid at frequency  $\omega_k$ .

Eq. 2.22 and Eq. 2.23, show that both the sequences  $X_p(n)$  and  $X_p(k)$  are periodic with period of  $N$  samples and  $X_p(k)$  may be determined exactly from one period of  $X_p(n)$ .

#### Relation between z transform and DFT

Let  $X(n)$  be a finite duration sequence defined as

$$X(n) = \begin{array}{ll} X_p(n) & ; \quad 0 \leq n \leq N-1 \\ 0 & ; \quad \text{all other } n \end{array}$$

where  $X_p(n)$  is periodic with a period of  $N$  samples. The z transform of  $X(n)$  is

$$X(z) = \sum_{n=0}^{N-1} X(n) z^{-n}$$

Evaluation of  $X(z)$  at point  $z = e^{j2\pi k/N}$  i.e. at a point on the unit circle with angle  $\frac{2\pi k}{N}$  gives

$$\begin{aligned} X(z) \Big|_{z=e^{j2\pi k/N}} &= X[e^{j2\pi k/N}] \\ &= \sum_{n=0}^{N-1} X(n) e^{-j2\pi kn/N} \end{aligned} \quad 2.24)$$

Since

$$X_p(n) = X(n) \quad ; \quad 0 \leq n \leq N-1$$

Eq. 2.23 and Eq. 2.24 give

$$X_p(k) = X[e^{j2\pi k/N}]$$

Thus the DFT coefficients of a finite duration sequence are the values of the z transform of that same sequence at N evenly-spaced points around the unit circle.

### 2.5 Sliding or Running Spectral Measurement

To find the amplitude  $X_p(k)$  of fundamental frequency  $\omega_1$  from eq. 2.23, we see that

$$X_p(1) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi n/N} \quad 2.25$$

One complete period is required to obtain the amplitude. For a sliding type of measurement, each time a current sample  $n$  and its previous  $N$  samples are used for the computation. Thus at any point  $n$ , the spectrum is a function of  $(n-l)$  samples where  $0 \leq l \leq N-1$  and  $N$  is number of samples in one period. The spectrum for the fundamental component  $s_n(z_1)$  can be written as

$$S_n(z_1) = \sum_{l=0}^{N-1} x(n-l) e^{-j2\pi l/N} \quad 2.26$$

Let  $n-l = m$ , then Eq. 2.26 becomes

$$S_n(z_1) = \sum_{m=n-N+1}^{n-1} x(m) e^{-j2\pi(n-m)/N}$$

or

$$S_n(z_1) = \sum_{m=n-N+1}^n x(m) z_1^{-(n-m)} \quad 2.27$$

eq. 2.27 can be expanded into

$$S_n(z_1) = X(n) + X(n-1)z_1^{-1} + (n-2)z_1^{-2} + \dots X(n-N+1)z_1^{-(N-1)} \quad 2.28$$

To measure  $S_n(z_1)$  for successive values of  $n$ , i.e.  $S_0(z_1)$ ,  $S_1(z_1)$ ... etc, a window, of duration  $N$  samples, is moved one sample ahead and the measurement is repeated. This type of measurement is called a sliding or running spectral measurement. Eq. 2.28 shows that a sliding spectral measurement at a single value of  $z$  i.e. at  $z = z_1$  is equivalent to an finite impulse response filter [FIR]. Eq. 2.28 can be written as,

$$S_n(z_1) = h(0)X(n) + h(1)X(n-1) + \dots h(N-1)X(n-N+1) \quad 2.29$$

where  $h(n)$  is the impulse response of a filter given by

$$h(n) = z_1^{-n} \quad ; \quad 0 \leq n \leq N-1 \quad 2.30$$

Fig. 2.3 shows the direct convolution realization of the spectral measurement of Eq. 2.27.

Considering two successive spectral measurements  $S_{n-1}(z_1)$  and  $S_n(z_1)$ , a recursive relation is obtained of the form

$$S_n(z_1) = z_1^{-1} S_{n-1}(z_1) + X(n) - z_1^{-N} X(n-N) \quad 2.31$$

Eq. 2.31 is implemented as shown in Fig. 2.4.  $z_1^{-N}$  and  $z_1^{-1}$  in Eq. 2.31 represents the delay of  $N$  samples and one sample respectively and  $z_1$ 's are complex coefficient multipliers.

### Extraction of fundamental component

To extract the fundamental frequency from a periodic waveform Eq. 2.31 is used in which  $z_1$  is given by

$$z_1 = e^{j2\pi/N}$$

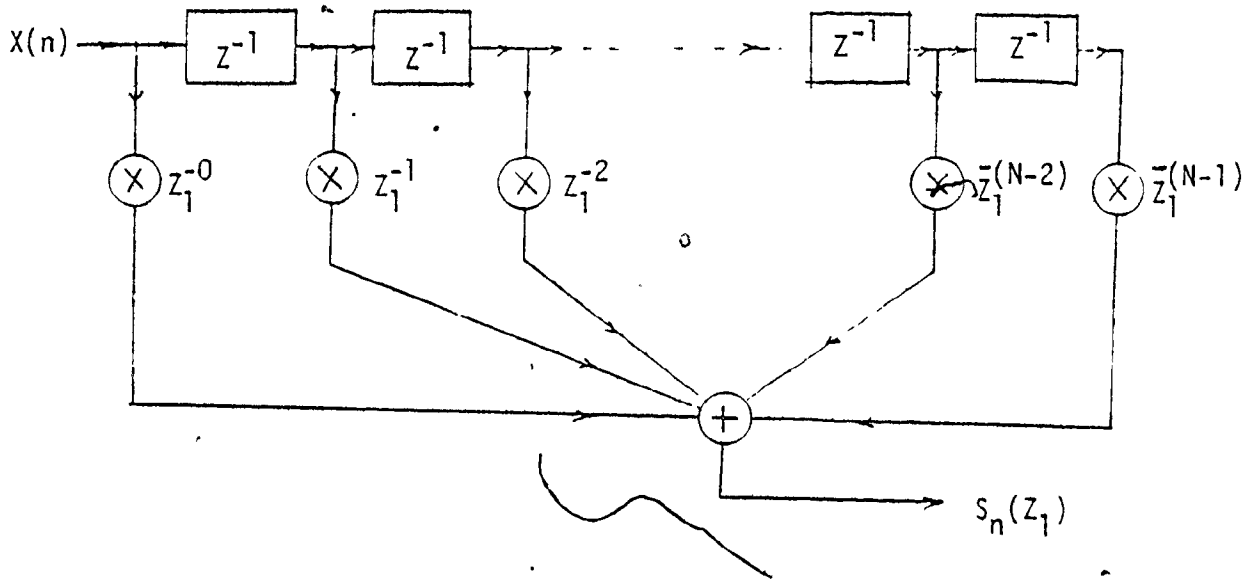


Fig. 2.3. FIR implementation of spectrum analysis.

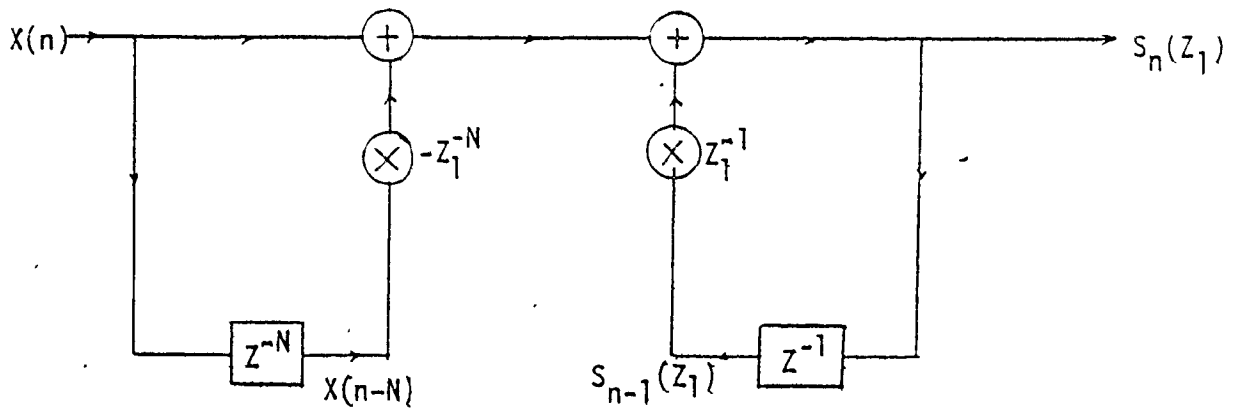


Fig. 2.4. Recursive implementation of spectrum analysis.

For sixteen samples in one period ( $N = 16$ ),

$$\begin{aligned} z_1 &= e^{j2\pi/16} \\ &= a + jb \end{aligned}$$

where 'a' and 'b' are constants given by

$$\begin{aligned} a &= \cos 22.5^\circ = 0.92388 \quad \text{and} \\ b &= \sin 22.5^\circ = 0.38268 \end{aligned}$$

So Eq. 2.31 can be written as

$$S_n(z_1) = (a - jb)S_{n-1}(z_1) + X(n) - X(n-N)$$

since

$$z_1^{-1} = a - jb$$

and

$$z_1^{-N} = e^{-j(2\pi/N) \cdot N} = 1$$

$S_{n-1}$  is a filtered output of previous sample and is a complex quantity.

Let

$$S_{n-1} = P_{n-1} + jQ_{n-1}$$

Eq. 2.31 becomes

$$\begin{aligned} S_n(z_1) &= (P_{n-1} + jQ_{n-1})(a - jb) + X(n) - X(n-N) \\ &= a P_{n-1} + bQ_{n-1} + X(n) - X(n-N) + j[aQ_{n-1} - bP_{n-1}] \\ &= P_n + jQ_n \end{aligned} \tag{2.32}$$

where

$$\begin{aligned} P_n &= aP_{n-1} + bQ_{n-1} + X(n) - X(n-N) \quad \text{and} \\ Q_n &= aQ_{n-1} - bP_{n-1} \end{aligned}$$

The magnitude of  $S_n$  is obtained from its real and imaginary part by

$$|S_n| = \sqrt{P_n^2 + Q_n^2}$$

Thus to extract a fundamental component, Eq. 2.32 is implemented using digital hardware.  $X(n-N)$  is a delayed sample and is obtained by using a microprocessor in which  $N$  successive samples are stored and at each discrete time the current sample value  $X(n)$  and the delayed sample value  $X(n-N)$  are taken out from the memory. A fast multiplier TDC 1003J is used for multiplication. The system block diagram and circuit details are discussed in Chapter III.

## CHAPTER III

### SYSTEM HARDWARE

A block diagram for the impedance calculation of a transmission line is shown in Fig. 3.1.

#### 3.1 Data Acquisition

Voltage and current waveforms on a transmission line are fed to a preprocessor, which scales the signals to a range suitable for the analog to digital converter used. Preprocessed voltage and current waveforms are multiplexed in an analog multiplexer and then sampled and digitized in an analog to digital converter. The digitized samples of voltage and current are then stored into the read/write memory of the microcomputer, for further analysis.

##### 3.1.1 Analog Multiplexing

Since the A/D converter is an expensive component, multiplexing analog inputs to A/D is an economical approach. The analog multiplexer time-shares an A/D converter among two analog channels, one for voltage and one for current. The multiplexer used is DG172BK, the pin diagram for which is shown in Fig. 3.2.

Preprocessed voltage and current waveforms are fed to the two inputs of the analog multiplexer. Channel select signals are derived from the sampling pulses from a circuit shown in Fig. 3.3. Sixteen samples are

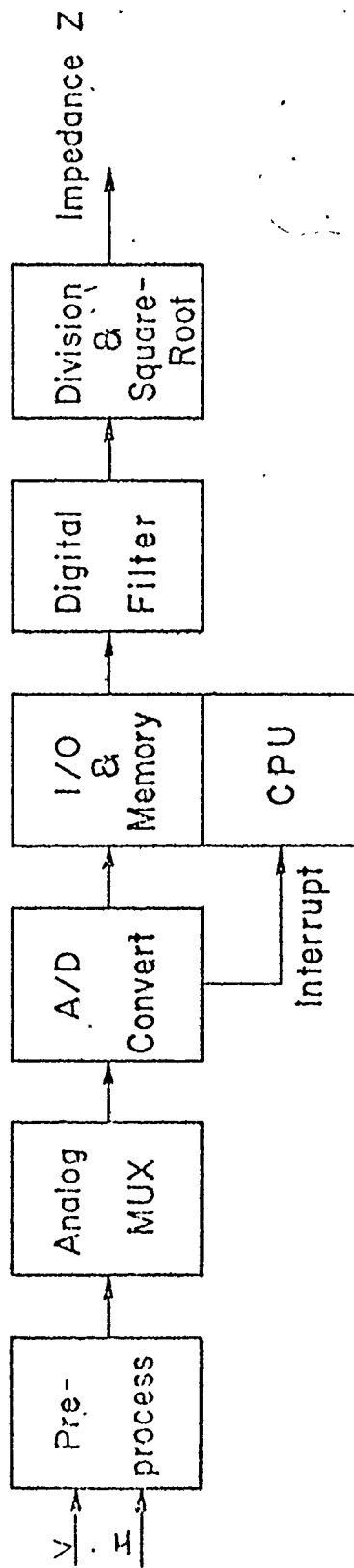
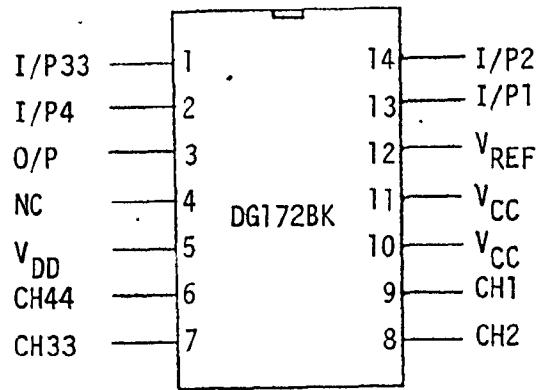


Fig.3-1. System Block Diagram





Pin No.	Destination	Description of Function
1,2,13,14	I/P3,I/P4,I/P1,I/P2	Analog inputs
3	O/P	Multiplexed output
4	NC	Not connected
5	$V_{DD}$	Negative supply (-15V)
6,7,8,9	CH4,CH3,CH2,CH1	Channel select signals
10,11	$V_{CC}$	Positive supply (+5V)
12	$V_{REF}$	Reference voltage (+10V)

Fig. 3.2. Analog multiplexer.

taken in one cycle. So the sampling period  $T_s$  is 1.042 msec, for a power line frequency of 60 Hz. The pulse duration  $T_1$  of the first monostable M1 in Fig. 3.4 is decided by the time taken for A/D conversion and for storing the digitized sample into the microcomputer memory. The pulse duration of the second monostable M2 is 5  $\mu$ sec. The output of M2 and the sampling pulses are fed to an OR gate, the output of which is used as a convert command for the A/D conversion.

$Q_1$  and  $\overline{Q_1}$  outputs of M1 are used as channel select signals for the analog multiplexer such that during time  $T_1$ , the voltage waveform is present and during  $T_2$ , the current waveform is present on the multiplexed output of the analog multiplexer.

### 3.1.2 Analog to Digital Conversion

The selection of the A/D converter is based on accuracy, speed and cost. For calculating on-line impedance, it is essential to have a fast A/D conversion. The converter used is a Burr-Brown ADC85KG. This is a 12 bit converter with a conversion time of 10  $\mu$ sec. CTC (Complement Two's Complement) logic is used. The pin connections and specifications are given in Appendix A.

The multiplexed output from the analog multiplexer is fed to the analog input of the A/D converter. The convert command is derived from the sampling pulses as shown in Fig. 3.3. The A/D converter gives a status word output which is "HIGH", during conversion and goes "LOW" when the conversion is complete. Fig. 3.5 shows the A/D converter waveforms.  $T_c$  is the conversion time.

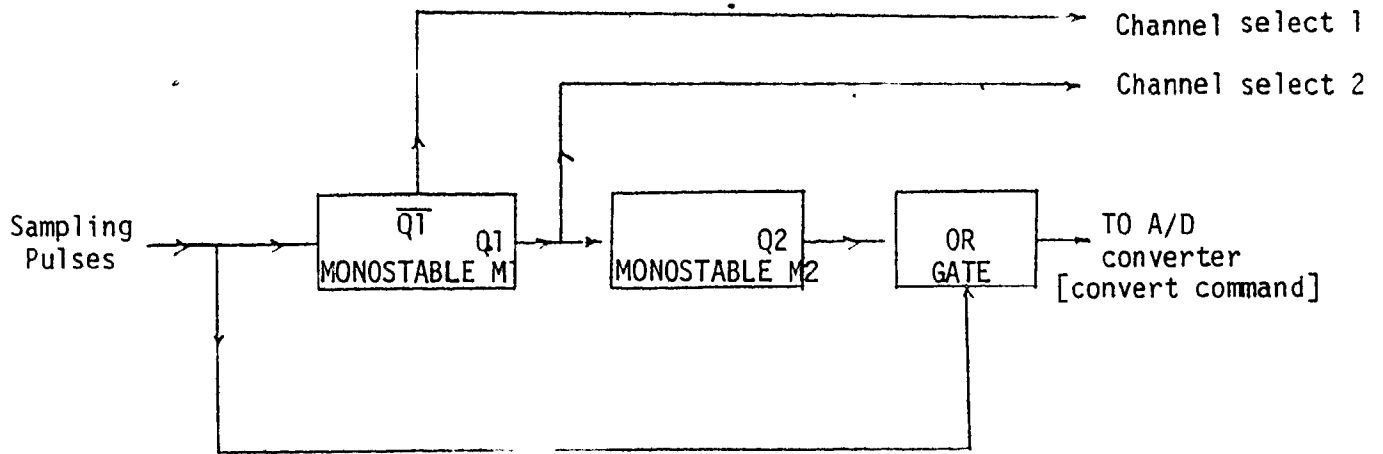


Fig. 3.3. Block diagram for channel select .

