THE DESIGN, CONSTRUCTION, AND TESTING OF A REACTIMETER

by

Kim Allen Jones

Thesis submitted to the Graduate Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Nuclear Science and Engineering

APPROVED:

T. F. Parkinson, Chairman

P. R. Rony

R. J. Onega

October, 1977

Blacksburg, Virginia

ACKNOWLEDGMENTS

I am very pleased that I have been able to contribute to this area of research. I would like to express my gratitude to my faculty advisors; to Dr. Thomas F. Parkinson who motivated me to do my best, to Dr. Ronald J. Onega who had the patience to answer many of my questions, and to Dr. Peter R. Rony who shared his knowledge of digital electronics and microcomputers with me. A special thanks goes to the reactor personnel who constructed the electrical hardware I required.

I dedicate this to my mother and father, and , and to the seven families who treated me as one of their own; and , and , and , and , and , and , and and .

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

CHAPTER I

INTRODUCTION

A reactimeter is a useful piece of equipment to have on a nuclear reactor and is becoming widely used in the nuclear industry. The reactimeter records the neutron flux and calculates the reactivity by using a computer algorithm. Reactor operators and nuclear engineers use reactimeters for the following applications:

(1) Determination of critical control rods settings in
  - criticality experiments,
  - investigations relating to the so-called stuck rod problem
  - investigations of the symmetry properties of a core;

(2) Calibration of control rods;

(3) Determination of the reactivity equivalents of
  - fuel assemblies,
  - reflector assemblies,
  - irradiation rigs,
  - detectors;

(4) Determination o f temperature coefficient of the reactivity;

(5) Determination of the level of xenon poisoning

(6) Determination of the power output feedback coefficient;

(7) Determination of the reactivity burnup of

- fuel assemblies,
- control rods,
- active irradiation rigs;

(8) Testing the period channel.[1]

The main proposed uses of the reactimeter for the VPI & SU reactor are to monitor the change of reactivity as samples are inserted into and removed from the reactor for irradiation, for control rod calibration, and for reactor testing.

The history of the reactimeter, or reactivity meter, is relatively short. The first instruments to be called a reactivity meter started to appear in the early 1970's, even though methods to determine the neutron reactivity of nuclear reactors were around since the early 1950's. Early reactivity measurements were made using a period meter and a rate meter. From the reactor period and the change in neutron flux, the reactivity could be calculated. Today, the recommended way to measure the change in neutron population is to employ either an analog or digital method. One of the more common methods is to use an ionization chamber, whose current is converted by high gain amplifier into a voltage. The voltage signal can be converted into the reactivity of the reactor.

The current development status on the reactimeter is the use of more than one detector and the development of a Californium-252 ionization counter. A better signal is produced when more than one ionization chamber is used. By locating the ionization chambers at different places in the reactor core, the noise picked up by one detector can be eliminated by the other detectors. A Californium-252 counter is used as a correlation counter to compare the fission events in the reactor core to the fission events in the Californium. For further references, see the General Reference page following this Introduction.

The point reactor kinetics equations with six delayed neutron groups and no feedback effects are employed to calculate the reactivity from the neutron flux. This model is simple, accurate, and involves seven coupled differential equations. The prompt jump approximation is used to solve the system of equations and is valid under conditions of less than prompt critical. The prompt critical condition is reached when the reactivity inserted is equal to the delayed neutron fraction $\rho = \beta$. For conditions equal to or greater than prompt critical, delayed neutrons do not govern the reactor period, and the neutron flux increases rapidly during very short periods which are determined by the prompt neutrons [pg. 441 in (2)].

The reactimeter is comprised of a compensated ion chamber (CIC) and a microcomputer with auxiliary equipment. A Keithley Micro-microammeter moniters the CIC's signal and sends two signals to the microcomputer. One signal is a normalized analog voltage that is converted to a digital signal by a digital panel meter before being conveyed to the microcomputer. The other signal requires a special interface between the Keithley meter and the microcomputer.

The purpose of this thesis is to design and construct the hardware for the reactimeter, and to develop the software program that converts neutron flux into reactivity.

GENERAL REFERENCES

I. E. E. Gross and J. H. Marable, "Static and Dynamic Multiplication Factors and Their Relation to the Inhour Equation," Nuclear Science and Engineering, 7 (1960), 281-291.

II. S. B. Grunst, J. C. Connor, E. D. McGarry, and D. E. Conway, "The Reliability and Physical Interpretation of Measurements in a Reactivity Measurement Facility," Transactions of the American Nuclear Society, 8 (May 1965), 262-263.

III. L. Anselmi, W. Hage, H. Hettinger, H. Hohmann, and S. Kumpf, "Aspects in the Use of the Inverse Neutron Kinetics Technique," Nuclear Instruments and Methods, 98, No. 3(1972), 485-491.

IV. J. T. Mihalczo and V. K. Paie, "Theory of Correlation, Measurement in Time and Frequency Domains with Cf 252," Annals of Nuclear Energy, 2 (1975), 97-105.

V. J. B. Bullock, "Criteria for Reactivity Anomoly Monitoring in Nuclear Reactor," Transactions of the American Nuclear Society, 22 (Nov. 1975), 558-559.

VI. Tai Ping Lung and Lawrence Ruby, "Two Detector Reactivity Monitor Utilizing Ion Chambers," Transactions of the American Nuclear Society, 22 (1975), 690-691.

VII. W. R. Sheets, "Digital Period Meter," Nuclear Technology, 24 (Oct. 1974), 99-101.

CHAPTER II

THEORY

Introduction

A reactimeter converts neutron flux measurements into reactivity. Both of these quantities are generally time dependent. It is important to be able to predict the time behavior of the neutron multiplication factor, k, or reactivity, ρ, by changes in the neutron flux. To accomplish the above, the point reactor kinetics with delayed neutrons and feedback effects are used.

For the purpose at hand, the point reactor kinetics equations can be applied to the VPI & SU reactor. Some of the neutron constants are not the literature values, but are effective values that depend on the geometry and physics of the core. The number of point reactor kinetics equations depends on the number of delayed neutron groups, which range from one to six, used and the number of feedback equations required. In the most conventional form, the reactivity is difficult to determine, but by applying the prompt jump approximation the reactivity can easily be calculated.

Background

The neutron flux, ϕ, is not really a flux as one would think of the term in a physics context. Instead, it is a simple characterization of the total rate at which neutrons pass through a unit area regardless of the neutron direction [pg. 110 in (3)]. The neutron flux (neutrons/cm^2-sec) is defined as the density (neutrons/cm^3) multiplied by the average velocity (cm/sec). Reactivity, ρ, is a function of the effective neutron multiplication factor, k. Fast neutrons are produced in a fission event and usually scatter about the reactor until they are destroyed in an absorption reaction or leak out of the system. Some of the absorption is in the fissile fuel, which induces fission and produces more neutrons, thus starting a new generation of neutrons.