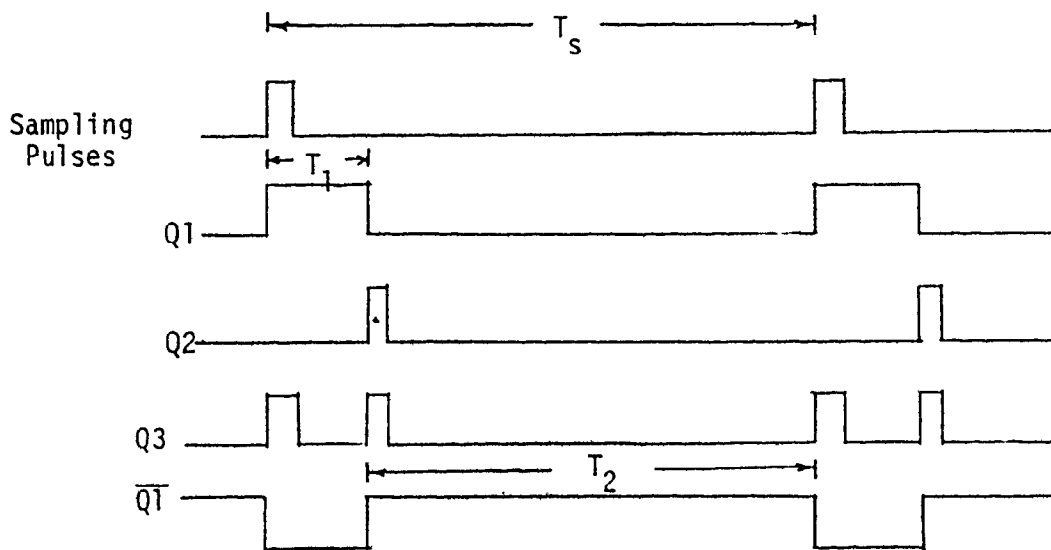


Fig. 3.4. Waveforms for channel select.

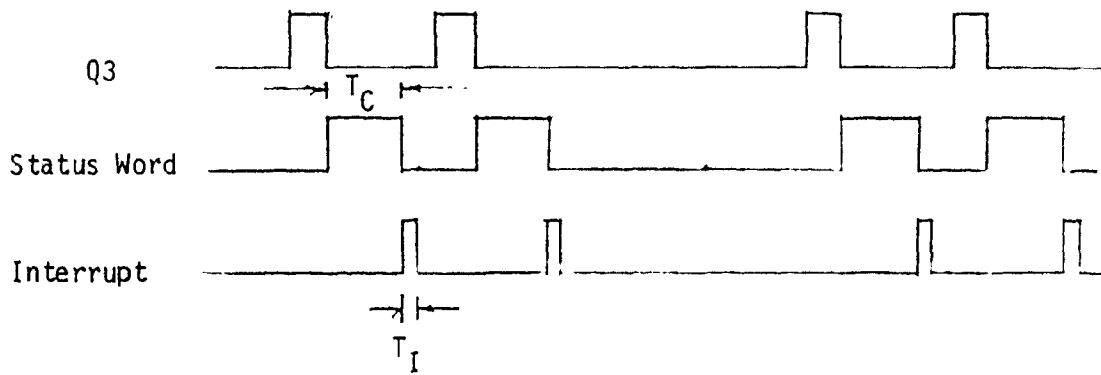


Fig. 3.5. Waveforms for A/D converter.

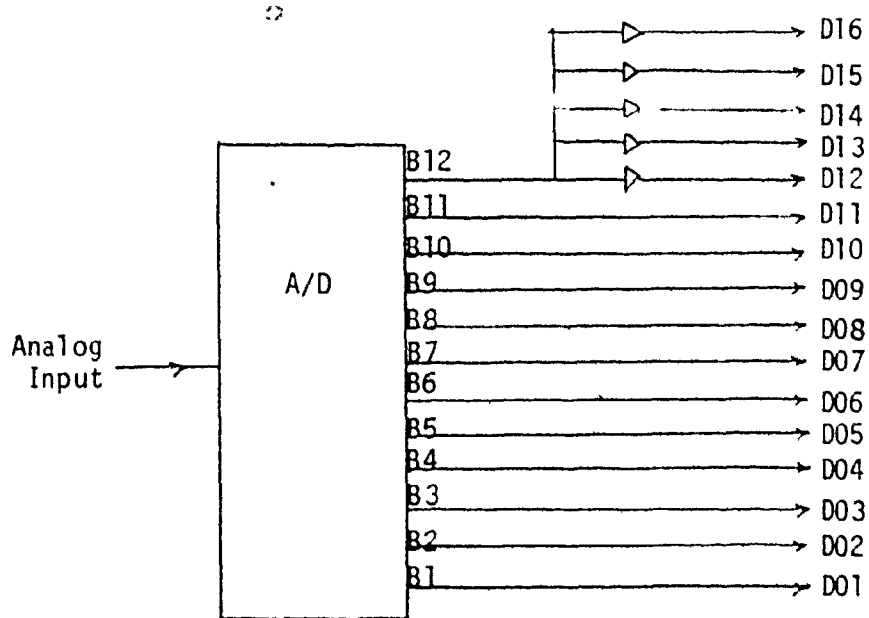


Fig. 3.6. 12-bit to 16-bit conversion.

When the conversion is complete a pulse of duration  $T_f$  is derived. This is used to interrupt the microprocessor to indicate that a digitized sample is ready to be stored in the memory and initiates the data-storage routine.

### 3.1.3 Storing the Data Into Memory

The Intel 8085 is a 8 bit microprocessor and the 12 bit digitized sample must be stored in two successive words. A 12 bit to 16 bit conversion is done using "AND" gates and is shown in Fig. 3.6. This 16 bit word is then stored in memory using a software program called data-storage subroutine stored in the ROM of microcomputer. The subroutine is discussed in detail in Chapter IV

## 3.2 Selection of the Microcomputer

The microcomputer system used is an Intel SDK85. One of the major factors in selecting this system is its relatively faster speed, availability and on-board peripheral devices which include 0.5K RAM, 2K ROM, I/O ports, keyboard and display. There is also an area for expansion. Because of the keyboard and the display, together with monitor routines in ROM, it is easy to edit and debug programs. Two interrupts are available for the user. The 8085 has an instruction cycle time of 1.3  $\mu$ sec. The instruction set is very broad and includes some 16 bit instructions.

The schematic diagram for the SDK85 system is given in Fig. 3.7. The instruction set for the 8085 microprocessor is given in Appendix B.

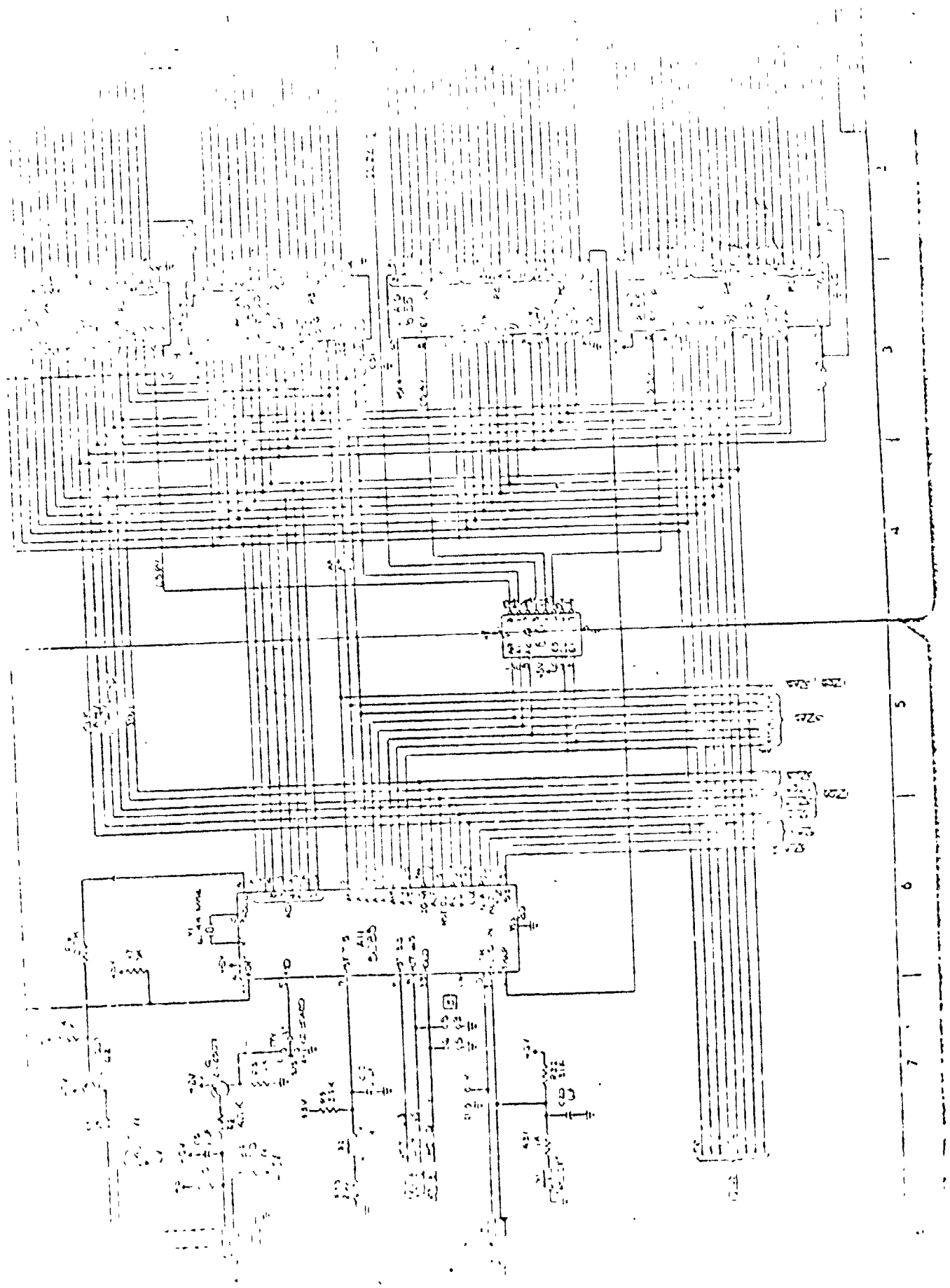



Fig. 3.7. Schematic diagram of SDK85 single board microcomputer.

### 3.3 Digital Filtering

The digital filter to be used is described in section 2.5. This is a time based design and its implementation requires several components including multipliers, adders and registers. These registers are used to hold the present and past values of samples fed to the filter and the previous output of the filter itself. Even though the cost of adders and registers had dropped rapidly during past few years, the multiplier is still considered to be an expensive component. Therefore in any kind of direct implementation, with cost as one of the significant factors, the objective is to keep the number of expensive components to a minimum, and a design with a single multiplier is attractive.

### 3.4 Choice of the Multiplier

Since the digitized voltage and current samples are 12 bit with dual polarity, a 12 bit x 12 bit fast multiplier is required capable of 2's complement multiplication. For the digital filter implementation of Eq. 2.31, many additions and subtractions are required during filter operation and it is advantageous to have an accumulator included in the multiplier.

TRW TDC1003J, a 12 bit x 12 bit multiplier-accumulator, with a typical multiply and accumulate time of 175 nsec is used. A data sheet for the device is given in Appendix C. 

### 3.5 Hardware Implementation of the Digital Filter

The filter equation to be implemented is discussed in Section 2.5, and is given by

$$S_n = (P_{n-1} + jQ_{n-1})(a - jb) + X(n) - X(n-N) \quad 3.1$$

where  $P_{n-1}$  is the real part of the previous filtered output  $S_{n-1}$

$Q_{n-1}$  is the imaginary part of the previous filtered output  $S_{n-1}$

$$a = \cos(2\pi/N) = 0.92388$$

$$b = \sin(2\pi/N) = 0.38268$$

$X(n)$  is a current sample and  $X(n-N)$  is the  $N^{\text{th}}$  previous sample.

Sixteen samples are stored in designated memory locations of the microprocessor. After receiving voltage and current sample values for each sampling pulse,  $[X(n) - X(n-N)]$  for both voltage and current are calculated and sent to output ports. Separating the real and imaginary parts of Eq. 3.1 we get

$$\begin{aligned} S_n &= aP_{n-1} + bQ_{n-1} + X(n) - X(n-N) + j[aQ_{n-1} - bP_{n-1}] \\ &= P_n + jQ_n \end{aligned} \quad 3.2$$

where  $P_n = aP_{n-1} + bQ_{n-1} + X(n) - X(n-N)$  and

$$Q_n = aQ_{n-1} - bP_{n-1}$$

Implementation of Eq. 3.2 is achieved using the circuit shown in Fig.

3.8. M1 and M2 are two 4:1 multiplexers [SN74157], to multiplex various inputs required for the sequential multiplications. The output registers R1 through R10 are used to store previous values  $(P_{n-1}, Q_{n-1})$ , and the



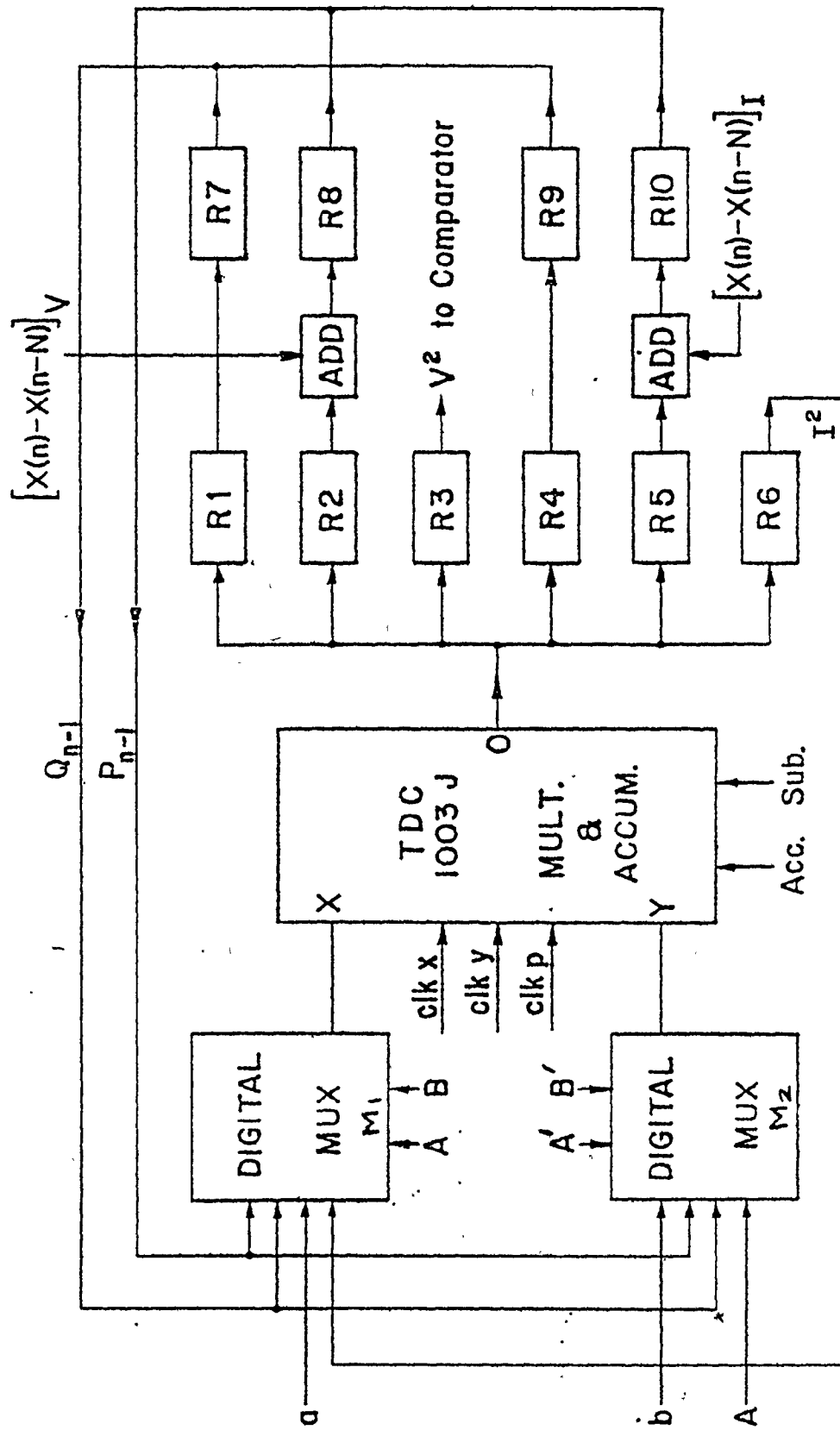


Fig. 3.8. Filter block diagram.

present values ( $P_n, Q_n$ ) of real and imaginary part of the filtered output of voltage and current, and to store filtered  $V^2$  and  $I^2$  quantities.

$R_1$  through  $R_6$  are Hex D type shift registers with clear [74LS174] and  $R_7$  through  $R_{10}$  are tristate D type shift registers [74173]. The tristate registers are used so that the same output bus can be used for voltage and current filtering. While voltage samples are filtered, the output registers for current [ $R_9, R_{10}$ ] are disabled offering a high impedance to the bus. Similarly when current samples are filtered, the voltage output registers [ $R_7, R_8$ ] are disabled.

The multiplier requires certain clock signals for its operation.  $ClkX$  and  $ClkY$  are required to take the data present at inputs  $X$  and  $Y$ .  $ClkP$  clocks out the multiplication on a bus.  $ACC$  and  $SUB$  are two clocks, which decide whether addition or subtraction is to be done.

Prior to multiplication, valid data has to be present at the inputs  $X$  and  $Y$ , before  $ClkX$  and  $ClkY$  occur. Data at the input of the multiplier is selected by selecting appropriate control signals  $A, B, A'$  and  $B'$  for multiplexers  $M_1$  and  $M_2$  respectively. Table 3.1 gives the truth table for the multiplexers  $M_1$  and  $M_2$ .

All of these control clocks for the multiplier as well as for the multiplexers are derived from the microprocessor. Two output ports [port 29 and port 2A] of 8 bits each are used as control ports, and are shown in Fig. 3.9. Clock pulses are sent out in required sequence using OUTPUT [PORT] instructions.