Suppose it was possible to measure the number of neutrons in two successive fission generations using one group diffusion theory, which considers only thermal neutrons. A ratio of the two numbers could be defined as the multiplication factor, k, characterizing the chain reaction [pg. 75 in (3)]. Let k be defined as the effective multiplication factor, that is

$$k \equiv \frac{\text{the number of neutrons in the } i^{th} \text{ generation}}{\text{the number of neutrons in the } (i-1)^{th} \text{ generation}} \tag{2.1}$$

It is more convenient to measure the ratio of the deviation of the neutron multiplication factor from unity, a quantity which is defined as the reactivity, $\rho(t)$, such that [pg. 239 in (3)],

$$\rho(t) \equiv \frac{k(t) - 1}{k(t)} . \tag{2.2}$$

Point Reactor Kinetics

In the field of nuclear reactor kinetics, a model is needed that enables one to predict the neutron reactivity of the time-dependent neutron flux. The model that is used is the point reactor kinetics model, which assumes that the reactor dynamics are position-independent and are determined by the fundamental mode of the spatial flux distribution [pg. 202 in (3)].

The point reactor kinetics equations can be derived from the one-speed diffusion equation [pp. 238-239 in (3)] or from the more sophisticated neutron transport equation.(4) In their most conventional form, the point reactor kinetics equations are as follows:

$$\frac{d\phi(t)}{dt} = [\frac{\rho(t) - \beta}{\Lambda}]\ \phi(t) + \sum_{i=1}^{6} \lambda_i\ C_i(t) \tag{2.3}$$

$$\frac{dC_i(t)}{dt} = \frac{\beta_i\ \phi(t)}{\Lambda} - \lambda_i\ C_i(t) \qquad i = 1, 6, \tag{2.4}$$

where $\phi(t)$ is the thermal neutron flux (neutrons/cm^2-sec),

$\rho(t)$ is the time dependent reactivity,

$C_i(t)$ is the neutron precursor flux of group i, (neutrons/cm^2-sec),

λ_i is the decay constant of precursor i (sec^{-1}),

β_i is the delayed neutron fraction of precursor i,

β is the total delayed neutron fraction, and

Λ is the neutron generation time (sec) [pg. 239 in (3)].

The initial conditions for equations (2.3) and (2.4) at $t \leq 0$ are,

$$\begin{aligned} \phi(0) &= \phi_0, \\ C_i(0) &= C_{i0} = \phi\ \beta_i/\Lambda\ \lambda_i, \\ \rho(0) &= \rho_0 = 0. \end{aligned} \tag{2.5}$$

When considering neutron flux, there are two types of neutrons that are of concern, prompt and delayed neutrons. Prompt neutrons are the result of a fission event and make up the majority (approximately 99.3%) of the neutron flux, have an average energy of 2 MeV, and occur within 10^{-17} seconds after a fission event. Delayed neutrons are produced by the decay of neutron precursors, which are unstable fission products. The neutron precursor decays by emitting a beta particle from the nucleus to form an emitter. In the low energy state the emitter will decay by gamma or beta emission, but in the higher energy state the emitter will decay by neutron emission. In both cases the daughter nuclide may not be stable and further gamma or beta decay may take place; however, no further neutron emission will occur (see Fig. 2.1). Delayed neutrons appear from 10^{-4} seconds to five minutes in the system after an initial fission event and have an average energy of 0.5 MeV.

In the point reactor kinetics equations, $C_i(t)$ has the same units as $\phi(t)$. When referring to neutron precursors, one usually is concerned about the density. Therefore, let $\overline{C_i}(t)$ be the i^{th} precursor density (precursor of group i/cm^3), $\overline{v_i}$(cm/sec) be the velocity of the i^{th} precursor, λ_i(sec^{-1}) be the decay constant of the i^{th} precursor, and β_i be the delayed neutron fraction of the i^{th} precursor, such that $\beta = \sum_{i=1}^{6} \beta_i$. β is the total fraction of delayed neutrons per neutron emitted in one fission, and $(1 - \beta)$ is the total number of prompt neutrons per neutron emitted. If $C_i(t)$ is to have the same units as $\phi(t)$(neutrons/cm^2/sec), then $C_i(t)$ must be equal to