The output of the multiplier is a bus which is connected to the registers  $R_1$  to  $R_6$ . The multiplier output is stored in one register at a time, depending on which one of them is clocked. These clock pulses

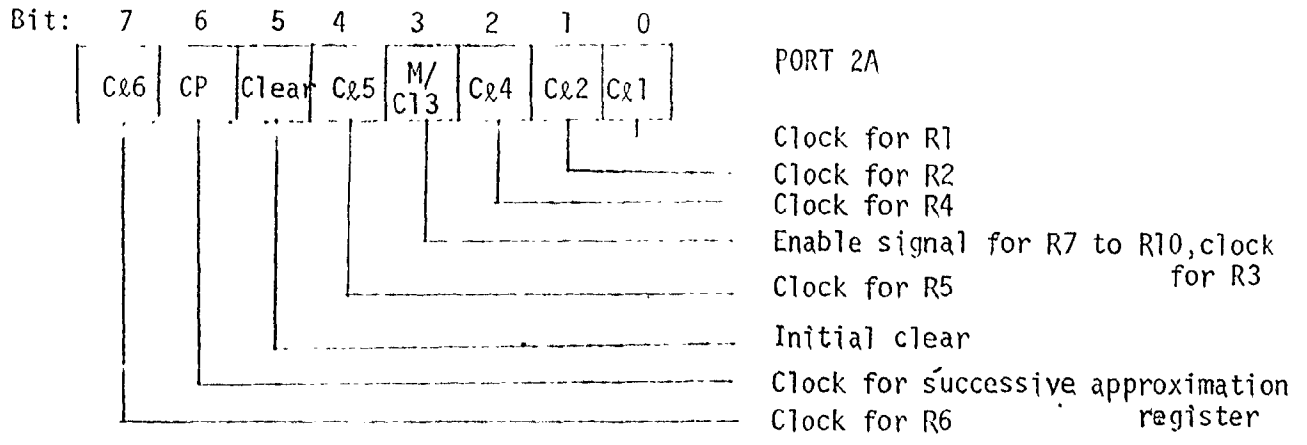
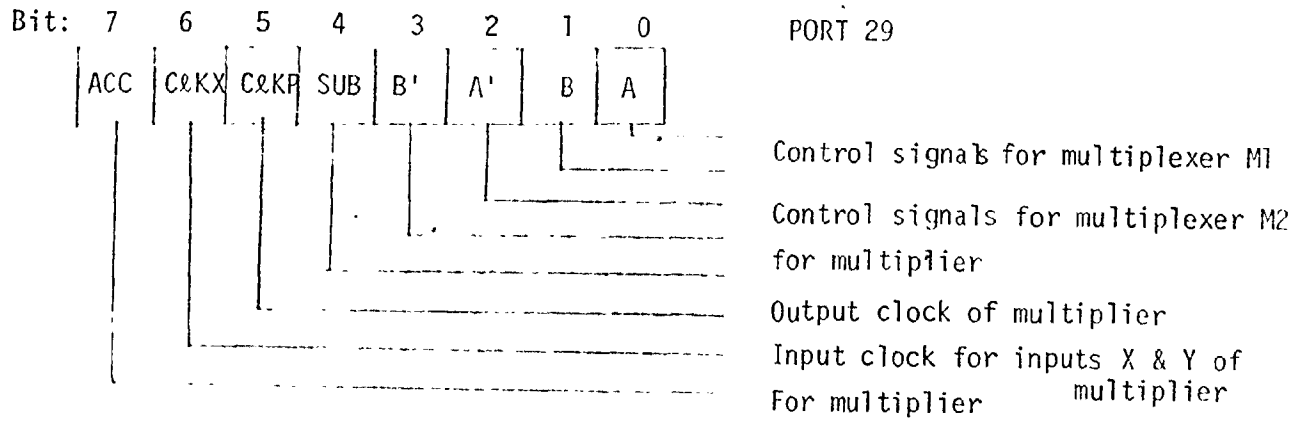


Fig. 3.9. Control ports.

Table 3.1 Control Signals for Multiplexers

B	A	output	B	A	output
0	0	$Q_{n-1}$	0	0	b
0	1	a	0	1	$P_{n-1}$
1	0	$P_{n-1}$	1	0	$Q_{n-1}$
1	1	$I^2$	1	1	A

$Cl_7$  to  $Cl_6$  are also microprocessor controlled. Control port 2A is used for these clocks as shown in Fig. 3.9. Bit 5 of port 2A is a "CLR" bit which clears all the output registers from  $R_7$  to  $R_{10}$ .

Bit 6 of port 2A is a clock for successive approximation register used for division and is discussed in Section 3.7.

Bit 3, which is  $Cl_3$  is also used as an enable signal M for tristate registers  $R_7$  through  $R_{10}$ .

### 3.6 Sequence of the Filter Operation

1. Initially output registers  $R_7$  through  $R_{10}$  are cleared [which makes  $S_0 = 0 + j0$ ].

Filter Operation	Derived Expression
2. Enable voltage registers. Disable current registers.	
Input 'b', ' $P_{n-1}$ ' to multiplier. Multiply and store in the accumulator.	$b \cdot P_{n-1}$
Input 'a', ' $Q_{n-1}$ ' to multiplier. Multiply, subtract the previous product from it and output to the register $R_1$ .	$a \cdot Q_{n-1}$ $aQ_{n-1} - bP_{n-1}$
3. Input 'a', ' $P_{n-1}$ ' to multiplier. Multiply and store in the accumulator.	$a \cdot P_{n-1}$
Input 'b', ' $Q_{n-1}$ ' to multiplier. Multiply. Add previous product to it and output to register $R_2$ .	$b \cdot Q_{n-1}$ $aP_{n-1} + bQ_{n-1}$

4. Add " $X_{(n)} - X_{(n-N)}$ " for voltage from microprocessor to contents of register  $R_2$  in adder. The contents of  $R_1$  to  $R_7$ . Move contents of adder to  $R_8$ .
- $$aP_{n-1} + bQ_{n-1} + [X_{(n)} - X_{(n-N)}]_V$$
5. Input ' $Q_{n-1}$ ', ' $Q_{n-1}$ ' to multiplier. Multiply and store in accumulator.
- $$Q_{n-1}^2$$
- Input " $P_{n-1}$ ", " $P_{n-1}$ " to multiplier. Multiply. Add previous product to it and output to register  $R_2$ .
- $$P_{n-1}^2$$
- $$(P_{n-1}^2 + Q_{n-1}^2)_V$$
6. Disable voltage registers and enable current register i.e.  $R_4$  to  $R_6$ .
- Input ' $b_1$ ', ' $P_{n-1}$ ' to multiplier. Multiply and store in the accumulator.
- $$b \cdot P_{n-1}$$
- Input ' $a$ ', ' $Q_{n-1}$ ' to multiplier. Multiply. Subtract previous product from the accumulator and output the result to register  $R_4$ :
- $$a \cdot Q_{n-1}$$
- $$aQ_{n-1} - bP_{n-1}$$
7. Input ' $a$ ', " $P_{n-1}$ " to multiplier. Multiply and store in the accumulator.
- $$a \cdot P_{n-1}$$
- Input ' $b$ ', ' $Q_{n-1}$ ', to multiplier. Multiply. Add previous product to it and output to register  $R_5$ .
- $$b \cdot Q_{n-1}$$
- $$aP_{n-1} + bQ_{n-1}$$
8. Add " $X_{(n)} - X_{(n-N)}$ " for current from the microprocessor to the contents of register  $R_5$  in adder. Move contents of  $R_4$  to  $R_9$ .
- $$aP_{n-1} + bQ_{n-1} + [X_{(n)} - X_{(n-N)}]_I$$

Move contents of  $R_5$  to  $R_{10}$ .

9. Input ' $Q_{n-1}$ ', " $Q_{n-1}$ " to multiplier. Multiply  $Q_{n-1}^2$   
and store in the accumulator.
- Input ' $P_{n-1}$ ', ' $P_{n-1}$ ' to multiplier. Multiply.  $P_{n-1}$   
Add the previous product to it and output the  
result to register  $R_6$ .  $(P_{n-1}^2 + Q_{n-1}^2)_I$

After this sequence, the filtered outputs of voltage and current are available in registers  $R_3$  and  $R_6$  respectively.

The operation after filtering is the division of peak amplitude of square of voltage ( $v^2$ ) by peak amplitude of square of current ( $I^2$ ). The division is implemented using a successive approximation register and the fast multiplier.

### 3.7 Implementation of Division

The block diagram for implementing the division is given in Fig. 3.10.

The successive approximation register used is a 12 bit DM2504. The data sheet and truth table are given in Appendix D.

#### 3.7.1 Principle of Operation

Initially when  $\bar{s}$  goes from 'Low' to 'High',  $Q_{11}$  bit is set to 'low' and all other outputs go 'high'. This state is equivalent to 0111,1111,1111 in binary. This output A of successive approximation register (SAR) is fed to the multiplier input. Other input to multiplier

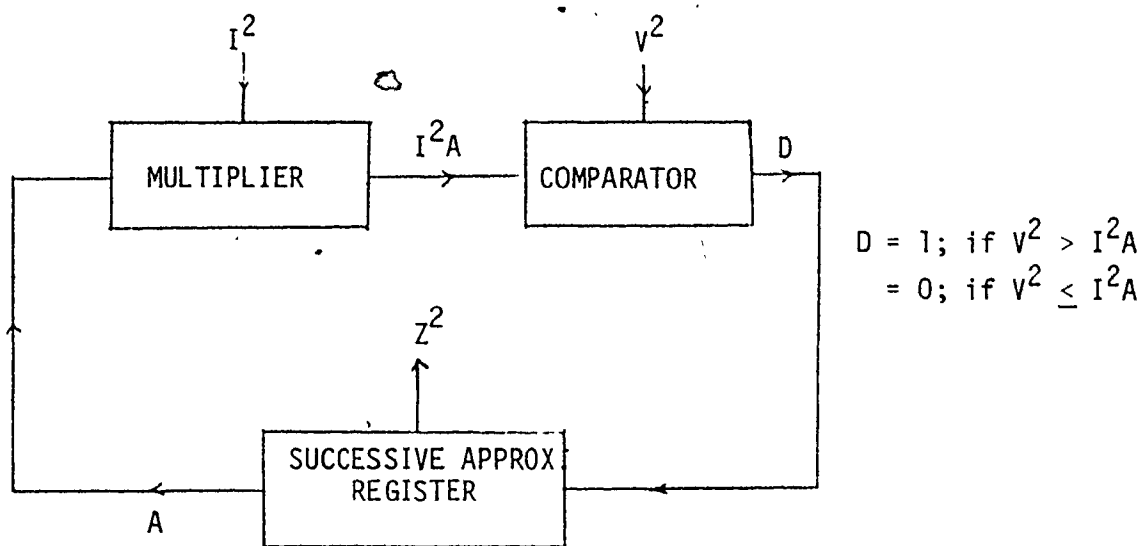


Fig. 3.10. Block diagram for division.

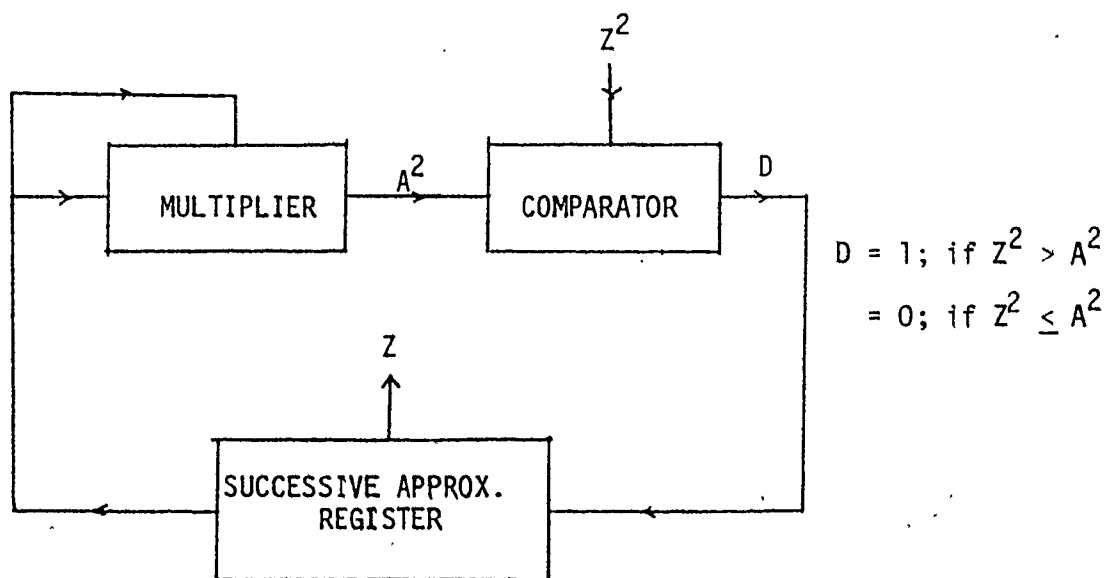


Fig. 3.11. Block diagram for square root.

is  $I^2$ . The product  $AI^2$  is compared with  $V^2$  in an magnitude comparator [7483]. If  $V^2$  is greater than  $AI^2$ , then 'D' (Appendix D) is set to '1'. If  $V^2$  is less than  $AI^2$  or equal to  $AI^2$  then 'D' is set to '0'.

At the next clock pulse  $C_p$ , the output of SAR is D011,1111,1111. This is multiplied by  $I^2$  and  $AI^2$  is compared with  $V^2$  and so on. The total number of clock pulses required for complete division is  $n+2$  where  $n$  is number of bits. For 12 bit SAR the total number of clock pulses required is 14. After that, the output of SAR gives the result of division. For a 12 bit SAR, the maximum number that can be handled is 4095.

The same principle of division can be used for square rooting. The result of division gives  $z^2$ . The block diagram for square rooting is shown in Fig. 3.11.

### 3.7.2 Sequence of Operation for the Division

Operation	Output of SAR
1. Initially " $\bar{s}$ " is set "LOW".	XXXX,XXXX,XXXX
2. Clock (SRA) Set " $\bar{s}$ " "HIGH". Clock (SRA) again.	0111,1111,1111
3. Input "A", and " $I^2$ " to multiplier. Multiply, output the result on the multiplier bus. Compare $AI^2$ and $V^2$ . Set D appropriately to $D_i$ .	



4. Clock (SRA)

$D_i$ : 011, 1111, 1111

Repeat steps 2 to 4 eleven times. The output of SRA gives the result of division,  $z^2$ .

## CHAPTER IV

### SOFTWARE

#### 4.1 Introduction

The operations required for on-line digital impedance calculation such as data-acquisition, calculating  $[X(n) - X(n-N)]$  values for filtering, filtering of voltage and current sample and division are all governed by a software program which is stored in the 4K EPROM. A flow chart for the process is shown in Fig. 4.1. A brief description of major steps involved in the digital impedance calculation process is given below.

#### 4.2 Initialization

The first block in the flow chart corresponds to initialization. At the beginning of a program, the ports of the SDK85 are assigned as Input or Output. Table 4.1 lists the assignment of different ports used and their functions.

The next operation is to clear output registers  $R_7$  to  $R_{10}$  shown in Fig. 3.8. The pointers are then set, which indicate the starting addresses for memory locations in which data is to be stored, the total number of samples to be stored, and the locations in which calculated impedance values are to be stored. Memory locations 2070 to 2087 of a 512 byte RAM are used for storing digitized samples of voltage and current. The interrupt is enabled. The microprocessor then executes a

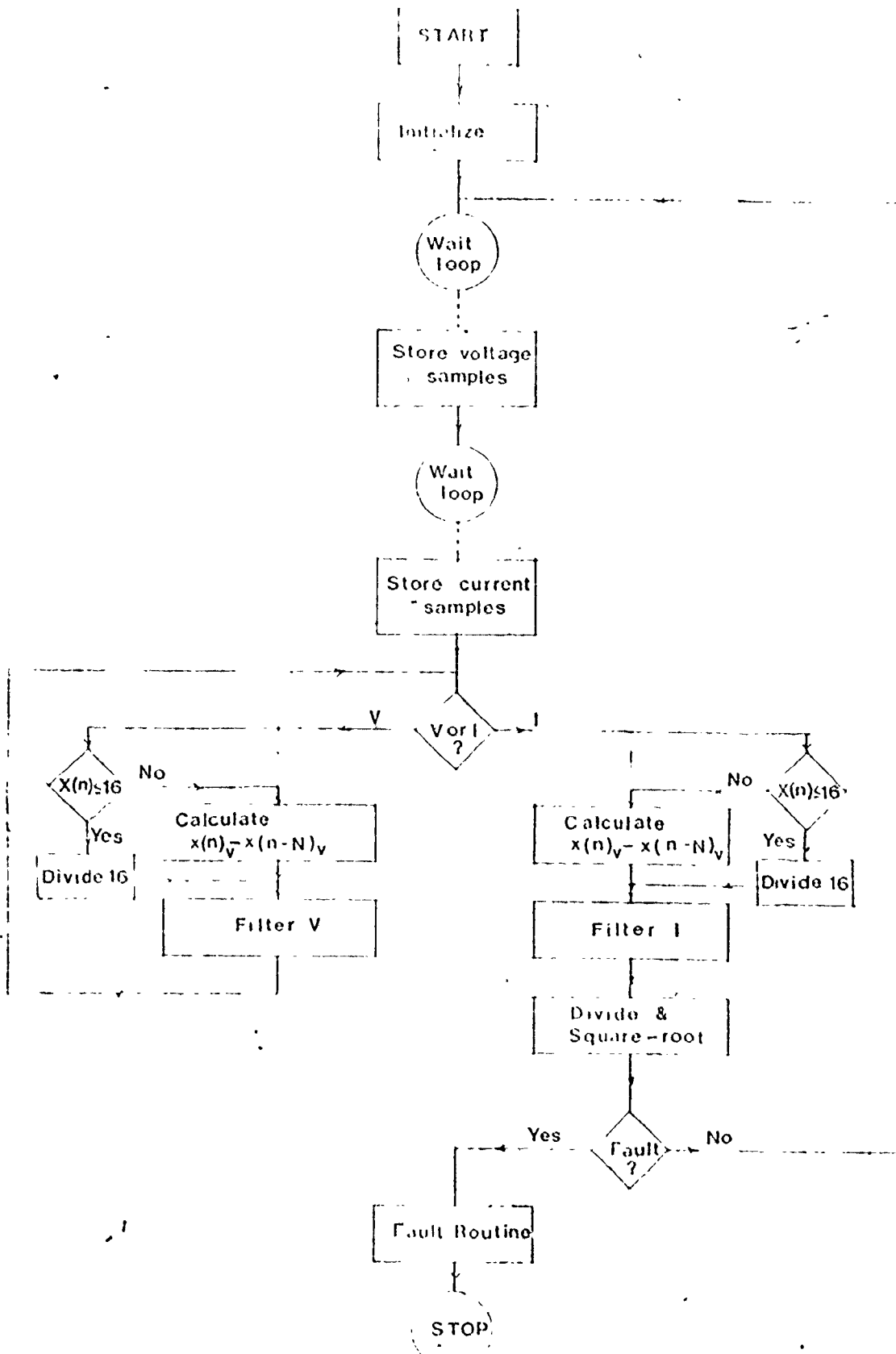


Fig. 4.1. Flow chart for Digital Impedance calculation.

Table 4.1

PORT	ASSIGN	DESCRIPTION
00/01	Output	To output $[X(n) - X(n-N)]$ for filter.
29/2A	Output	Used as control ports.
21/22	Input	To input data $X(n)$ from A/D converter.
09	Input	To take in impedance value $z$ from SAR.
02/03		Control status registers for ports 00/01.
20/28		Control status registers for ports 29/2A.

wait loop until an interrupt occurs.

#### 4.3 Interrupt

The SDK85 system has five interrupt inputs: INTR, RST5.5, RST6.5, RST7.5 and Trap. The three restart interrupts have programmable masks and these interrupts cause the internal execution of RST, saving the program counter in the stack and branching to the restart address. INTR and RST6.5 are the two interrupts available for users and RST6.5 is used in the implementation of the present system. This interrupt is high level sensitive, and points the control to monitor-reserved RAM location 20C8. So when the interrupt occurs, the instruction stored in location 20C8 is executed. Serving of this interrupt disables all future interrupts until the EI instruction is executed.

As discussed in Section 3.1.2, the interrupt signal is issued after every A/D conversion. Once the interrupt occurs, control passes to

location 20C8 where a 'JUMP' instruction is stored: This takes the control to a subroutine called DATAIN, which stores the data sequentially in memory locations 2070 to 20B7.

#### 4.4 Subroutine DATAIN

This subroutine takes in the data present on port 21 and port 22 and stores it in memory locations addressed by a pointer. Since the data is a 16 bit word and is in complemented form because of CTC logic used for A/D conversion, it is complemented before storing in two successive locations.

#### 4.5 Calculating $[X(n) - X(n-N)]$

For calculating  $[X(n) - X(n-N)]$  for voltage and current, 17 samples are stored in particular pattern. The memory map for data storage is given in Table 4.2. The first sample for voltage and current is stored in locations 2070 to 2073 as well as in locations 20B4 to 20B7 for easy subtraction.

Thus the expression  $[X(n) - X(n-N)]$  for voltage is obtained by subtracting the contents of the next two memory locations for the voltage sample from the present two memory locations. The same applies for current samples. This operation is shown in Table 4.3, for samples seventeen, eighteen and nineteen.

#### 4.6 Filter Algorithm

The sequence of operation for the filter is explained in Section

Table 4.2

Memory Location	16 bit Word (2 bytes); cycle 1	16 Bit Word (2 bytes); cycle 2	16 Bit Word (2 bytes); cycle 3 . .
2070/71	$X_{1v}(\text{MSB})/X_{1v}(\text{LSB})$	$X_{18v}(\text{MSB})/X_{18v}(\text{LSB})$	$X_{35v}(\text{MSB})/X_{35v}(\text{LSB})$
2072/73	$X_{1I}(\text{MSB})/X_{1I}(\text{LSB})$	$X_{18I}(\text{MSB})/X_{18I}(\text{LSB})$	$X_{35I}(\text{MSB})/X_{35I}(\text{LSB})$
2074/75	$X_{2v}(\text{MSB})/X_{2v}(\text{LSB})$	$X_{19v}(\text{MSB})/X_{19v}(\text{LSB})$	$X_{36v}(\text{MSB})/X_{36v}(\text{LSB})$
2076/77	$X_{2I}(\text{MSB})/X_{2I}(\text{LSB})$	$X_{19I}(\text{MSB})/X_{19I}(\text{LSB})$	$X_{36I}(\text{MSB})/X_{36I}(\text{LSB})$
2078/79	$X_{3v}(\text{MSB})/X_{3v}(\text{LSB})$	$X_{20v}(\text{MSB})/X_{20v}(\text{LSB})$	$X_{37v}(\text{MSB})/X_{37v}(\text{LSB})$
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
20AC/AD	$X_{16v}(\text{MSB})/X_{16v}(\text{LSB})$	$X_{33v}(\text{MSB})/X_{33v}(\text{LSB})$	$X_{50v}(\text{MSB})/X_{50v}(\text{LSB})$
20AE/AF	$X_{16I}(\text{MSB})/X_{16I}(\text{LSB})$	$X_{33I}(\text{MSB})/X_{33I}(\text{LSB})$	$X_{50I}(\text{MSB})/X_{50I}(\text{LSB})$
20B0/B1	$X_{17v}(\text{MSB})/X_{17v}(\text{LSB})$	$X_{34v}(\text{MSB})/X_{34v}(\text{LSB})$	$X_{51v}(\text{MSB})/X_{51v}(\text{LSB})$
20B2/B3	$X_{17I}(\text{MSB})/X_{17I}(\text{LSB})$	$X_{34I}(\text{MSB})/X_{34I}(\text{LSB})$	$X_{51I}(\text{MSB})/X_{51I}(\text{LSB})$
20B4/B5	$X_{1v}(\text{MSB})/X_{1v}(\text{LSB})$	$X_{18v}(\text{MSB})/X_{18v}(\text{LSB})$	$X_{35v}(\text{MSB})/X_{35v}(\text{LSB})$
20B6/B7	$X_{1I}(\text{MSB})/X_{1I}(\text{LSB})$	$X_{18I}(\text{MSB})/X_{18I}(\text{LSB})$	$X_{35I}(\text{MSB})/X_{35I}(\text{LSB})$

Table 4.3

(n) current sample value	memory locations	16 bit data stored	output on port $O_0$ & $O_1$
17	20B0/B1	$X_{17v}$	$X_{17v} - X_{1v}$
	20B2/B3	$X_{17I}$	$X_{17I} - X_{1I}$
	20B4/B5	$X_{1v}$	
	20B6/B7	$X_{1I}$	
18	2070/71	$X_{18v}$	$X_{18v} - X_{2v}$
	2072/73	$X_{18I}$	$X_{18I} - X_{2I}$
	2074/75	$X_{2v}$	
	2076/77	$X_{2I}$	
19	2074/75	$X_{19v}$	$X_{19v} - X_{3v}$
	2076/77	$X_{19I}$	$X_{19I} - X_{3I}$
	2078/79	$X_{3v}$	
	2080/81	$X_{3I}$	
20	2078/79	$X_{20v}$	$X_{20v} - X_{4v}$
	2080/81	$X_{20I}$	$X_{20I} - X_{4I}$
	2082/83	$X_{4v}$	
	2084/85	$X_{4I}$	

3.6. Memory location 2002 of RAM is used as a pointer, which decides whether the sample is for voltage or current and selects either the voltage filter subroutine or current filter subroutine, appropriately.

The filtering algorithm consists of writing appropriate data to the control ports; port 29 and port 2A shown in Fig. 3.9 and then sending these control signals out using an OUTPUT instruction.

To explain in brief, the program for step 5 in the sequence of operation for the filter in Section 3.5, one needs to get  $Q_{n-1}^2$  first. Thus we enter ' $Q_{n-1}$ ', to both of the multiplier inputs by selecting the proper controls for multiplexers  $M_1$  and  $M_2$  from table. For these inputs the control port 29 in Fig 3.9 becomes

0    1    0    0    0    1    1    0    Port 29

and the instructions are,

```
MVI A,46    move word 46 to accumulator,
OUT 29 ;    send out the contents of accumulator.
```

After the output instruction is executed the multiplier takes in the data and multiplies it. The product  $Q_{n-1}^2$  is clocked out to the output register of the multiplier. At the same time, the next data is made available at the inputs of the multiplier, which in this case is  $P_{n-1}$ . This is done by the instructions,

```
MVI A, A8
OUT 29
```

The next operation is to take in new data, multiply it and add to it previous product  $Q_{n-1}^2$ . To do this we use



```
MVI A, C8
```

```
OUT 29.
```

So now the multiplier accumulator contains  $P_{n-1}^2 + Q_{n-1}^2$ . To move this value to the output bus of multiplier we use

```
MVI A, E8
```

```
OUT 29.
```

To transfer this value in register  $R_3$  of Fig.3.8.  $R_3$  is clocked by setting bit 3 of port 2A,

```
MVI A, 2B;    move word 2B to accumulator
```

```
OUT 2A;       clock  $R_3$ .
```

These instructions complete step 4 of Section 3.5.

The filtering algorithm is self explanatory. A complete listing of the program is given in Appendix E.

The value  $[X(n) - X(n-N)]$  for voltage or current is calculated before the filter algorithm starts and is sent to the adders in Fig. 3.8. The filter algorithm is the same for voltage and current except for the clocking of the registers. The filtered outputs of voltage  $[V^2]$  and current  $[I^2]$  are stored in registers  $R_3$  and  $R_6$  respectively. Then the processor proceeds for the division algorithm.

#### 4.7 Division Algorithm

The sequence of operations discussed in Section 3.7.2 is followed.  $\bar{s}$  is set low initially during the first clock pulse and then set high until the division is complete. Fourteen clock pulses are required for complete

division, after which it stores the impedance value  $Z^2$  in successive approximation register. The division algorithm is again straightforward and is listed in Appendix E.

#### 4.8 Subroutine DIVIDE16

For the first sixteen samples the past values of voltage and current are zero and the filter Eq. 2.32 reduces to

$$S_n = (P_{n-1} + jQ_{n-1})(a - jb) + X(n) \quad 4.1$$

since  $X(n - N) = 0$ ;  $0 < n \leq 16$ .

Eq. 4.1 shows that the filtered output is sixteen times the peak value of voltage or current. This requires either scaling down of input waveforms or it restricts the input range of the A/D converter. To use the maximum range of the A/D converter without overloading it the first sixteen samples are divided by sixteen, so that Eq. 4.1 becomes

$$S_n = (P_{n-1} + jQ_{n-1})(9 - jb) + [X(n)/16]; \quad 0 < n \leq 16$$

In the program, the first sixteen samples are divided by 16 and sent out on port 00 and port 01, which for the rest of samples, outputs  $[X(n) - X(n - N)]$ .

Subroutine DIVIDE16 takes the sample in register pair H and L, shifts the contents of this register pair to the right four times using instruction "RHL". The most significant bit is duplicated and the lowest bit is shifted to the carry, keeping the sign bit unchanged.

#### 4.9 Impedance Algorithm

Once the division is complete, the contents of the successive approximation register are taken into the accumulator of the microprocessor. This value is then compared with a reference impedance value which is the normal impedance of the power line. If the value changes by more than  $\pm 5\%$ , the fault is 'detected' and calls a fault routine.

#### 4.10 Fault Routine

Once a fault is sensed, the microprocessor stores the next 50 values of impedance in the memory locations 2820 to 2851 of RAM.

## CHAPTER V

### DISCUSSION

#### 5.1 Test Results

The signal used for testing the system is  $v = 4 \cos(2\pi \times 60t)$  volts.

The current waveform is derived from the voltage using a voltage divider. Data is stored in locations 2070 to 20B7 of RAM for voltage and current. Impedance values are stored in locations 2820 to 2851 of RAM. Data values and the impedance values are given in Table 5.1 and Table 5.2 respectively. The program is written in assembly language and is listed in Appendix E.

#### 5.2 Software Simulation of Filter

The filter Eq. 2.32 discussed in Section 2.5 was simulated on a Hewlett Packard graphic terminal 2647 A. Test signals for voltage and current are taken as

$$V = X(J) = 1024 * \sin\omega_0 t + 102 * \sin 3\omega_0 t \quad \text{and}$$

$$I = Y(J) = 102.4 * \sin\omega_0 t$$

These equations are chosen for two reasons. 1. To find the effect of the third harmonics on the filter operation and 2. To find how fast the fault alarm can be issued to give an advance indication of the fault.

To simulate the fault, the voltage amplitude was reduced by a

Table 5.1

---

Memory Location	16 Bit Data Stored
2070/2071	F9C5
2072/2073	FDE5
2074/2075	FA98
2076/2077	FE38
2078/2079	FC38
207A/207B	FEC7
207C/207D	FE67
207E/207F	FF87
2080/2081	00D8
2082/2083	004F
2084/2085	0325
2086/2087	010B
2088/2089	04F1
208A/208B	0196
208C/208D	05F8
208E/208F	01E0
2090/2091	0615
2092/2093	01DA
2094/2095	0540
2096/2097	018B
2098/2099	0394
209A/209B	00F8
209C/209D	015C
209E/209F	003C
20A0/20A1	FEE8
20A2/20A3	FF70
20A4/20A5	FCA2
20A6/20A7	FEB8
20A8/20A9	FC8F
20AA/20AB	FEB1
20AC/20AD	FACF
20AE/20AF	FE27
20B0/20B1	F9D7
20B2/20B3	FDDE
20B4/20B5	F9C5
20B6/20B7	FDE5

---

Table 5.2

---

Memory Location	Impedance $ Z^2 $	Memory Location	Impedance $ Z^2 $
2820	00	2839	09
2821	FF	283A	09
2822	0B	283B	09
2823	0C	283C	09
2824	0C	283D	09
2825	0C	283E	0A
2826	0F	283F	0A
2827	0E	2840	09
2828	0A	2841	09
2829	09	2842	09
282A	0A	2843	09
282B	0A	2844	09
282C	0A	2845	0A
282D	0A	2846	0A
282E	0B	2847	0A
282F	0A	2848	0A
2830	0A	2849	09
2831	0A	284A	09
2832	0A	284B	09
2833	09	284C	09
2834	0A	284D	09
2835	0A	284E	09
2836	0A	284F	0A
2837	0A	2850	09
2838	09	2851	09

---

factor of 1.17 and current amplitude was introduced by a factor of 8.5. These factors are obtained using a circuit shown in Appendix G. The fault is simulated by shorting the load. After the fault, the voltage and current waveforms are,

$$V = X(J) = 870.4 * \sin\omega_0 t + 87.04 * \sin 3\omega_0 t \quad \text{and}$$

$$J = Y(J) = 870.4 * \sin\omega_0 t$$

The program is written in basic language and is listed in Appendix F. The impedance vs. number of samples is plotted in Fig. 5.1. For a  $\pm 5\%$  variation of impedance from its normal value, it is seen that a fault can be sensed as early as the 3rd sample, i.e. within 3.12 msec.

### 5.3 Accuracy

The accuracy of the impedance calculation depends on the number of bits used for A/D conversion. This is the digitization error in voltage sample, current sample and in coefficients 'a' and 'b' of the digital filter. The most significant bit is a sign bit for A/D conversion, hence the maximum possible error is  $\pm 1/2$  of  $2^{-11}$ , i.e.  $\pm 2^{-12}$ . Let this quantization error be denoted by  $\epsilon$ . Other factor affecting the accuracy of the truncation error introduced by each multiplication. This error is also  $\pm 2^{-12}$ .

Total error is calculated by calculating the error introduced in obtaining filtered voltage and current outputs and then calculating error introduced in division and square rooting. Complete error analysis is given in Appendix H and is found to be  $\pm 1.52\%$  of the maximum possible

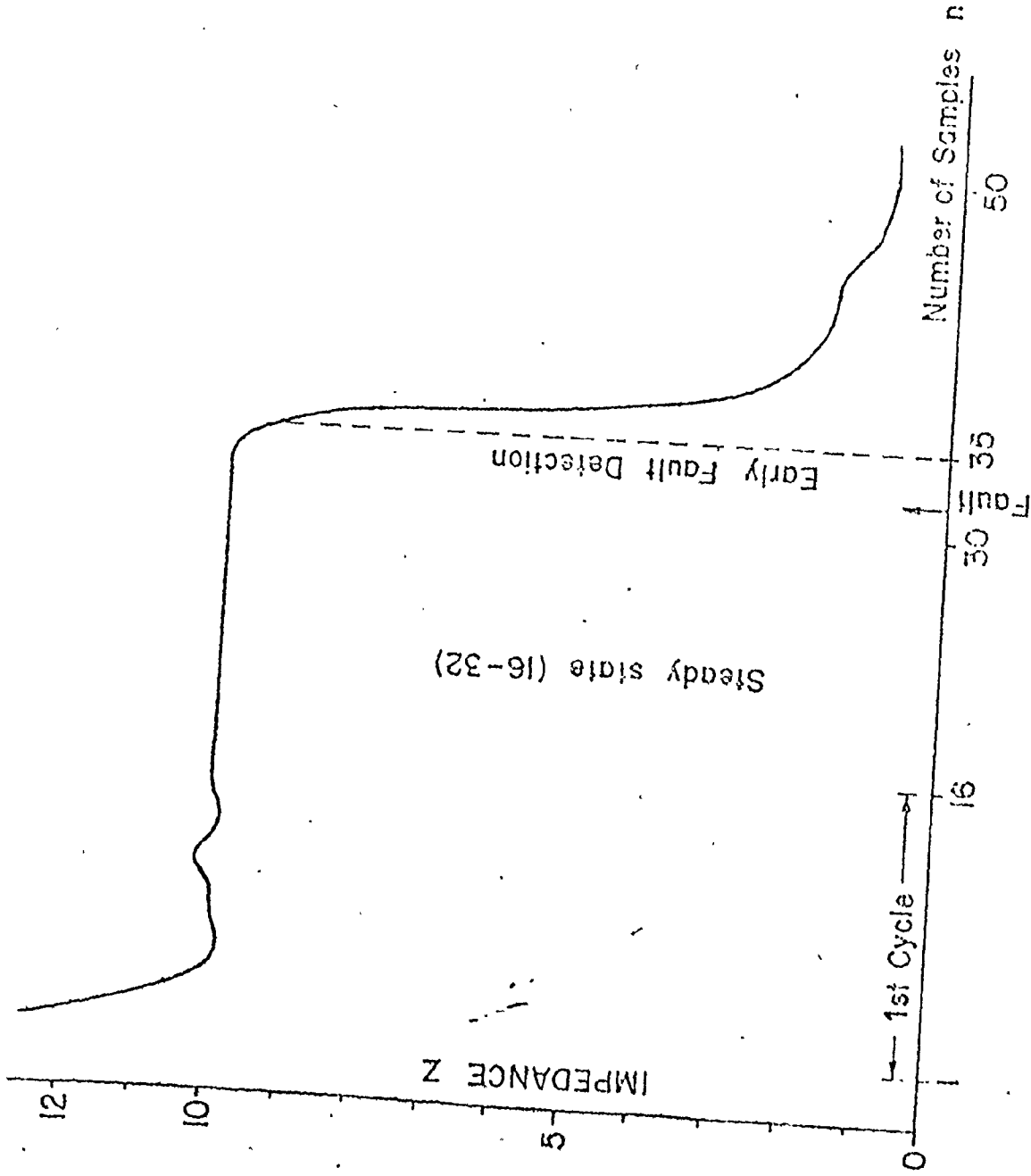


Fig. 5.1. Test Result. A fault is applied at the 30th sample time.



impedance value.

#### 5.4 Execution Time

To compute the execution time, the number of instructions have to be counted. A state time for the 8085 micrporcessor is 0.33  $\mu$ sec. Table 5.3 lists the number of states in each section of the program, the number of times the set is looped, and the total number of states.