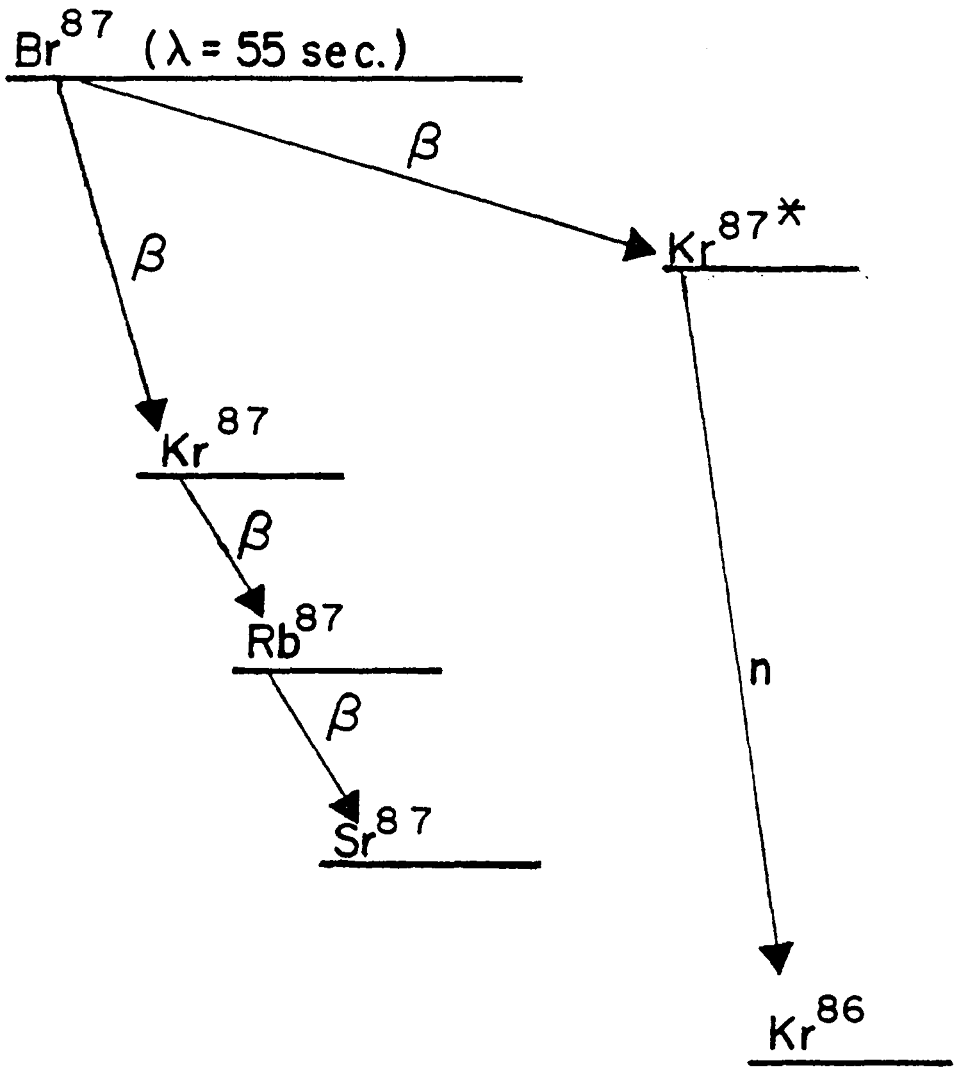


Fig. 2.1 Diagram for a Typical Neutron Precursor [pg. 13 in (7)]

$\overline{v_i} \times \overline{C_i(t)}$ [pg. 238 in (3)].

Even though the thermal neutron flux (neutrons with energies of 1 eV or less) has been discussed, the fast flux must also be considered. The multiplication factor, k, is determined by a six factor formula in which variable factors relating to the fast flux are taken into account [Chapt. 3 in (3)].

Neutron Generation Time versus Prompt Neutron Lifetime

In the point reactor kinetics equations, the neutron generation time, Λ, is used instead of the prompt neutron lifetime, ℓ. The prompt neutron lifetime is the mean time before one neutron is destroyed, and is the sum of the neutron slowing-down time and the neutron diffusion time. The slowing-down time is the time that the neutron spends above the thermal energy range after a fission event, while the diffusion time is the time that a neutron spends in the thermal range that ends when the neutron is absorbed or leaks out of the system.

If ℓ were used in equations (2.3) and (2.4) instead of Λ, Λ would be replaced by ℓ/k. The prompt neutron lifetime is based on the reciprocal probability of the destruction of the neutron.

$$\ell \equiv \frac{\text{Total neutron population in the system at time t}}{\text{Rate of neutron loss in the system}} \tag{2.6}$$

The rate of neutron loss includes neutron absorption in the fuel, neutron capture in non-fuel materials, and leakage out of the system [pg. 77 in (3)].

The neutron generation time is the mean time before one neutron generates one prompt neutron or one precursor. Λ is normalized to the fission event and is defined as, $\Lambda = 1/k$.(5) Prompt neutrons are removed by different processes and not all of these cause a new fission event; however, the production of all prompt neutrons occurs only one way, via a fission event. Also, the effect of delayed neutron precursors is expressed as a fractional production, β. It is more convenient to reference fission events than removal processes, and employ parameters based on production, ρ, β, and Λ.

Effective Delayed Neutrons Fraction

Each reactor has a characteristic effective delayed neutron fraction, β_{eff}. The actual delayed neutron fraction cannot be used because its value is too small. Delayed neutrons, when produced, have an average energy less than that of prompt neutrons, and thus slow down to thermal energies quicker. The overall effect is that there appears to be more delayed neutrons in the system than there actually are. The correction factor for a homogeneous fuel is,

$$\beta_i^* = \beta_i \exp B^2(\tau_p - \tau_i), \tag{2.7}$$

where β_i^* is the effective delayed neutron fraction of the ith group, τ_p is the Fermi age of the prompt neutrons, and τ_i is the Fermi age of the ith delayed neutron group [pg. 436 in (2)]. The uranium in the VPI & SU reactor is at least 90% or more U^{235}

in each fuel plate.[6] The Fermi age of a neutron is one-sixth the average distance squared (crow-flight) that a neutron travels, starting when it enters the system at energy E_0 and ending when thermal energy is reached at 1 eV [pg. 367 in (3)].

β_{eff} for the VPI & SU reactor is 0.00679, which can be calculated by measuring a known reactivity change over a known period and using a Reactirule sliderule to calculate β_{eff} (see Appendix A). According to equation (2.7), each effective delayed neutron group fraction should be calculated individually from its own Fermi age. Since a calculation for β_{eff} is being used, and $\beta_{eff} = \sum_{i=1}^{6} \beta_i^*$, and β is very close to β_{eff}, an approximation for β_i^* is used. For U^{235}, β_i/β are tabulated; therefore the approximation for β_i^* that is used is,

$$\beta_i^* = (\beta_i/\beta)\ \beta_{eff}. \qquad (2.8)$$

β_i^* is not exact, but is precise enough for the simple model employed. As will be shown, the values of β_i^* do not need to be calculated. In the development of the computer algorithm later in the chapter, the ratio of β_i^*/β_{eff} is used. This value reduces to β_i/β and these values are tabulated.

Prompt Jump Approximation

To understand the prompt jump approximation, one can start with the inhour equation. Using Laplace transforms to solve equations (2.3) and (2.4) simultaneously for $\phi(s)$, the following result is obtained,

$$\phi(s) = \frac{N_0[\Lambda + \sum_{i=1}^{6} \frac{\beta_i \lambda_i}{\lambda_i + s}]}{\Lambda s - \rho_0 + \beta - \sum_{i=1}^{6} \frac{\lambda_i \beta_i}{s + \lambda_i}} . \qquad (2.9)$$

The denominator can be shown to have seven distinct real roots, which implies $\phi(s)$ has seven poles on the real s-axis [pg. 21 in (7)]. The inverse transform of equation (2.9) is

$$\phi(t) = \sum_{i=1}^{7} A_j \exp(\omega_j t), \qquad (2.10)$$

where ω_j ($\sec^{-1}$) are the seven roots of the denominator of $\phi(s)$ for $s = j$. Setting the denominator of equation (2.9) equal to zero, the inhour equation becomes

$$\rho_0 = \beta + \Lambda \omega - \sum_{i=1}^{6} \frac{\beta_i \lambda_i}{\omega + \lambda_i} . \qquad (2.11)$$

Since $\beta = \sum_{i=1}^{6} \beta_i$, equation (2.11) becomes

$$\rho_0 = \Lambda \omega + \sum_{i=1}^{6} \frac{\beta_i \omega}{\omega + \lambda_i} , \qquad (2.12)$$

which is the most conventional form of the inhour equation [pg. 22 in (7)]. Six of the seven roots are based on six delayed neutron groups, and the seventh is primarily determined by the generation time and reactivity. ω_1 through ω_6 refer to the delayed neutron groups with ω_1 representing the delayed neutron groups which has the largest ω and ω_6 representing the delayed neutron group with the smallest ω.