To calculate the execution time, the time taken for initialization is not considered. The execution time for the program to calculate  $z^2$  is 758  $\mu$ sec. The number of states required for the square rooting is the same as that for the division. Hence  $z$  is calculated in 1.933 msec, which is within one sample period at 60 Hz.

#### 5.5 Size

The entire system comprises an SDK85 single board computer and an additional 9x5 inch board. A fast multiplier, an A/D converter and multiplexers are wire-wrapped on the expansion area of SDK85 board. The additional board has 40 IC's which include the successive approximation register, other registers and adders. This board is connected to the SDK85 board via two 50 pin flat cables.

A photograph for the complete system is shown in Fig. 5.2.

#### 5.6 Extensions and Improvements

The main consideration in designing a system for on-line

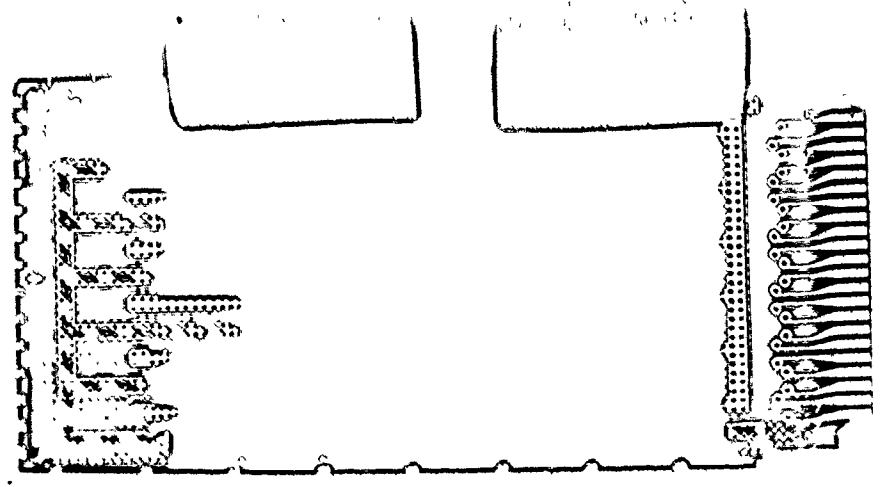
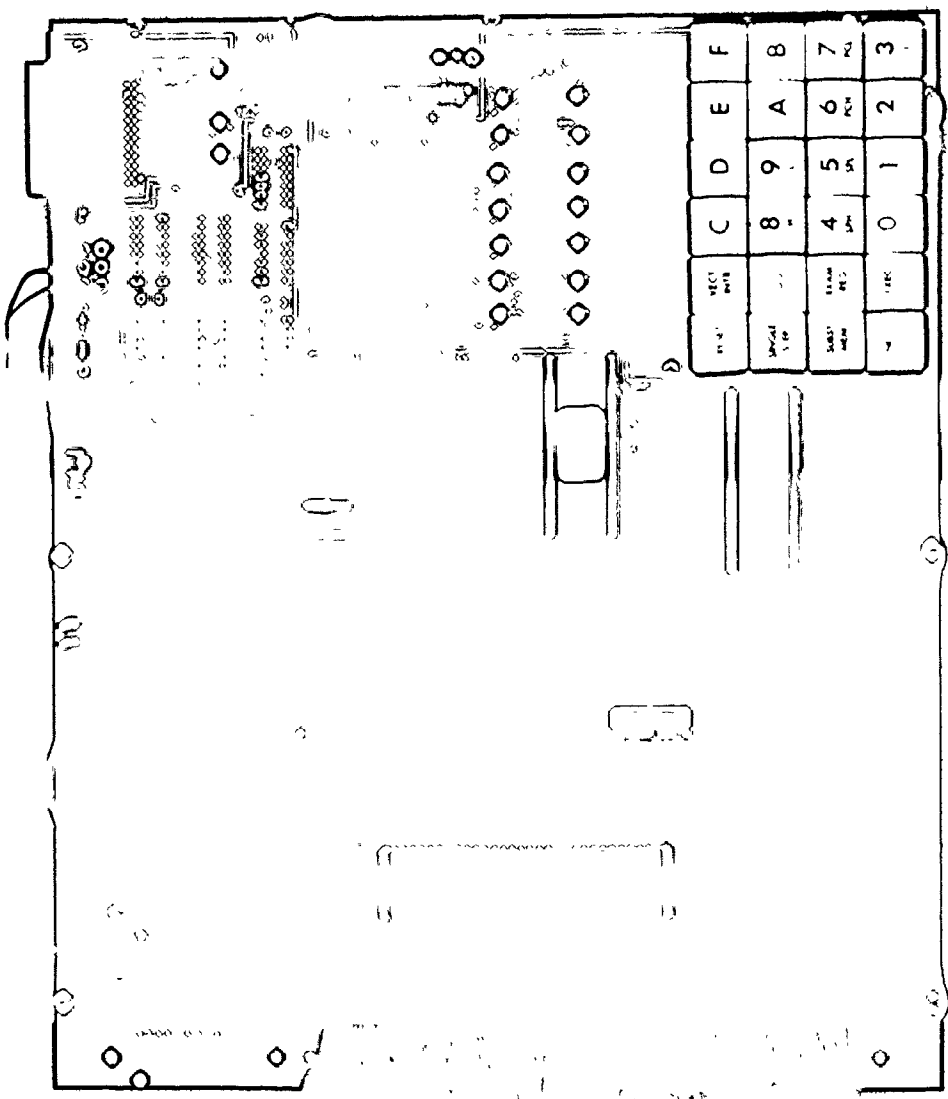


Table 5.3

Section	No. of States	No. of Timer Executed	Total No. of States
Initialization	338	1	338*
Routine $X_n - X_{n-m}$	142	2	284
Filter I	302	1	302
Filter V	302	1	302
Division	59	14	826
Impedance Storage	106	1	106
Divide 16	172	1	172**
Rst 6.5	10	2	20
Intrpt.	159	2	318
DataIn.	58	2	116
Total number of states to calculate $ z^2 $			2274

\* Initialization is done at the beginning of program only and hence is not accounted in total number of states.

\*\*Divide16 occurs only for the first 16 samples when  $X_n - X_{n-m}$  is not done, since DIVIDE16 takes less states than for  $X_n - X_{n-m}$ , DIVIDE16 is not considered in the total number of states.

impedance calculation is speed. Hence it is advantageous to use a faster microprocessor. Since a 12 bit word is used for the digital processing, it would be more appropriate to use a 16 bit microprocessors; these are now becoming available (Intel 8086, Motorola 68000, Zilog Z8000).

The Intel 8086 has a clock rate of 5MHz compared to 2MHz for the 8085 microprocessor and if used, the execution time would be reduced to 618  $\mu$ secs.

The fast multiplier TDC 1003J takes typically 175 nsec for multiplication and accumulation. Though the multiplier by itself is capable of operating at such a high speed, in the present system the speed is effectively reduced since the multiplier is controlled by a relatively slow microprocessor. The microprocessor has an instruction cycle time of 1.3  $\mu$ sec, and one multiplication, in effect is achieved in about 2.6  $\mu$ secs. (each multiplication takes two instruction cycles). To take better advantage of the fast multiplier, the control and clock signals should be derived from a clock drive circuit using faster digital hardware.

Recently Howard, Mitra and Mahbod [8] have shown that using such a clock driven circuit and fast multiplier a second order FIR filter function can be implemented in about 750 nsecs.

The work undertaken has shown that it is possible to monitor the on-line impedance characteristics for a single phase transmission line. The next task is to investigate the monitoring of three phases using a single microprocessor as a controller. To achieve this using serial processing, it is essential to cut down the execution time to about a third of the sampling period. If a digital filter is designed using an external clock drive as discussed by Howard [8], the filtering can be achieved in

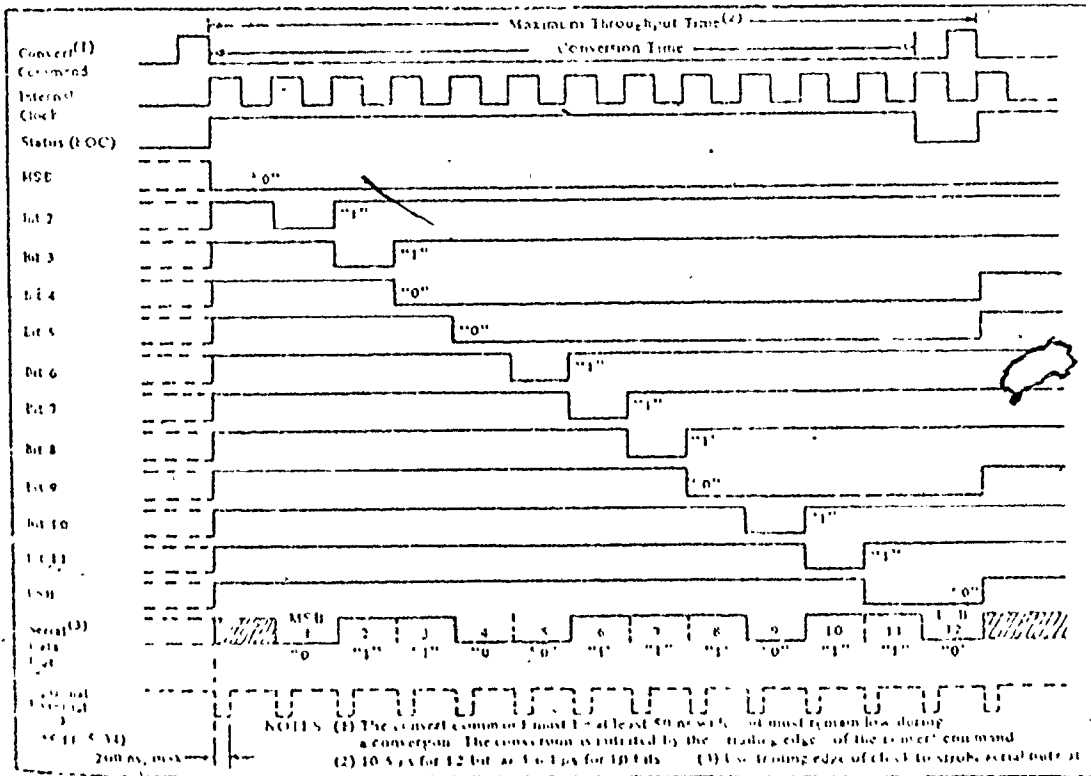
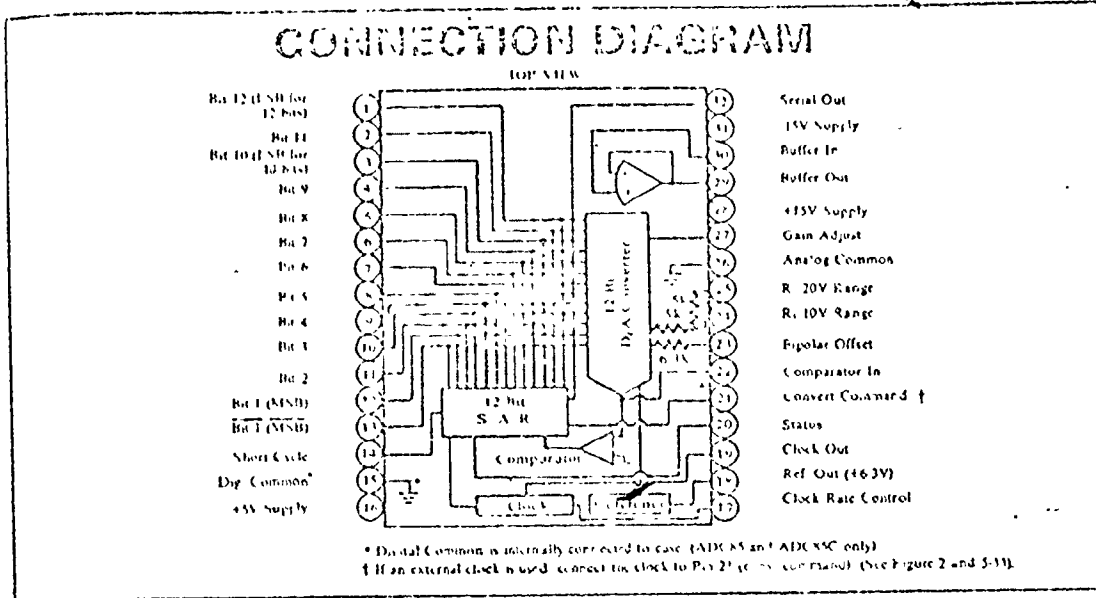
about 875 nsec. Division and square rooting will take about 2.45  $\mu$ sec. each. Hence a complete impedance calculation can be achieved well within 300  $\mu$ secs, taking into account the time taken by A/D conversion and the data storage routine.

APPENDICES



# APPENDIX A

## A/D CONVERTER (ADC85KG)



Timing diagram for ADC85

## APPENDIX B

## 8085A INSTRUCTION SET SUMMARY

Mnemonic	Description	Instruction Code(s)							Clock(s)	Cycles	Mnemonic	Description	Instruction Code(s)							Clock(s)	Cycles
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>					D <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>		
<b>MOVE, LOAD, AND STORE</b>										CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18	
MOV r1 r2	Move register to register	0	1	0	0	0	S	S	4	CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18	
MOV M r	Move register to memory	0	1	1	1	0	S	S	7	<b>RETURN</b>											
MOV r M	Move memory to register	0	1	0	0	0	1	1	0	7	RET	Return	1	1	0	0	1	0	0	1	10
MVI r	Move immediate register	0	0	0	0	0	1	1	0	7	RC	Return on carry	1	1	0	1	1	0	0	0	6/12
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10	RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10	RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10	ANZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10	RP	Return on positive	1	1	1	0	0	0	0	0	6/12
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10	RM	Return on minus	1	1	1	1	0	0	0	0	6/12
STAX B	Store A indirect	0	0	0	0	0	1	0	7	RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12	
STAX D	Store A indirect	0	0	0	1	0	0	1	7	RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12	
LDAX B	Load A indirect	0	0	0	0	1	0	1	7	<b>RESTART</b>											
LDAX D	Load A indirect	0	0	0	1	1	0	1	7	RST	Restart	1	1	A	A	A	1	1	1	12	
STA	Store A direct	0	0	1	1	0	0	1	13	<b>INPUT/OUTPUT</b>											
LDA	Load A direct	0	0	1	1	1	0	1	13	IN	Input	1	1	0	1	1	0	1	1	10	
SHLD	Store H & L direct	0	0	1	0	0	0	1	16	OUT	Output	1	1	0	1	0	0	1	1	10	
LHLD	Load H & L direct	0	0	1	0	1	0	1	16	<b>INCREMENT AND DECREMENT</b>											
XCHG	Exchange D & E H & L Registers	1	1	1	0	1	0	1	4	INR r	Increment register	0	0	0	0	1	0	0	4		
<b>STACK OPS</b>										DCR r	Decrement register	0	0	0	0	1	0	1	4		
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	12	INR M	Increment memory	0	0	1	0	1	0	10			
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	12	DCR M	Decrement memory	0	0	1	1	0	1	10			
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	12	INX B	Increment B & C registers	0	0	0	0	0	1	1	5		
PUSH PSW	Push A and Flags on stack	1	1	1	0	1	0	1	12	INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6	
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	1	10	INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6	
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	1	10	INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6	
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	1	10	DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6	
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	1	10	DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6	
XTHL	Exchange top of stack H & L	1	1	1	0	0	0	1	16	DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6	
SPHL	H & L to stack pointer	1	1	1	1	0	0	1	6	DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6	
<b>JUMP</b>										<b>ADD</b>											
JMP	Jump unconditional	1	1	0	0	0	0	1	10	ADD r	Add register to A	1	0	0	0	S	S	S	4		
JC	Jump on carry	1	1	0	1	1	1	1	10	ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4	
JNC	Jump on no carry	1	1	0	1	0	0	1	10	ADD M	Add memory to A	1	0	0	0	1	1	0	7		
JZ	Jump on zero	1	1	0	0	1	0	1	10	ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7	
JNZ	Jump on no zero	1	1	0	0	0	1	0	10	ADI	Add immediate to A	1	1	0	0	0	1	1	0	7	
JP	Jump on positive	1	1	1	0	0	1	0	10	ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7	
JP	Jump on minus	1	1	1	1	0	1	0	10	DAD B	Add B & L to H & L	0	0	0	0	1	0	0	1	10	
JPE	Jump on parity even	1	1	1	0	1	0	1	10	DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10	
JPO	Jump on parity odd	1	1	1	0	0	1	0	10	DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10	
PCHL	H & L to program counter	1	1	1	0	1	0	1	6	DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10	
<b>CALL</b>										<b>SUBTRACT</b>											
CALL	Call unconditional	1	1	0	0	1	1	0	18	SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4	
CC	Call on carry	1	1	0	1	1	1	0	9/18	SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4	
CNC	Call on no carry	1	1	0	1	0	1	0	9/18	SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7	
CZ	Call on zero	1	1	0	0	1	1	0	9/18	SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7	
CNZ	Call on no zero	1	1	0	0	0	1	0	9/18	SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7	
CP	Call on positive	1	1	1	0	1	0	0	9/18	SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7	
CM	Call on minus	1	1	1	1	1	0	0	9/18	<b>LOGICAL</b>											
										ANA r	And register with A	1	0	1	0	0	S	S	S	4	



8085A INSTRUCTION SET SUMMARY (Cont.)