ω_7 represents the prompt neutrons and is the smallest of all the ω roots.

The effect of ω can be seen in Figure 2.2. During time intervals of two seconds or less, $\phi(t)$ has the shape of the term, $A_7 \exp(\omega_7 t)$. For longer periods of time $\phi(t)$ follows the sum of the six delayed neutron group terms. Each term of the sum, $A_j \exp(\omega_j t)$, for i = 1 to 6, has a similar exponential shaped curve. During a step change in reactivity, the reactor flux has a very rapid transient behavior initially that is characteristic of the prompt neutron lifetime, and is followed by a slower transient behavior that is controlled by the delayed neutron groups. The time behavior of the neutron flux is essentially governed by the delayed neutron flux for systems below prompt critical [pg. 250 in (3)].

The prompt jump approximation makes use of the above fact for systems below prompt critical. The prompt neutron lifetime is assumed to be zero so that for a step reactivity insertion, the neutron flux level jumps from ϕ_0 to ϕ_1 (Fig. 2.2) instantaneously. The effect on the point reactor kinetics equations (2.3) and (2.4) is to neglect the time derivative, $\frac{d\phi}{dt}$, by setting it equal to zero. Delayed neutron production cannot change instantaneously during a step change. The prompt jump approximation predicts a reactivity jump that yields an instantaneous change in neutron flux from ϕ_0 to ϕ_1 given by $\phi_1(\beta - \rho_1) = \phi_0(\beta - \rho_0)$ [pg. 251 in (3)]. The prompt jump approximation is very useful and accurate for reactor systems below prompt critical. It is a good approximation for

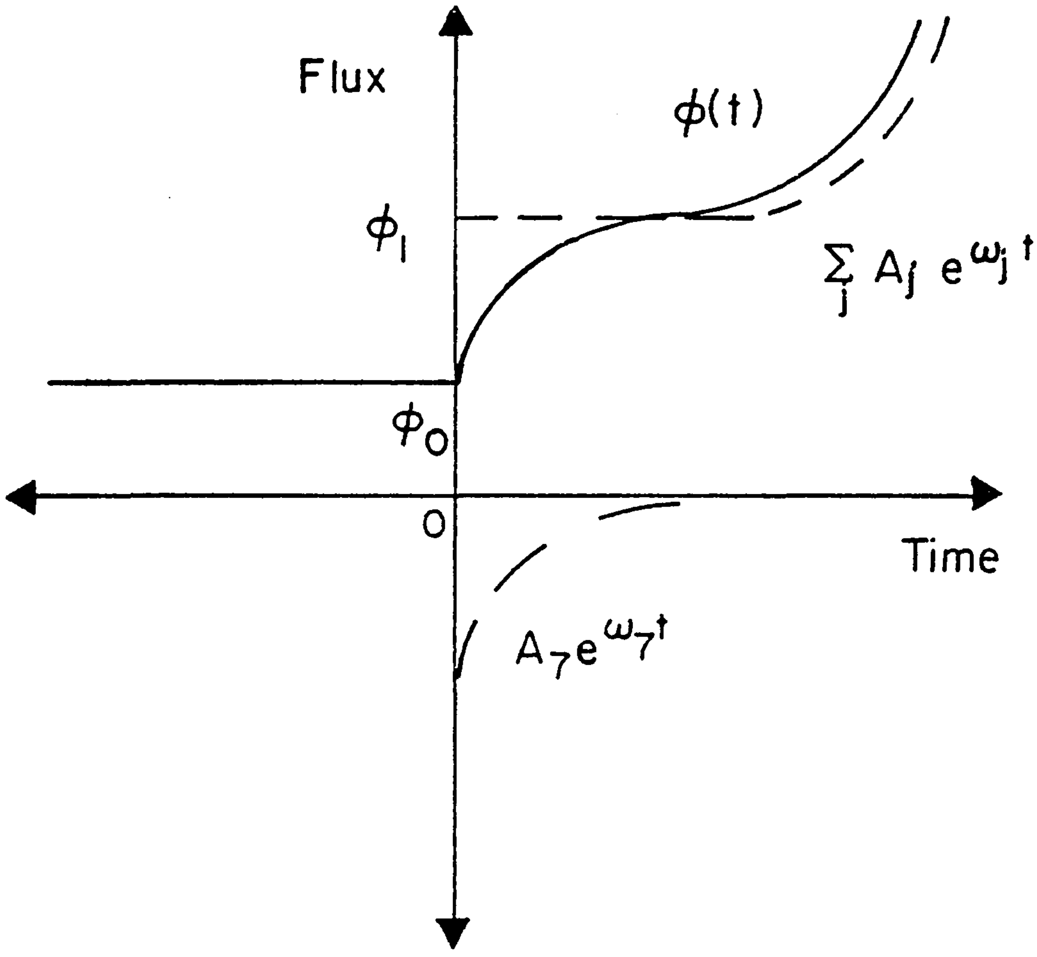


Fig. 2.2 Neutron Flux Change for a Positive Reactivity Insertion. [pg. 34 in (7)]

injection into and removal of samples from the reactor at criticality.

Feedback Effects

Feedback effects are changes in the physical and chemical properties of the reactor materials that cause changes in the neutron flux. To illustrate the above, the moderator temperature is a feedback effect. As the neutron flux increases, the moderator temperature increases and causes the neutron flux to decrease. Other feedback effects are fuel temperature and the build-up of neutron poisons, such as xenon, iodine, and samarium. No feedback equations are required in the algorithm, because the changes in the neutron flux occur internally in the reactor.

Six Delayed Neutron Groups

There are six delayed neutron groups that have distinct half lives and decay times (Table 2.1 and Table 2.2). The six groups can be collapsed into fewer groups and new decay constants can be calculated for each group (Table 2.3) by the equation [pg. 242 in (3)],

$$\frac{1}{\lambda} = \frac{1}{\sum_i \beta_i} \left(\sum_i \frac{\beta_i}{\lambda_i} \right). \tag{2.13}$$

The purpose for combining neutron groups would be to reduce the number of differential equations from seven to no less than two. The fewer delayed neutron groups used in the point reactor kinetics equations, the less accurate are the results. As can be seen from

Table 2.1, the delayed neutron groups are divided into the fewest groups possible to have good accuracy. According to the natural logs of the half lives, the difference between the number in any group is no more than 0.5, while the difference between groups is one.

Reducing the seven equation system to make the model simpler would decrease the accuracy of the results. The evaluation of each of the six delayed neutron groups represents the same type of calculation, with only the constants being different. A microcomputer can easily perform all six calculations in under one-half second. The time to perform the six calculations is not a major factor, so nothing is gained from a simplification from six groups to something less, and accuracy is lost.

Computer Algorithm Equations:

A simplification of equations (2.3) and (2.4) is needed before they can be used in the computer algorithm. Consider the constants first. Let $\alpha_R = \beta/\Lambda$ and $\beta_i/\beta = a_i$, where a_i is the relative neutron fraction. Reactivity is measured in dollars. When the reactivity change is equal to beta, $\rho(t) = \beta$, this quantity is called a dollar; therefore let $\rho'(t) = \rho(t)/\beta$. Finally, the precursor flux is defined as $Y_i(t) = C_i(t)\Lambda\ \lambda_i/\beta_i$. If one multiplies $\rho(t)$ by β/β and $\lambda_i C_i(t)$ by $\Lambda\beta_i\beta/\Lambda\beta_i\beta$ in equation (2.3) and everything in equation (2.4) by $\Lambda\ \lambda_i/\beta_i$ and simplifies, (2.3) and (2.4) become

$$\frac{d\phi}{dt} = \alpha_R[\phi(t)(\rho'(t) - 1) + \sum_{i=1}^{6} a_i Y_i(t)], \qquad (2.14)$$

Table 2.1

Delayed Neutron Precursors

Group Number	Precursor	Half-life second	ln $t_{\frac{1}{2}}$/sec
1	Br^{87}	54.5	3.99
2	I^{137}	24.4	3.19
	Br^{88}	16.3	2.79
3	$Br^{(89)}$	6.3	1.84
	$Rb^{(93,94)}$	~6.	1.79
4	I^{139}	2.0	.69
	(Cs, Sb, or Te)	(1.6-2.4)	.47 - .88
	$Br^{(90,91)}$	1.6	.47
	$Kr^{(93)}$	~1.5	.41
5	(I^{140} Kr ?)	0.5	-.69
6	(Br, Rb, As ?)	0.2	-1.61

*Uncertain quantities are indicated by parentheses. [pg. 99 in (2)]

Table 2.2

U^{235} Delayed Neutron Data

Group	Decay Constant (sec^{-1})	Fractional Yield β_i
1	0.0127 ± 0.0002	0.000247
2	0.0317 ± 0.0008	0.001385
3	0.115 ± 0.003	0.001222
4	0.311 ± 0.008	0.002645
5	1.40 ± 0.081	0.000832
6	3.87 ± 0.369	0.000169

$$\beta = \sum_{i=1}^{6} \beta_i = 0.0065 \pm 0.0002$$

[pg. 22 in (8)]

Table 2.3

U^{235} Delayed Neutron Decay Constants

for One, Two, and Three Groups

1 group	$\lambda = .102\ \sec^{-1}$
2 groups	$\lambda_1 = .0387\ \sec^{-1}$
	$\lambda_2 = .399\ \sec^{-1}$
3 groups	$\lambda_1 = .0285\ \sec^{-1}$
	$\lambda_2 = .202\ \sec^{-1}$
	$\lambda_3 = 1.57\ \sec^{-1}$

$$\frac{dY_i}{dt}(t) = \lambda_i(\phi(t) - Y_i(t)), \qquad i = 1 \text{ to } 6. \tag{2.15}$$

The above equations have the following initial conditions for $t \leq 0$:

$$\begin{aligned} \phi(0) &= \phi_0, \\ Y_i(0) &= Y_{i0} = \phi_0, \\ \rho'(0) &= \rho'_0 = 0. \end{aligned} \tag{2.16}$$

Dividing through by α_R, applying the prompt jump approximation, and then solving for $\rho(t)$ in equation (2.14), one obtains

$$\rho'(t) = \frac{\phi(t) - \sum_{i=1}^{6} a_i Y_i(t)}{\phi(t)} . \tag{2.17}$$

Everything but $Y_i(t)$ is known in equation (2.17). $Y_i(t)$ can be determined by solving the simple differential equation (2.15). Thus,

$$Y_i(t) = \phi_0 \exp(-\lambda_i\, t) + \lambda_i \int_0^t \phi(\tau) \exp(-\lambda_i(t - \tau)\, d\tau. \tag{2.18}$$

The integral can be evaluated by a numerical integration technique, such as Simpson's Rule or the Trapezoid Rule. Equations (2.17) and (2.18) are the equations on which the microcomputer algorithm is based [pp. 132-133 in (9)].

Conclusion:

Starting with the basic form of the point reactor kinetics equations, seven equations have been derived for use in the computer algorithm. Six delayed neutron groups are used for accuracy. Since the system is used to calculate the reactivity below prompt critical, the prompt jump approximation is employed. This model is simple and accurate to within two decimal places and is implemented in the software program.

CHAPTER III

EQUIPMENT

Introduction

The reactimeter for the VPI & SU reactor is comprised of four components: a compensated ion chamber, a micro-microammeter, a digital panel meter, and a microcomputer (see Fig. 3.1 for a block diagram).

The Westinghouse Type 6377 compensated ion chamber measures the thermal neutron flux from the reactor and outputs an electrical current that is directly proportional to the thermal neutron flux. The current is measured by a Keithley Model 411 micro-microammeter. The current range of the Keithley is from 10^{-11} to 10^{-3} amperes and is divided into seventeen logarithmic range settings. The Keithley has a normalized analog voltage signal for each range that determines the magnitude of the current in that range setting. This signal is converted into a digital signal by a digital panel meter, which is made by Analog Devices, Inc., before being read by the microcomputer. A second signal is required to interpret the range of the Keithley and relay such information to the microcomputer. The microcomputer is a Mark 80 that is made by E&L Instruments, Inc.