Mnemonic	Description	Instruction Code(1)							Clocks(2)	Mnemonic	Description	Instruction Code(1)							Clocks		
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>				D <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub> -D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>		D <sub>0</sub>	
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4	RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4	RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4	<b>SPECIALS</b>										
ANA M	And memory with A	1	0	1	0	0	1	1	0	7	CMA	Complement A	0	0	1	0	1	1	1	1	4
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7	STC	Set carry	0	0	1	1	0	1	1	1	4
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7	CMC	Complement carry	0	0	1	1	1	1	1	1	4
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7	DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
ANI	And immediate with A	1	1	1	0	0	1	1	0	7	<b>CONTROL</b>										
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7	EI	Enable interrupts	1	1	1	1	1	0	1	1	4
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7	DI	Disable interrupts	1	1	1	0	0	0	1	1	4
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7	NOP	No operation	1	0	0	0	0	0	0	0	4
<b>ROTATE</b>										<b>NEW 8085A INSTRUCTIONS</b>											
RLC	Rotate A left	0	0	0	0	0	1	1	1	4	SHL	Shift register left	1	0	1	0	0	0	0	0	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4	SHR	Shift register right	1	0	1	1	0	0	0	0	4

NOTES 1 DDD or SSS B 000 C 001 D 010 E 011 H 100 L 101 Memory, M, A 111  
 2 Two possible cycle times (6/12) indicate instruction cycles dependent on condition flags

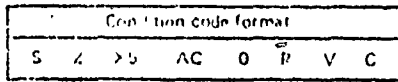
\*All mnemonics copyright © Intel Corporation 1977

SUMMARY OF 8085A INSTRUCTIONS

INSTR	STATES	CYCLES	NOTES	INSTR	STATES	CYCLES	NOTES
AR r	4	1		OUT	10	3	
AR I	7	2		PCHL	6	1	2
AR M	7	2		POP	10	3	
CALL	18	5	2	PUSH	12	3	2
CCN	9/18	2/5	2,4	ROT	4	1	
CMC	4	1		RET	10	3	
CMA	4	1		RCN	6/12	1/3	2
DAA	4	1		RIM*	4	1	
DAD	10	3		RST	12	3	2
DCR r	4	1	3	SHLD	16	5	
DCR M	10	3		SIM*	4	1	
DCX	6	1	2	STA	13	4	
DI	4	1		STAX	7	2	
EI	4	1		STC	4	1	
HLT	5	1	5	XCHG	4	1	
IN	10	3		XTHL	16	5	
INR r	4	1	3	SPLH	6	1	2
INR M	10	3		<b>*New Instruction</b>			
INX	6	1	2	<b>NOTES</b>			
JMP	10	3		1 Two possible state-cycle times indicates dependence on condition flag.			
JCN	7/10	2/3	4	2 Increase in states over 8080 due to necessity of a Tg during M1.			
LDS	13	4		3 Decreased from 5 to 4 states during M1.			
LDAX	7	2		4 Instruction branches over 4M address latch if condition false.			
LHLD	16	5		5 5 cycles to get a HLT state, 1 cycle necessary to get out of a HLT.			
LXU	10	3					
MVI M	10	3					
MVI r	7	2					
MOV M,r	7	2					
MOV r,M	7	2					
MOV r,r	4	1	3				
NOP	4	1					

NEW CONDITION CODES.

V = bit 1  
X5 = bit 5

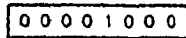


2's complement overflow  
Underflow (Ox) or overflow (Ox)  
 $X5 = 01 \cdot 02 + 01 \cdot R + 02 \cdot R$ , where  
01 = sign of operand 1, 02 = sign of operand 2,  
R = sign of result. For subtraction and comparisons,  
replace 02 with 0Z.

DSUB (double subtraction)

$(H) (L) = (H) (L) - (B) (C)$

The contents of register pair B and C are subtracted from the contents of register pair H and L. The result is placed in register pair H and L. All condition flags are affected.



(08)

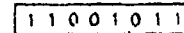
cycles: 3  
states: 10  
addressing: register  
flags: Z, S, P, CY, AC, X5, V

RSTV (restart on overflow)

If (V):

$(SP) - 1 = (PCH)$   
 $(SP) - 2 = (PCL)$   
 $(SP) = (SP) - 2$   
 $(PC) = 40$  hex

If the overflow flag V is set, the actions specified above are performed, otherwise control continues sequentially



(CB)

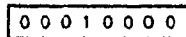
cycles: 1 or 3  
states: 6 or 12  
addressing: register indirect  
flags: none

ARHL (arithmetic shift of H and L to the right)

$(H7) = (H7), (Hn-1) = (Hn)$

$(L7) = (Lo), (Ln-1) = (Ln); (CY) = (Lo)$

The contents of register pair H and L are shifted right one bit. The uppermost bit is duplicated and the lowest bit is shifted into the carry bit. The result is placed in register pair H and L. Note: only the CY flag is affected.



(10)

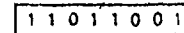
cycles: 2  
states: 7  
addressing: register  
flags: CY

SHLX (store H and L indirect through D and E)

$((D)(E)) = (L)$

$((D)(E)+1) = (H)$

The contents of register L are moved to the memory location whose address is in register pair D and E. The contents of register H are moved to the succeeding memory location



(D9)

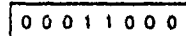
cycles: 3  
states: 10  
addressing: register indirect  
flags: none

RDEL (rotate D and E left through carry)

$(Dn+1) = (Dn), (Do) = (E7)$

$(CY) = (D7), (En+1) = (En); (Eo) = (CY)$

The contents of register pair D and E are rotated left one position through the carry flag. The low-order bit is set equal to the CY flag; and the CY flag is set to the value shifted out of the high-order bit. Only the CY and the V flags are affected.



(18)

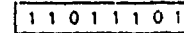
cycles: 3  
states: 10  
addressing: register  
flags: CY, V

JNX5 (jump on not X5)

If (not X5)

$(PC) = (\text{byte } 3) (\text{byte } 2)$

If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction, otherwise control continues sequentially



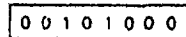
(DD)

low order address  
high order address  
cycles: 2 or 3  
states: 7 or 10  
addressing: immediate  
flags: none

LDHI (load D and E with H and L plus immediate byte)

$(D) (C) = (H) (L) + (\text{byte } 2)$

The contents of register pair H and L are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected.



(28)

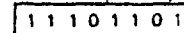
data  
cycles: 3  
states: 10  
addressing: immediate register  
flags: none

LHLX (load H and L indirect through D and E)

$(L) = ((D)(E))$

$(H) = ((D)(E)+1)$

The contents of the memory location whose address is in D and E, are moved to register L. The contents of the succeeding memory location are moved to register H



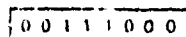
(ED)

cycles: 3  
states: 10  
addressing: register indirect  
flags: none

LDSI (load D and E with SP plus immediate byte)

$(D) (E) = (SP)(SP) + (\text{byte } 2)$

The contents of register pair SP are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected



(39)

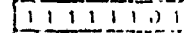
data  
cycles: 3  
states: 10  
addressing: immediate register  
flags: none

JX5 (jump on X5)

If (X5)

$(PC) = (\text{byte } 3) (\text{byte } 2)$

If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction, otherwise control continues sequentially



(FD)

low order address  
high order address  
cycles: 2 or 3  
states: 7 or 10  
addressing: immediate  
flags: none

## APPENDIX C

## FAST MULTIPLIER-ACCUMULATOR

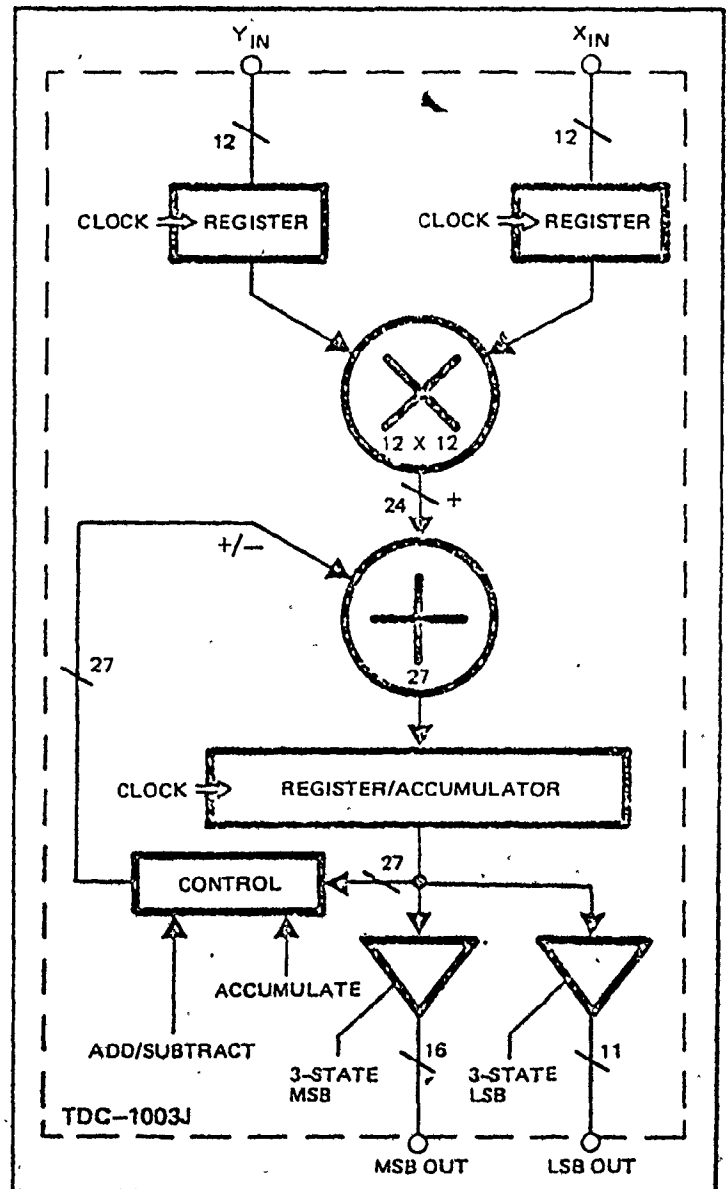
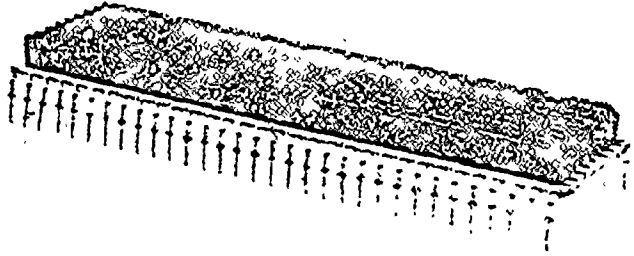
## Model: TDC1003J

The TDC1003J is a multifunction arithmetic unit capable of performing 12 x 12 multiplication as well as product accumulation. It has an additional feature of permitting the accumulator contents to be subtracted from the next product instead of being added, if desired. Input registers are provided in addition to the product accumulation register.

The TDC1003J is directly implementable as the central building block for digital filters, particularly FFTs, for complex multipliers, and for recursive and nonrecursive filter elements.

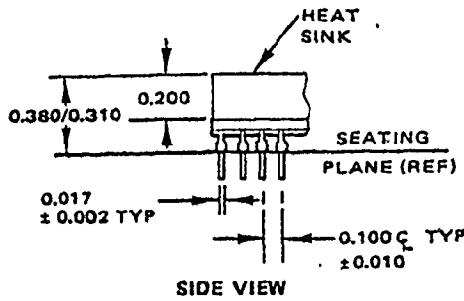
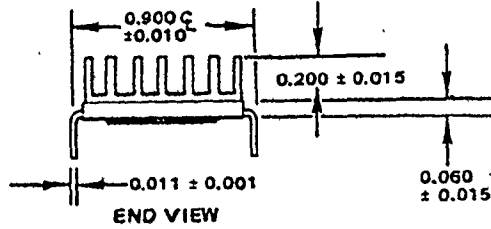
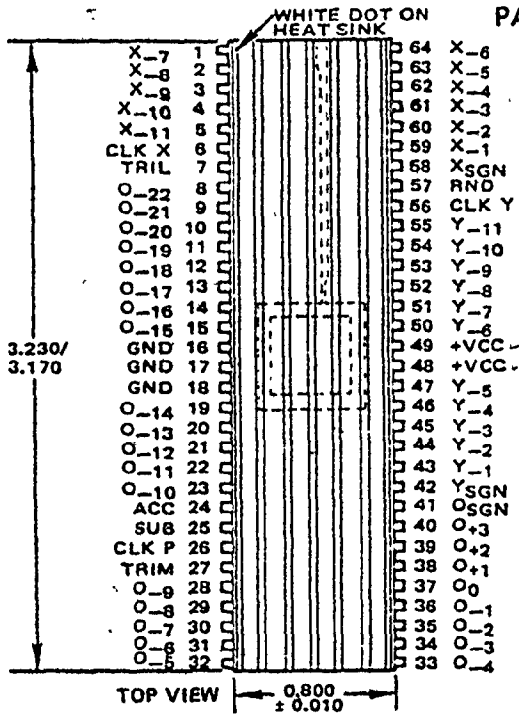
## FEATURES

- 12 x 12 Bit Parallel, Two's Complement Multiplication
- Controllable Accumulation Either + or -
- 175 nsec Typical Multiply and Accumulate Time
- Much Lower Power/Faster Speed than Equivalent MSI Multiplication-Accumulation Systems
- Round Control
- 27-Bit Accumulation Capacity
- Single Chip, Bipolar Technology
- Asynchronous Mode Multiply
- Radiation Hard
- TTL Input and Output
- Three State Outputs
- Single Power Supply, +5 Volts
- Dual In-Line Package or Flat Pack
- 2.5 Watts Power Consumption



TDC1003J

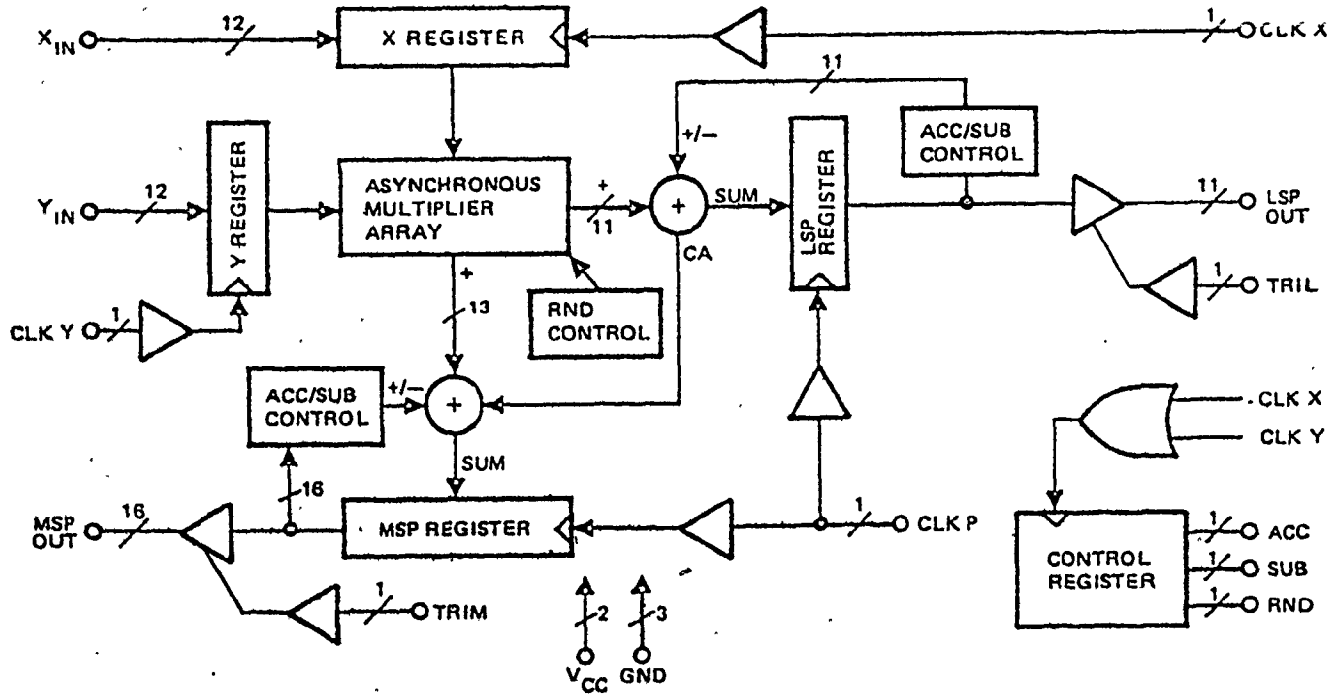
PACKAGE INFORMATION



NOTES:

- DIMENSIONS IN INCHES
- ALL V<sub>CC</sub> AND GND PINS MUST BE CONNECTED.

LOGICAL BLOCK



CONTROLS

- CLK X, X<sub>IN</sub> REGISTER CLOCK } REGISTER CLOCK FOR RND, ACC, AND SUB IS (CLK X + CLK Y)
- CLK Y, Y<sub>IN</sub> REGISTER CLOCK }
- CLK P, OUTPUT REGISTER CLOCK
- TRIL, LSP THREE STATE CONTROL
- TRIM, MSP THREE STATE CONTROL
- RND, ADDS 2<sup>-12</sup> TO PRODUCT (FRACTIONAL 2S COMPLEMENT FIELD)
- ACC, ENABLES ACCUMULATOR MODE
- SUB, CONTROLS ADDITION/SUBTRACTION OF ACCUMULATOR CONTENTS

FORMAT: 2'S COMPLEMENT FRACTIONAL NOTATION  
(NOTE 3)

X INPUT	X <sub>SGN</sub>	X <sub>-1</sub>	X <sub>-2</sub>	X <sub>-3</sub>	X <sub>-4</sub>	X <sub>-5</sub>	X <sub>-6</sub>	X <sub>-7</sub>	X <sub>-8</sub>	X <sub>-9</sub>	X <sub>-10</sub>	X <sub>-11</sub>
	SGN	2 <sup>-1</sup>	2 <sup>-2</sup>	2 <sup>-3</sup>	2 <sup>-4</sup>	2 <sup>-5</sup>	2 <sup>-6</sup>	2 <sup>-7</sup>	2 <sup>-8</sup>	2 <sup>-9</sup>	2 <sup>-10</sup>	2 <sup>-11</sup>

Y INPUT	Y <sub>SGN</sub>	Y <sub>-1</sub>	Y <sub>-2</sub>	Y <sub>-3</sub>	Y <sub>-4</sub>	Y <sub>-5</sub>	Y <sub>-6</sub>	Y <sub>-7</sub>	Y <sub>-8</sub>	Y <sub>-9</sub>	Y <sub>-10</sub>	Y <sub>-11</sub>
---------	------------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	------------------	------------------

OUTPUT FORMAT WHEN NONACCUMULATING PRODUCTS (ACC = 0) FOR PINOUT DIAGRAM O<sub>n</sub> = PR<sub>n</sub>

PR SGN +4	PR SGN +3	PR SGN +2	PR SGN +1	PR 0	PR -1	PR -2	PR -3	PR -4	PR -5	PR -16	PR -17	PR -18	PR -19	PR -20	PR -21	PR -22
2 <sup>-4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>-1</sup>	2 <sup>-2</sup>	2 <sup>-3</sup>	2 <sup>-4</sup>	2 <sup>-5</sup>	2 <sup>-16</sup>	2 <sup>-17</sup>	2 <sup>-18</sup>	2 <sup>-19</sup>	2 <sup>-20</sup>	2 <sup>-21</sup>	2 <sup>-22</sup>
SGN	SGN	SGN	SGN													

NOTE 1

OUTPUT FORMAT WHEN ACCUMULATING PRODUCTS (ACC = 1) FOR PINOUT DIAGRAM O<sub>n</sub> = S<sub>n</sub>

SUM SGN +4	S +3	S +2	S +1	S 0	S -1	S -2	S -3	S -4	S -5	S -16	S -17	S -18	S -19	S -20	S -21	S -22
2 <sup>-4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>	2 <sup>-1</sup>	2 <sup>-2</sup>	2 <sup>-3</sup>	2 <sup>-4</sup>	2 <sup>-5</sup>	2 <sup>-16</sup>	2 <sup>-17</sup>	2 <sup>-18</sup>	2 <sup>-19</sup>	2 <sup>-20</sup>	2 <sup>-21</sup>	2 <sup>-22</sup>
SGN																

NOTE 2

NOTE 1. When nonaccumulating, all four MSB will indicate the sign of the product. The PR-0 term will also indicate the sign except for the one exceptional case when multiplying -1 \* -1. Note that, with the additional significant bits available on this multiplier, -1 \* -1 is a valid operation yielding a +1 product.

NOTE 2. There is no change in the format whether one is accumulating the sum of products or simply doing single products. However, the three additional most significant bits are provided to allow valid summation beyond that available for a single multiplication product. For further clarification, no difference exists between this organization and one which would have the product accumulation off chip in a separate 27-bit wide adder. Taking the sign at the most significant bit position guarantees that the largest number field will be used. In operation the sign will be extended into the lesser significant bit positions when the accumulated sum only occupies a right hand portion of the accumulator. As an example, when the sum only occupies the least three bit positions then the sign will be extended through the 24 most significant positions.

The latter factor allows one to detect imminent overflow/underflow should this be desired. Using an off chip exclusive OR gate connected to the sign and the next most significant bit will flag imminent overflow/underflow. When the two inputs are different, the exclusive OR gate goes to a logic one state. In this case four more multiply-accumulate cycles would be allowable without overflow/underflow, but a fifth could possibly cause overflow/underflow depending upon the magnitude of the sum steps.

NOTE 3. Format is shown using a 2s complement fractional notation. In this notation the location of the binary point signifying separation of the integer and fractional fields is just after the sign, between the sign and the next most significant bit for the multiplier inputs. This scheme is carried over to the output format, except that an extended significance to the integer field is provided (to extend the utility of the accumulator). Consistent with the input notation the output binary point is located between the PR-0 and PR-1 bit positions (for the nonaccumulate mode). For the accumulate mode the binary point position is the same between the S+0 and S-1 bit positions.

It is arbitrary where the binary point is considered located as long as one is consistent with both input and output formats. One can consider the number field entirely integer, i.e., with the binary point just to the right of the least significant bit for input, product, and accumulated sum.

## absolute maximum ratings over operating temperature range

Supply voltage . . . . .	-0.5 to 7.0 V
Input voltage . . . . .	.0 to 5.5 V
Output voltage . . . . .	0 to 5.5 V
Operating temperature range . . . . .	.0°C to 70°C
Storage temperature range . . . . .	-65°C to 150°C
Lead temperature (10 seconds) . . . . .	300°C
Junction temperature . . . . .	175°C

## recommended operating conditions

	TDC1003			UNIT
	MIN	NOM	MAX	
Supply voltage, $V_{CC}$	4.5	5.0	5.5	V
Clock pulse width (measured at 1.5 V level)	25			ns
Input register setup time, $T_S$ (see Figure 1)	5			ns
Input register hold time, $T_H$ (see Figure 1)	15			ns
Operating ambient temperature	0		70	°C

## electrical characteristics over recommended temperature range

PARAMETER	TEST CONDITIONS	TDC1003			UNIT
		MIN	TYP	MAX	
$V_{IH}$ High-level input voltage		2.0			V
$V_{IL}$ Low-level input voltage				0.8	V
$V_{OH}$ High-level output voltage	$V_{CC} = \text{NOM}, I_{OH} = -0.4 \text{ mA}$	2.4	2.7		V
$V_{OL}$ Low-level output voltage	$V_{CC} = \text{MIN}, I_{OL} = 4.0 \text{ mA}$		0.3	0.5	V
$I_{IH}$ High-level input current	$V_{CC} = \text{MAX}, V_{IH} = 2.4$		-2	75	$\mu\text{A}$
$I_{IL}$ Low-level input current	$V_{CC} = \text{MAX}, V_{IL} = 0.4$		-5	-75	$\mu\text{A}$
$I_{IN}$ Clocks*	$V_{CC} = \text{MAX}, V_{IH} = 2.4$			75	$\mu\text{A}$
$I_{IL}$ Clocks*	$V_{CC} = \text{MAX}, V_{IL} = 0.4$			-0.75	mA
$I_{CC}$ Supply current	$V_{CC} = \text{NOM}$		500	750	mA

At  $T_{\text{ambient}} = 25^\circ\text{C}$ ,  $V_{CC} = \text{NOM}$ .

\* Clock P is two equivalent clock input loads.

switching characteristics,  $V_{CC} = 5.0$ ,  $T_A = 25^\circ\text{C}$  (see Figure 1)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Multiply accumulate time, input register clock To output register clock, $T_{MA}$	See Figure 5		150	175	ns
Output delay $T_D$	Load 1, see Figures 3, 6		40	50	ns
Three state output delay Output enable	Load 2, see Figures 4, 6		40	50	ns
Output disable	Load 2, see Figures 4, 6		30	40	ns

12 X 12 BIT PARALLEL MULTIPLIER-ACCUMULATOR

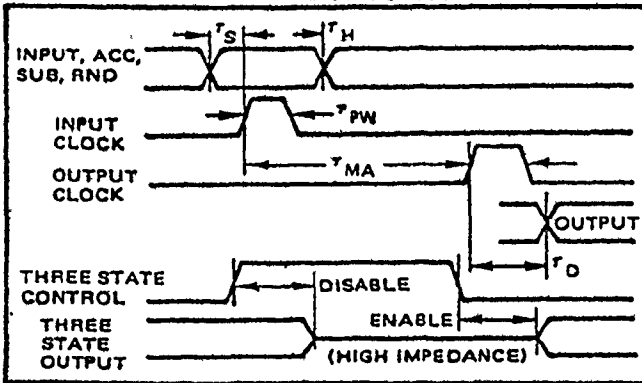


Figure 1. Timing Diagram

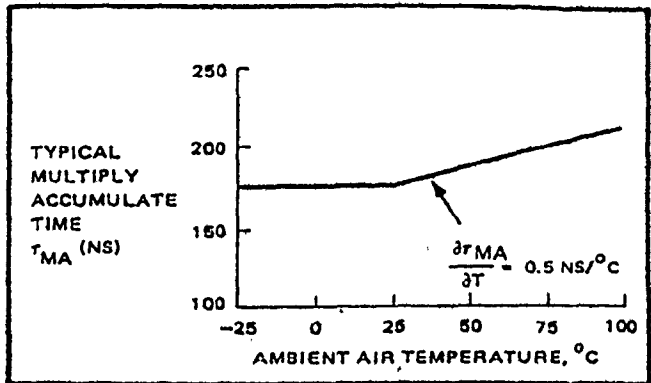


Figure 5. Multiply and Accumulate Time Versus Temperature

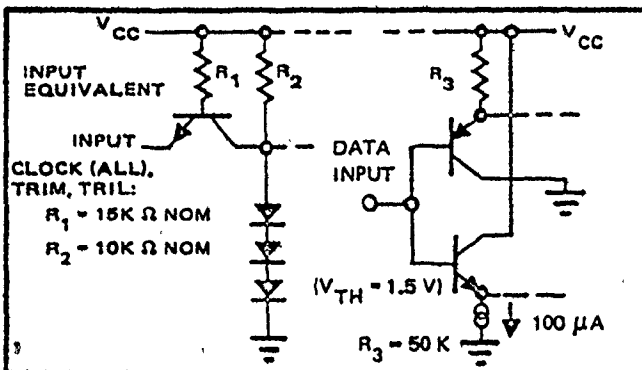


Figure 2. Input Schematics

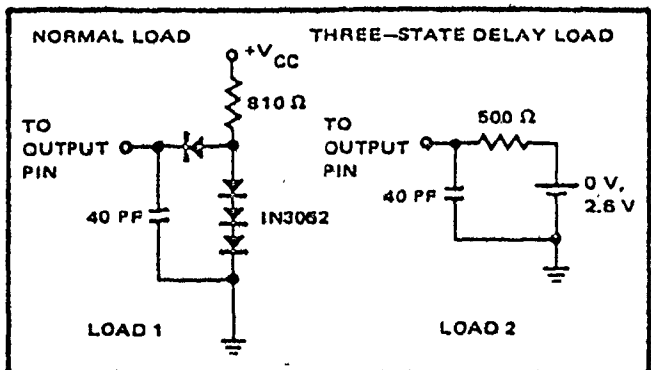


Figure 6. Test Loads for Delay Measurements

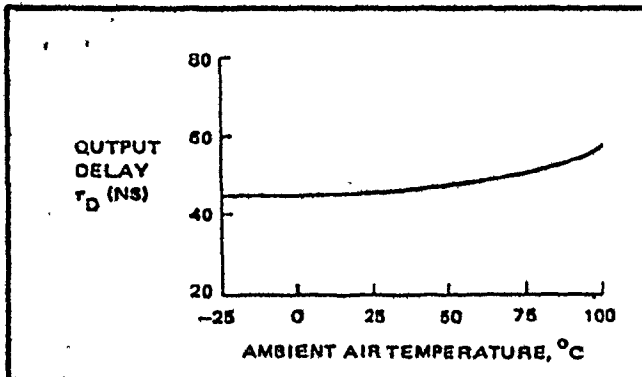


Figure 3. Output Delay Versus Temperature

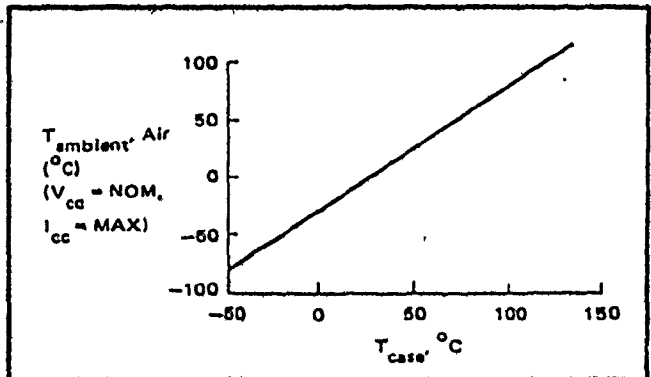


Figure 7.  $T_{\text{ambient Air}}$  Versus  $T_{\text{case}}$

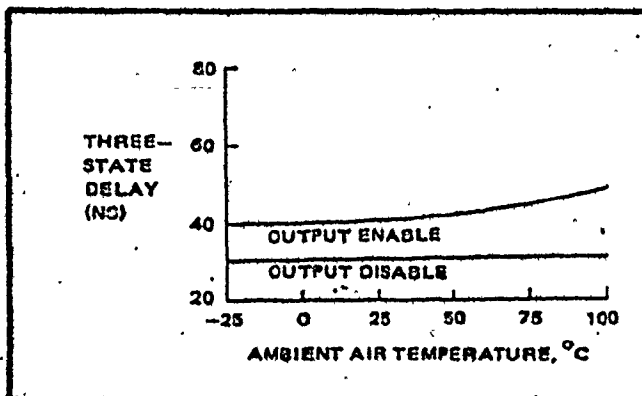


Figure 4. Three State Delay Versus Temperature

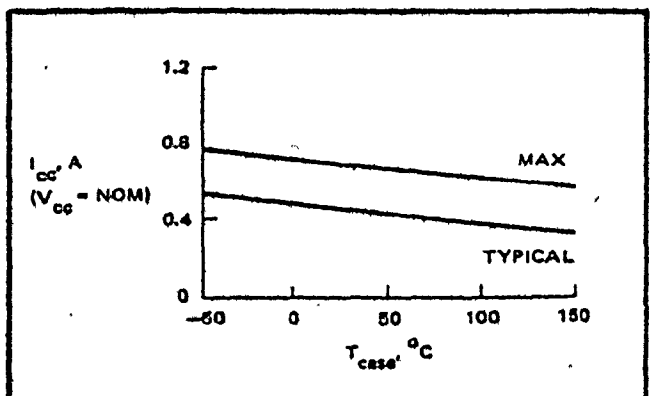


Figure 8.  $I_{cc}$  Versus  $T_{\text{case}}$

APPENDIX D

SUCCESSIVE APPROXIMATION REGISTER

DM2502, DM2503, DM2504 successive approximation registers

general description

The DM2502, DM2503 and DM2504 are 8-bit and 12-bit TTL registers designed for use in successive approximation A/D converters. These devices contain all the logic and control circuits necessary in combination with a D/A converter to perform successive approximation analog-to-digital conversions.

DM2503 and DM2504 operate over  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ ; the DM2502C, DM2503C and DM2504C operate over  $0^{\circ}\text{C}$  to  $+70^{\circ}\text{C}$ .

The DM2502 has 8 bits with serial capability and is not expandable.

The DM2503 has 8 bits and is expandable without serial capability.

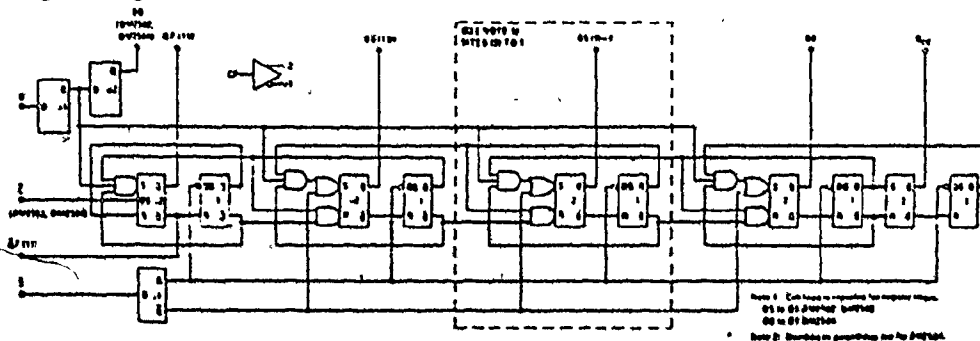
The DM2504 has 12 bits with serial capability and expandability.

All three devices are available in ceramic DIP, ceramic flatpak, and molded Epoxy-B-DIPs. The DM2502,

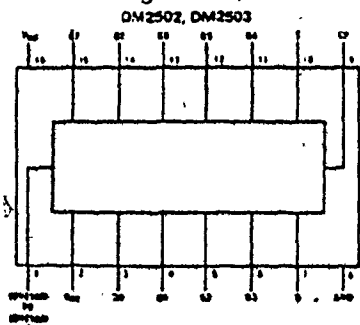
features

- Complete logic for successive approximation A/D converters
- 8-bit and 12-bit registers
- Capable of short cycle or expanded operation
- Continuous or start-stop operation
- Compatible with D/A converters using any logic code
- Active low or active high logic outputs
- Use as general purpose serial-to-parallel converter or ring counter

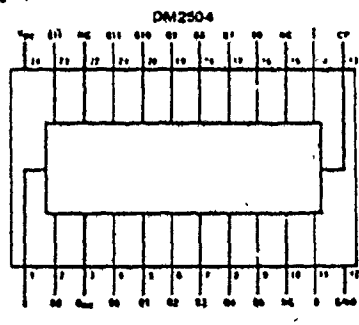
logic diagram



connection diagrams (Dual-In-Line and Flat Packages)



for info  
 Order Number DM2502L, DM2502CL, DM2503J  
 or DM2503CJ  
 See Package 17  
 Order Number DM2502CN or DM2503CN  
 See Package 23  
 Order Number DM2502W, DM2502CW, DM2503W,  
 or DM2503CW  
 See Package 41



for info  
 Order Number DM2504F or DM2504CF  
 See Package 5A  
 Order Number DM2504J or DM2504CJ  
 See Package 17A  
 Order Number DM2504CN  
 See Package 28A



APPENDIX E

IMPEDANCE CALCULATION PROGRAM

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
0800	31C220	INITIAL:	LXI SP, C220	Load stack pointer.
0803	3E08		MVI A, 08	
0805	30		SIM	
0806	3E3C		MVI A, 3C	
0808	D320		OUT 20	Set the interrupt mask
080A	3E3F		MVI A, 3F	
080C	D328		OUT 28	Assign the input - output ports.
080F	3EFF		MVI A, FF	
0810	D302		OUT 02	
0812	D303		OUT 03	
0814	3E00		MVI A, 00	
0816	D30A		OUT 0A	
0818	D30B		OUT 0B	
081A	3E00		MVI A, 00	
081C	D32A		OUT 2A	Clear R7 through R10
081E	210020		LXI H, 0020	
0821	3600		MVI M, 00	Set the pointers as shown in following table.
0823	23		INX H	
0824	3E00		MVI M, 00	
0826	210020	MAIN:	LXI H, 0020	
0829	3C00		MVI M, 00	
082B	23		INX H	

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	Pointer	COMMENT
φ82C	367φ		MVI M, 7φ		Function
φ82E	23		INX H	2φφ4	Checks First 16 samples.
φ82F	362φ		MVI, M, 2φ	2φφ1	Checks for voltage & current
φ831	23		INX H		sample for selecting proper
φ832	367φ		MVI, M, 7φ		filter routine.
φ834	23		INX H	2φφ2	check for each cycle.
φ835	362φ		MVI M, 2φ	2φφ3/	Points for successive
φ837	23		INX H	2φφ4	data storage location
φ838	362φ		MVI M, 20	2φφ5/	Points for Filter
φ83A	23		INX H	2φφ6	operation.
φ83B	3628		MVI M, 28		
φ83D	23		INX H		
φ83E	364φ		MVI M, 40	Location	Comment
φ84φ	Fb		EI	207φ/	Data storage for voltage
φ841	7C		HLT	2φB7	and current samples
φ842	Fb		EI	282φ/50	Impedance values storage
φ843	76	LOOP1:	HLT		enable interrupt.
φ844	2Aφφ2φ	LOOP2:	LHLD φφ2φ		
φ847	7d		MOV A, L		
φ848	FE2φ		CPI 2φH		
φ84A	FA2Eφ9		JM DIVIDE16		check for first 16 samples. X(n)
φ84D	2Aφ52φ		LHLD φ52φ		if X(n) < 16 ; jump to DIVIDE16
φ85φ	56		MOV, D, M		Load sample value X(n) in
φ851	23		INX H		Register pair D and E.