Compensated Ion Chamber:

The compensated ion chamber (CIC) is a Westinghouse Type 6377 and is designed to detect thermal neutron fluxes from 2.5×10^{2} to 2.5×10^{10} neutrons/cm^{2}-sec, in fields where very high gamma radiation

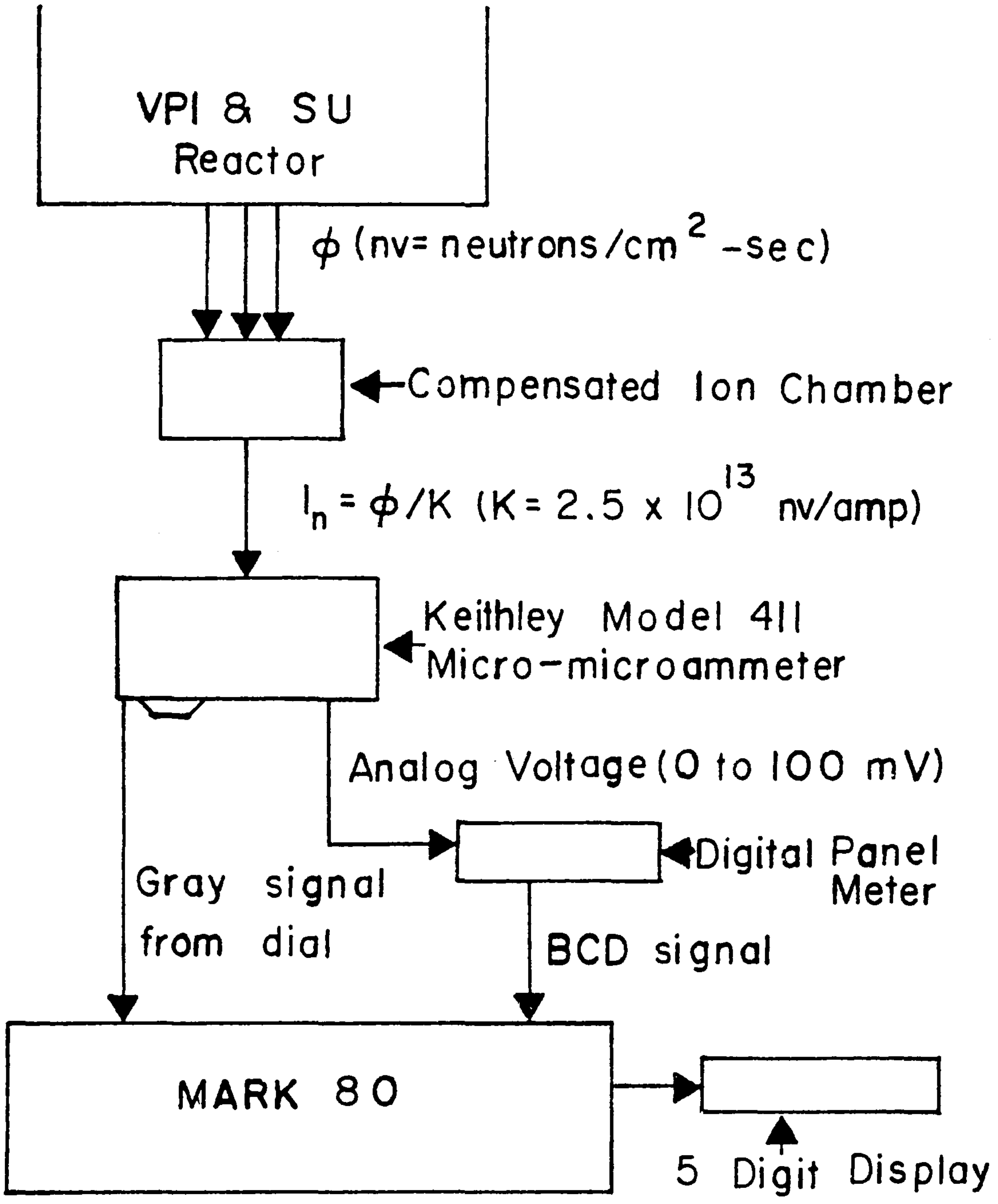

VPI & SU Reactor
ϕ (nv= neutrons/cm^2 -sec)
Compensated Ion Chamber
$I_n = \phi/K$ (K= 2.5 x 10^{13} nv/amp)
Keithley Model 411 Micro-microammeter
Analog Voltage (0 to 100 mV)
Digital Panel Meter
Gray signal from dial
BCD signal
MARK 80
5 Digit Display

Fig. 3.1 Block Diagram of Reactimeter

is present. The CIC can be operated in any position at any temperature up to 175°F. It has an operating voltage range from 300 to 800 volts that is set to 425.3 volts DC, and a compensating voltage that is set to -43.2 volts DC. The current output is directly proportional to the thermal neutron flux, $I_n = K \times \phi$, where $K = 2.5 \times 10^{13}$ nv/amperes (nv = neutrons/cm^2/sec).

A compensated ion chamber records neutron flux by using two regions of equal volume and three parallel plates (see Fig. 3.2), which may be cup shaped for greater surface area. One region has a boron coating, enriched in boron-10, covering the inside faces of the plates and has a positive potential on the outside plate. The boron-10 has a high capture cross section for thermal neutrons, and emits an alpha particle after absorption of a neutron: $^{10}B(^{1}n,\ ^{4}\alpha)^{7}Li$. The alpha particle along with the gamma radiation produces a current between the plates. The other region has a negative potential on the outside plate and has a current due only to gamma radiation in the direction opposite to that in the other region. The middle plate is the signal collector and is grounded. The middle plate has a boron coating on the face in the region with the positive potential, but has none on the face in the region with the negative potential. Since the gamma rays are approximately the same in both regions, the final current output of the middle plate is due only to the thermal neutrons. Thus, the current output from the CIC is directly proportional to the thermal neutron flux [pp. 312-313 in (10)]. Specifications for the Westinghouse Type 6377 CIC are located in

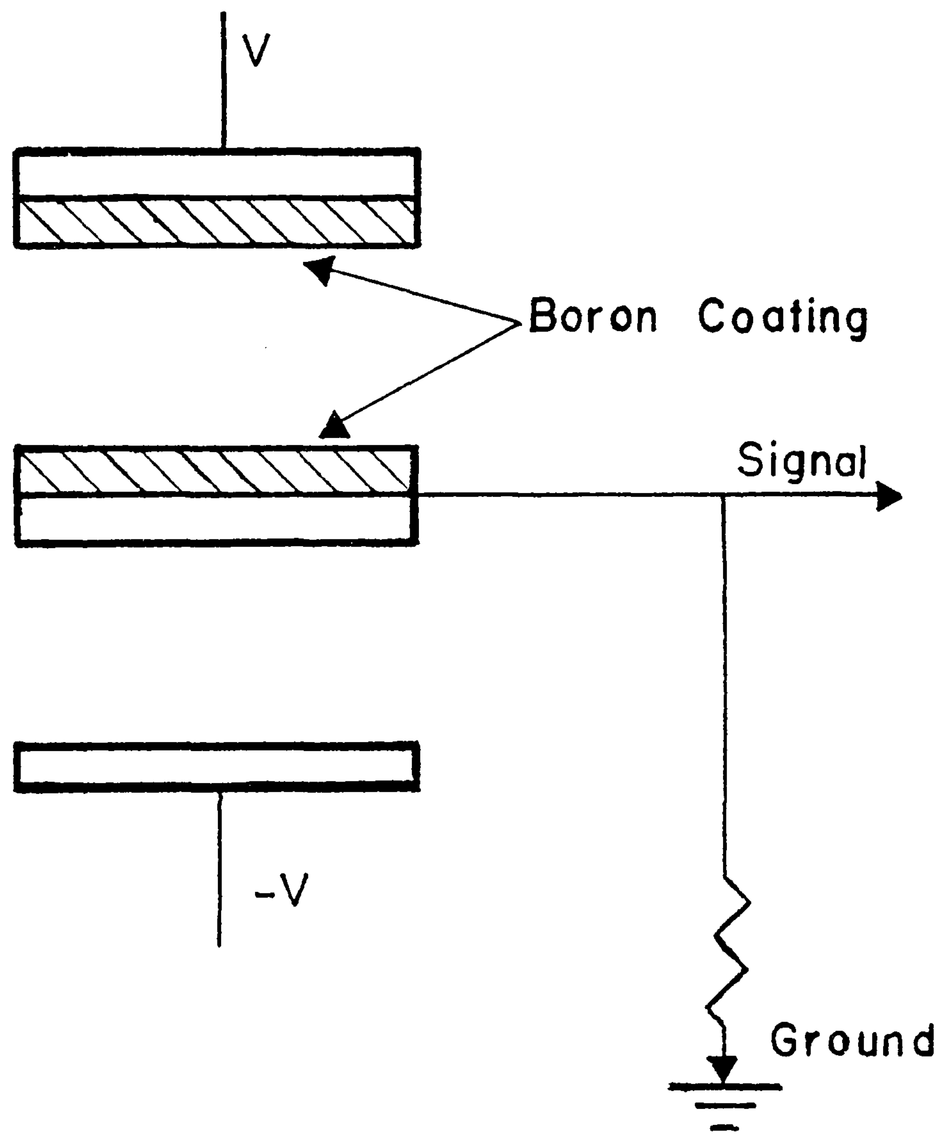


Fig. 3.2 Compensated Ion Chamber Detector [pg. 313 in (10)]

Appendix C.

Keithley Model 411 Micro-microammeter

The Keithley Model 411 Micro-microammeter has seventeen current ranges from 10^{-11} to 10^{-3} amperes. The Keithley meter uses a resistor network to partition the current into smaller logarithmic decades as follows: 1×10^{-11}, 3×10^{-11}, 1×10^{-10}, 3×10^{-10}, 1×10^{-9}, ... , 3×10^{-4}, and 1×10^{-3} amperes. The lowest range setting, 1×10^{-11}, is not used because values of neutron flux at this setting are of no concern. There is a ± 2% accuracy for the ranges from 1×10^{-3} to 3×10^{-7} and a ± 4% accuracy for all other ranges. The input current to the Keithley meter is the output current from the CIC, which can be read from a meter on the front panel. The Keithley meter has a 0 to 100 mV output on the back panel from which a normalized signal is produces for each of the sixteen ranges. The normalized voltage is an analog signal that must be converted to a digital signal before it can be used by the microcomputer. The mantissa of the electrical current is determined from this signal, which represents a number either between one and three, or a number between three and ten (see Table 3.1). Since there is no method in the manufacturer's design to determine the actual range setting, such as, by using an analog or digital signal from which the limits of the mantissa and the magnitude of the characteristic can be determined, a method must be devised. This limitation poses an interesting interfacing problem. Specifications for the Keithley meter are

Table 3.1

Current Measured vs Voltage Output

Correlation between the current read by the Keithley meter and the normalized voltage output.

Range Setting	Current Range Limits		Current Measured (amps)	Voltage Output
	Low	High		
3×10^{-10}	1×10^{-10}	3×10^{-10}	2.0×10^{-10}	50.0
			1.5×10^{-10}	25.0
			2.5×10^{-10}	75.0
1×10^{-9}	3×10^{-10}	1×10^{-9}	4.0×10^{-10}	14.3
			6.5×10^{-10}	50.0
			8.0×10^{-10}	71.5
3×10^{-9}	1×10^{-9}	3×10^{-9}	2.0×10^{-9}	50.0
			1.1×10^{-9}	5.0
			1.6×10^{-9}	30.0
1×10^{-5}	3×10^{-6}	1×10^{-5}	5.5×10^{-6}	35.7
			6.5×10^{-6}	50.0
			7.5×10^{-6}	64.3
3×10^{-5}	1×10^{-5}	3×10^{-5}	1.0×10^{-5}	0.0
			1.5×10^{-5}	25.0
			2.5×10^{-5}	75.0
			3.0×10^{-5}	100.0

located in Appendix C.

Digital Panel Meter

The digital panel meter (DPM) is a 3 1/2-Digit AC line-powered DPM Model AD2009/S made by Analog Devices, Inc. The AD2009/S is designed around TTl logic circuits and is TTL/DTL (transistor-transistor logic/diode-transistor-logic) compatible. It is capable of analog input voltages from 0 to 200 mV and outputs a parallel binary-coded-decimal (BCD) digital signal with an accuracy of ± 0.1% reading, ± 1 digit. Under external control, the AD2009/S can be triggered to give readings up to 100 conversions per second, but under internal control has a conversion rate of six conversions per second. All thirteen digital output lines are valid when the status lines are low. Analog Devices, Inc. documentation is located in Appendix C.

Mark 80 Microcomputer

The microcomputer is a Mark 80^R which is made by E&L Instruments, Inc. and is used in conjunction with IF-101 interface board, also made by E&L Instruments Inc. The heart of the Mark 80 is a CPI-80/B Central Processor and Interface Controller printed circuit board which contains the Intel 8080 microprocessor chip. The minimum memory requirements for the Mark 80 is 1 K of R/W (Read/Write) memory, that can be obtained from MB-80/B memory circuit board. The Mark 80^R can be expanded up to 64 K of memory by using sixteen memory boards and a chassis rack. The Mark 80 has some unique features and concepts in data busing and I/O (input/output).(11) See Bugbook III, reference 11 for programming and basic interface techniques.