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
Φ852	5E		MOV E, M	
Φ853	23		INX H	increment pointer
Φ854	22 Φ52Φ		SHLD Φ52Φ	and restore the value
Φ857	23		INX H	
Φ858	23		INX H	
Φ859	46		MOV B, M	load 16th previous sample in
Φ85A	23		INX H	register pair B and C
Φ85B	4E		MOV C, M	[X(n-N)]
Φ85C	EB		XCHG	
Φ85D	48		DSUB	subtract [X(n-N)] from X(n)
Φ85E	7d		MOV A, L	
Φ85F	D3Φ1		OUT Φ1	output the result
Φ861	7c		MOV A, H	
Φ862	D3ΦΦ		OUT ΦΦ	
Φ864	2AΦ12Φ	LOOP3:	LHLD Φ12Φ	
Φ867	7d		MOV A, L	
Φ868	FEΦΦ		CPI ΦΦH	check for voltage or current
Φ86A	CAE3Φ8		JZ FLTRV	sample. If voltage jump to FLTRV
Φ86D	21Φ12Φ1		LXI H, Φ12Φ	
Φ87Φ	35		DCR M	Filter current sample.
Φ871	3EΦ2	FLTRI:	MVI A, Φ2	Input to multiplier 'b'; Pn-1'
Φ873	D329		OUT 29	
Φ875	3E+2		MVI A, 42	get b·Pn-1 product
Φ877	D329		OUT 29	

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
Φ879	3EB9		MVI A, B9	Input 'a', Qn-1
Φ87B	D329		OUT 29	
Φ87D	3ED9		MVI A, D9	get Qn-1 - bPn-1
Φ87F	D329		OUT 29	
Φ881	3EAS		MVI A, AS	
Φ883	D329		OUT 29	
Φ885	3E2F		MVI A, 2F	Output the result aQn-1 - bPn-1 to R1. Input a, Pn-1.
Φ887	D32A		OUT 2A	
Φ889	3E45		MVI A, 45	get a.Pn-1
Φ88B	D329		OUT 29	
Φ88D	3EAΦ		MVI A, AΦ	input 'b', Qn-1
Φ88F	D329		OUT 29	
Φ891	3ECP		MVI A, CΦ	get bQn-1 + aPn-1
Φ893	D329		OUT 29	
Φ895	3EAC		MVI A, A6	input 'Qn-1', Qn-1
Φ897	D329		OUT 29	
Φ899	3EBF		MVI A, 3F	output bQn-1 + aPn-1 to R2.
Φ89B	D32A		OUT 2A	
Φ89D	3E46		MVI A, 46	get Qn-1
Φ89F	D329		OUT 29	
Φ8A1	3EA8		MVI A, A8	input 'Pn-1', Pn-1
Φ8A3	D329		OUT 29	
Φ8A5	3EC8		MVI A, C8	get Pn-1 + Qn-1
Φ8A7	D329		OUT 29	

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
Φ8A9	3EE8		MVI A, E8	
Φ8AB	D329		OUT 29	
Φ8AD	3EBF		MVI A, BF	Output the result $P_{n-1} + 8n-7$ to B3
Φ8AF	D32A		OUT 2A	
Φ8B1	Φ6ΦE		MVI B, ΦE	set the counter, to loop 14 times.
Φ8B3	3EΦ1		MVI A, Φ1	for division.
Φ8B5	D329		OUT 29	
Φ8B7	3EGF		MVI A, 6F	set S low.
Φ8B9	D32A		OUT 2A	
Φ8BB	3E2F		MVI A, 2F	set S high
Φ8BD	D329		OUT 29	
Φ8BF	3E6F	DIVISION:	MVI A, 6F	Clock SAR to get output A
Φ8C1	D32A		OUT 2A	
Φ8C3	3E4F		MVI A, 4F	input to multiplier 'A', 'I2'
Φ8C5	D329		OUT 29	
Φ8C7	3E2F		MVI A, 2F	get A12
Φ8C9	D329		OUT 29	
Φ8CB	D32A		OUT 2A	
Φ8CD	Φ5		DCR B	Decrement the counter.
Φ8CF	C2BFΦ8		JNZ DIVISION	Loop 14 times.
Φ8D1	CD1Φ2Φ		CALL TEST	
Φ8D4	21Φ22Φ		LAI H Φ22Φ	
Φ8D7	34		INR M	
Φ8D8	7E		MOV A, M	

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
Φ8D9	FE11		CPI 11 H	if one cycle is complete,
Φ8DB	CA26Φ8		JZ MAIN	jump to routine MAIN.
Φ8DE	Fb		EI	Else wait until interrupt occurs
Φ8DF	76		HLT	to store voltage sample.
Φ8E1	C342Φ8		JMP LOOP1	Jump to LOOP1
Φ8E3	21Φ12Φ	FLTRV:	LXI H, Φ12Φ	Filter the voltage sample.
Φ8E6	34		INR M	
Φ8E7	3E2Φ		MVI A, 2Φ	
Φ8E9	D32A		OUT 2Φ	
Φ8EB	3EΦ2		MVI A, Φ2	Take 'b', P <sub>n-1</sub>
Φ8ED	D329		OUT 29	
Φ8EF	3E42		MVI A, 42	Get b·P <sub>n-1</sub> product
Φ8F1	D329		OUT 29	
Φ8F3	3E89		MVI A, 89	input 'a', Q <sub>n-1</sub>
Φ8F5	D329		OUT 29	
Φ8F7	3E09		MVI A, 09	Get a·Q <sub>n-1</sub> - b·P <sub>n-1</sub>
Φ8F9	D329		OUT 29	
Φ8FB	3EAS		MVI A, AS	
Φ8FA	D329		OUT 29	
Φ8FF	3E21		MVI A, 21	Output the result a·Q <sub>n-1</sub> - b·P <sub>n-1</sub>
Φ9Φ1	D32A		OUT 2A	to R4. Input 'a', P <sub>n-1</sub>
Φ9Φ3	3E45		MVI A, 45	Get a·P <sub>n-1</sub> product
Φ9Φ5	D329		OUT 29	





ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
092E	2A 05 20	DIVIDE16:	LHLD 0520	take sample in register pair
0931	56		MOV D, M	D and E.
0932	23		INX H	
0933	5E		MOV E, M	
0934	EB		XCHG	
0935	10		ARHL	move contents of D&E to H&L.
0936	10		ARHL	Rotate the register pair H&L
0937	10		ARHL	to right one bit; four times.
0938	10		ARHL	
0939	EB		XCHG	
093A	23		INX H	
093B	22 05 20		SHLD 0520	
093E	7A		MOV A, D	
093F	D3 00		OUT 00	output the result of DIVIDE16
0941	7B		MOV A, E	
0942	D3 01		OUT 01	
0944	21 00 20		LXI H, 0020	
0947	34		INR M	
0948	C3 64 08		JMP LOOP3	JUMP to LOOP3
29C8	C3 4E 09	RST 6.5:	JMP INTRPT	JUMP to routine INTRPT
				when interrupt occurs after A/D conversion is complete.

ADDRESS	DATA OR INSTRUCTION	LABEL	MNEMONIC	COMMENT
094E	2A0320	INTRPT:	LHLD 0320	Check for first sample in the cycle.
0951	7d		MOV A,L	IF it is, then store in locations
0952	FE70		CPI 70H	20B4 to 20B5 for voltage
0954	C26109		JNZ GO	
0957	21B420		LXI H,B420	
095A	CD7709		CALL DATAIN	Call data storage routine DATAIN
095D	2A0320		LHLD 0320	
0960	7d		MOV A,L	
0961	FE72	GO:	CPI 72H	if it is first current sample, store
0963	C26C09		JNZ GO1	in locations 20B6/20B7
0966	21B620		LXI H,B620	
0969	CD7709		CALL DATAIN	
096C	2A0320	GO1:	LHLD 0320	
096F	CD7709		CALL DATAIN	Call data storage routine DATAIN.
0972	20		INX H	
0973	220320		SHLD 0320	
0976	C9		RET	Return to main program.
0977	DB22	DATAIN:	IN 22	Take in Data from Post 22
0979	2F		CMA	Complement the data
097A	77		MOV M,A	store in memory location.
097B	23		INX H	
097C	DB21		IN 21	take in data from Post 21



## APPENDIX F

This program in basic is used to simulate the filter equation on graphic terminal 2647A

```

10. DIM X(50),SV(50,2),SI(50,2),Y(50),AI(50),AV(50),Z(50)
20. PLOT
   LOCATE (100,180,50,100)
   SCALE (1,50,0,16)
   FXD (2,2)
   LGRID (2,2,0,0,2,2)
   LET K = (3.14592/18)
   LET SV(1,1) = 0
   LET SI(1,1) = 0
   LET SV(1,2) = 0
   LET SI(1,2) = 0
   FOR J = 2 to 17 STEP 1
     X(J) = 1024* SIN((J-1)*K)+102*SIN((J-1)*K*3)
     Y(J) = 1024 * SIN((J-1)*K)/10
     SV(J,1) = SV((J-1),1)*.92388+SV((J-1),2)*.38268+X(J)
     SV(J,2) = -SV((J-1),1)*.38268+SV((J-1),2)*.92388
     AV(J) = SV(J,1) ^ 2 + SV(J,2) ^ 2
     AV(J) = SQR(AV(J))
     SI(J,1) = SI((J-1),1)*.92388+SI((J-1),2)*.38268+Y(J)
     SI(J,2) = -SI((J-1),1)*.38268+SI((J-1),2)*.92388
     AI(J) = SI(J,1) ^ 2+SI(J,2) ^ 2
     AI(J) = SQR(AI(J))
     Z(J) = AV(J)/AI(J)
     PLOT(J,Z(J))
     PRINT J,AV(J),AZ(J),Z(J)
   NEXT J
   FOR J=18 to 33 STEP 1
     X(J) = 1024*SIN((J-1)*K)+102*SIN((J-1)*K*3)
     Y(J) = 1024*SIN((J-1)*K)/10
     SV(J,1) = SV((J-1),1)*.92388+SV((J-1),2)*.38268+X(J)-X(J-16)
     SV(J,2) = -SV((J-1),1)*.38268+SV((J-1),2)*.92388
     SJ(J,1) = SI((J-1),1)*.92388+SI((J-1),2)*.38268+Y(J)-Y(J-16)
     SJ(J,2) = -SI((J-1),1)*.38268+SI((J-1),2)*.92388
     AV(J) = SV(J,1) ^ 2+SV(J,2) ^ 2
     AI(J) = SI(J,1) ^ 2 + SI(J,2) ^ 2
     SV(J) = SQR(AV(J))
     AI(J) = SQR(AI(J))
     Z(J) = AV(J)/AI(J)
     PLOT(J,Z(J))
     PRINT J,AV(J),AI(J),Z(J)
   NEXT J
   FOR J=34 to 50 STEP 1

```

```
X(J) = 870.4*SIN((J-1)*K)+87.04*SIN((J-1)*K*3)
Y(J) = 870.4*SIN((J-1)*K)
SV(J,1) = SV((J-1),1)*.92388+SV((J-1),2)*.382688+X(J)-X(J-16)
SV(J,2) = -SV((J-1),1)*.382688+SV((J-1),2)*.92388
SI(J,1) = SI((J-1),1)*.92388+SI((J-1),2)*.382688+Y(J)-Y(J-16)
SI(J,2) = -SI((J-1),1)*.382688+SI((J-1),2)*.92388
AV(J) = SV(J,1)2+SV(J,2)2
AI(J) = SI(J,1)2+SI(J,2)2
AV(J) = SQR(AV(J))
AI(J) = SQR(AI(J))
Z(J) = AV(J)/AI(J)
PLOT (J,Z(J))
PRINT J,AV(J),AZ(J),Z(J)
NEXT J
END
```

APPENDIX GFAULT SIMULATION

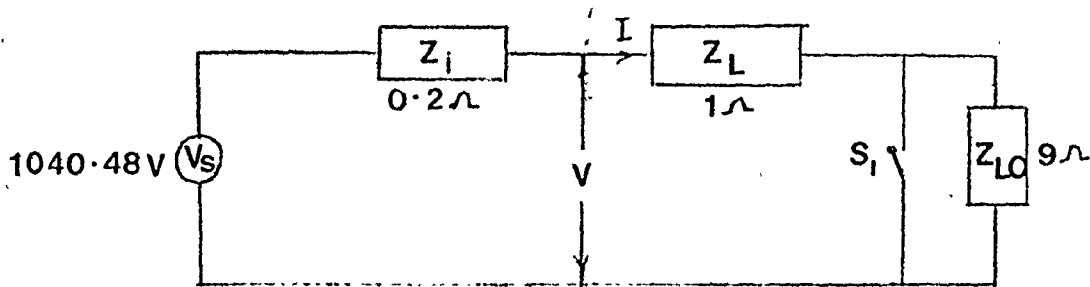
(All impedances are resistive)

Let  $Z_i$  be the source impedance

$Z_L$  be the line impedance and

$Z_{LO}$  be the load impedance and

$V_s$  be the source voltage



when switch is open

$$V = \frac{1040.48}{10.2} = 1024 \text{ volts}$$

$$I = \frac{1024}{10} = 102.4 \text{ Amp.}$$

When load is short circuited by closing switch  $S_1$ ,

$$V = \frac{1040.48}{1.2} = 870.4 \text{ volts and}$$

$$I = \frac{870.4}{1} = 870.4 \text{ Amps.}$$

Thus the voltage is reduced by a factor of 1.17 and the current is increased by a factor of 8.5.

APPENDIX H  
ERROR ANALYSIS

Let  $\epsilon$  be a quantization error of a sample coefficient or truncation error in each multiplication.

For 12 bit A/D converter,  $\epsilon = \pm 2^{-12}$

Let  $E_n$  be the error in real part  $P_n$  or  $P_{n-1}$  and

$E'_n$  be the error in imaginary part  $Q_n$  or  $Q_{n-1}$

then we get, from Eq. 2.32

$$(P_n + E_n) = (a + \epsilon)(P_{n-1} + E_n) + \epsilon + (b + \epsilon)(Q_{n-1} + E'_n) + \epsilon + X(n) + \epsilon - X(n-N) - \epsilon \quad (1)$$

and  $(Q_n + E'_n) = (a + \epsilon)(Q_{n-1} + E'_n) - (b + \epsilon)(P_{n-1} + E_n) + 2\epsilon \quad (2)$

Simplifying equations 1 and 2 and neglecting terms such as  $\pm \epsilon E_n$ ,  $\pm \epsilon E'_n$ , we obtain errors,

$$\pm E_n = \pm \epsilon P_{n-1} + a E_n + b E'_n + \epsilon Q_{n-1} + 4\epsilon \quad (3)$$

$$\pm E'_n = \pm \epsilon Q_{n-1} + a E'_n + b E_n + \epsilon P_{n-1} + 2\epsilon \quad (4)$$

Solving equations 3 and 4 for  $E'_n$  we get

$$E'_n = \frac{\pm \epsilon [(1+a)b] [Q_{n-1} + P_{n-1}] + 4\epsilon b + 2\epsilon a + 2\epsilon}{[b^2 + (1+a)^2]}$$

Taking maximum values of  $P_{n-1}$  and  $Q_{n-1}$  and taking coefficients a and b as unity for simplicity, we get

$$\pm E'_n = \pm 2.8 \times 2^{-12}$$

Substituting this value of  $E'_n$  in eq. 3 we get,

$$\pm E_n + \epsilon P_{n-1} + a E_n + b(\pm 2.8 \times 2^{-12}) + \epsilon Q_{n-1} + 4\epsilon$$

$$\pm (1+a)E_n = \pm \epsilon P_{n-1} + b(\pm 2.8 \times 2^{-12}) + \epsilon Q_{n-1} + 4\epsilon$$

$$\therefore E_n = \pm 4.4 \times 2^{-12}$$

To obtain the square of the amplitude  $S_n^2$ ,

$$S_n^2 = P_n^2 + Q_n^2$$

Let  $E_n''$  be the error in square amplitude we get,

$$\begin{aligned} S_n^2 + E_n'' &= (P_n + E_n')^2 + (Q_n + E_n')^2 + 2E_n' \\ &= \pm 16.4 \times 2^{-12} \end{aligned}$$

$\therefore$  Total error introduced in obtaining filtered output  $V^2$  and  $I^2$  is  $\pm 2 \times 16.4 \times 2^{-12}$  which is  $\pm 32.8 \times 2^{-12}$ .

Error introduced in division and square rooting is only truncation error in multiplication, as one input to multiplier ' $I^2$ ' is fixed and no error is introduced in the successive approximation register.

$\therefore$  Total error for division is  $14e$  since 14 multiplications are required.

$$\therefore \text{error in division} = \pm 14 \times 2^{-12}$$

$$\text{and error in square root} = \pm 14 \times 2^{-12}$$

$\therefore$  Total error introduced in impedance calculation is

$$\pm 32.8 \times 2^{-12} + \pm 14 \times 2^{-12} + \pm 14 \times 2^{-12} = \pm 60.8 \times 2^{-12} = \pm 0.0152$$

Taken as a percentage of maximum possible impedance value the accuracy can be represented as  $\pm 1.52\%$ .



## BIBLIOGRAPHY

- [1] G. D. Rockefeller, "Fault Detection With a Digital Computer", IEEE Transactions on Power Apparatus and Systems, Vol. PAS 88, No. 4, April, 1969.
- [2] W. Breingan, M. Chen, and T. Gallen, "The Laboratory Investigation of a Digital System for the Protection of Transmission Lines", Ibid, Vol. PAS 98, No. 2, March/April, 1979.
- [3] B. J. Mann and J. F. Morrison, "Digital Calculation of Impedance for Transmission Line Protection", Ibid, Vol. PAS 90, No. 1, Jan./Feb. 1971.
- [4] M. Ramamoorthy, G. R. Slemon and S. D. T. Robertson, "High Speed Protection of Power Systems Based on Improved Power System Model", International Conference on Large High Tension Electric Systems, 1968.
- [5] J. W. Horton, "Walsh Functions for Digital Impedance Relaying of Power-Lines", IBM Journal of Research and Development. Nov. 1976.
- [6] W. J. Smolinski, "An Algorithm for Digital Impedance Calculations Using a Single PJ Section Transmission Line Models", IEEE Transaction on Power Apparatus and Systems, Vol. PAS 98, No. 5, Sept./Oct. 1979.
- [7] L. R. Rabiner and G. Gold, "Theory and Application of Digital Signal Processing", Prentice-Hall 1975.
- [8] J. A. Howard, S. K. Mitra and B. Mahbod, "A Novel Single-Multiplier Implementation of IIR and FIR Digital Filters Using Tri-State Logic",

IEEE Transactions on Instrumentation and Measurement, Vol. IM-28,  
No. 3, Sept. 1979.

[9] Intel MCS-85 User's Manual.

[10] SDK-85 Manual.

[11] HP 2647 A User's Manual.