CHAPTER IV

HARDWARE

Introduction

The purpose of the hardware in the reactimeter is to allow for an exchange of information between the Mark 80 and the peripheral devices. In total there are five external circuits connected to the Mark 80. There are two sources of input data that together represent one piece of data. The digital panel meter interface allows a three-digit mantissa to be read into the Mark 80, where the Keithley interface supplies the characteristic of the mantissa. The third circuit, the output interface, receives a number, consisting from one to five digits plus the sign, that represents the final calculated result. A clock circuit, the fourth external circuit, is required to generate an interrupt to the Mark 80 every 0.2 seconds to restart it for a new calculation. The final circuit is one that automatically single steps the Mark 80 at a speed of 500 kHz instead of the normal operating speed of 2 MHz.

Keithley Micro-microammeter Interface

The Keithley interface is the more complex of the two data input interfaces. The interface involves the transmission of a binary signal from the range dial of the Keithley to the Mark 80. The first problem that arises is that there is no signal of any kind generated by the range dial in the manufacturers specifications.

The method used to create a signal is the construction of a Gray shaft encoder that uses the Gray code to generate a binary signal for each range setting.

In the Grade code, which is a binary code, only one bit changes at a time when the switch changes between consecutive positions. This allows for the least amount of error in interpreting the binary signal. The Gray code, with its binary representation and binary code equivalent value, is listed in Table 4.1 [pg. 218 in (12)].

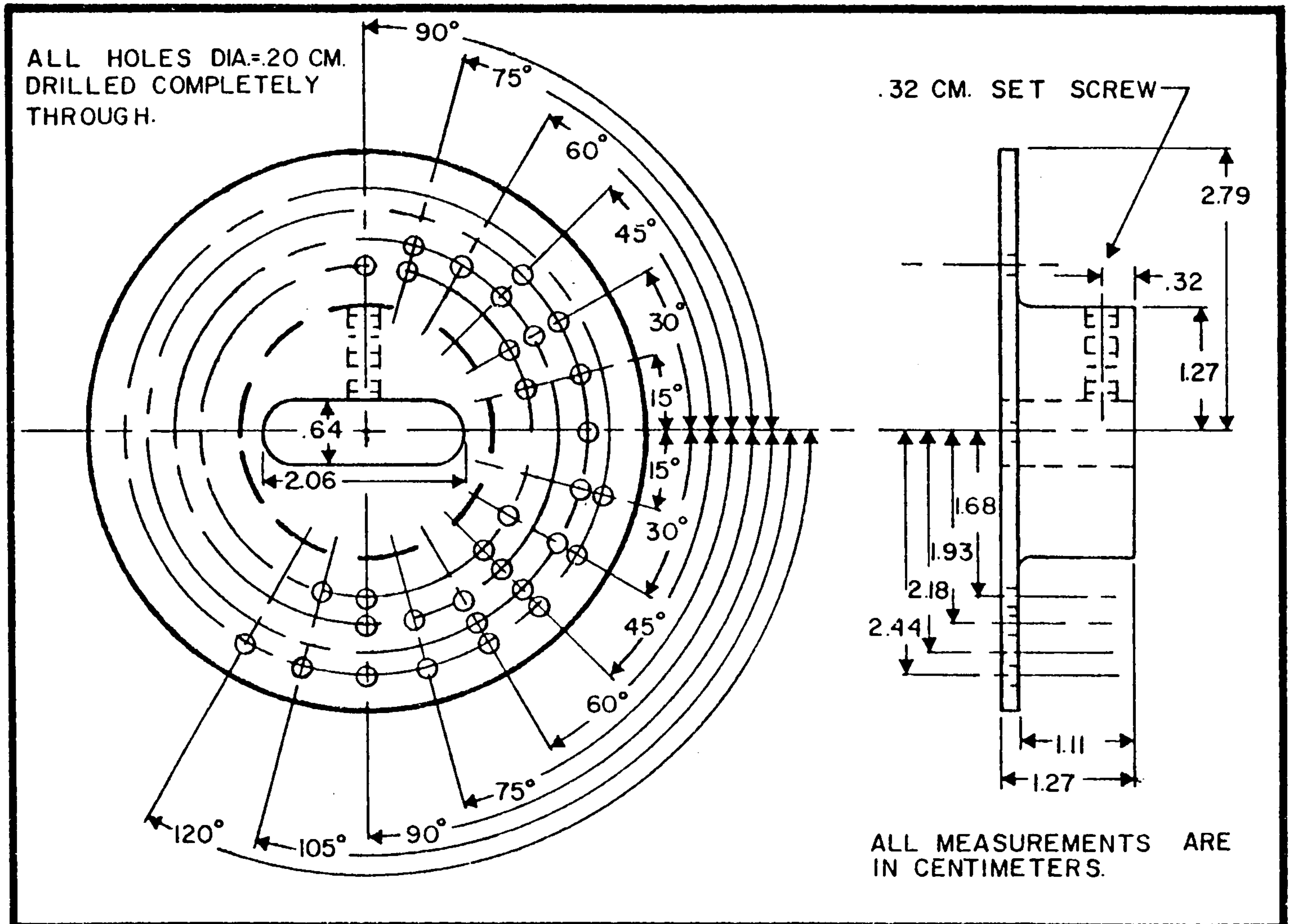
A Gray shaft encoder uses the Grady code to assign a binary value to each range of the Keithley meter. Four bits can represent the sixteen ranges. The Gray shaft encoder was manufactured according to the drawing in Figure 4.1. As can be noticed, there are 0.20-cm holes drilled around the encoder disk at different radii. The binary code, generated by light passing through the holes lying on a single radius, is detected by a phototransistor array. The Gray shaft encoder is fastened to the range dial shaft on the inside of the Keithley meter. Each radius is assigned a binary weight. The holes lying on the circles with radii 1.68, 1.93, 2.18, and 2.44 cm represent bits DATA0, DATA1, DATA2, and DATA3, respectively.

The light detector is a FTK0040 9-element NPN planar phototransistor array that has a high illumination sensitivity. Each phototransistor channel is electrically isolated, is on a 0.254 cm center, and requires a Schmitt trigger for operation (see documentation in Appendix C). A Schmitt trigger is a hybrid analog/digital device in which the output pulse of the trigger remains at a constant

Table 4.1

Gray Code

Position	Gray Code	Binary Equivalent Value
0	0000	0
1	0001	1
2	0011	3
3	0010	2
4	0110	6
5	0111	7
6	0101	5
7	0100	4
8	1100	12
9	1101	13
10	1111	15
11	1110	14
12	1010	10
13	1011	11
14	1001	9
15	1000	8



NOT TO SCALE	FIG. 4.1	GRAY SHAFT ENCODER	KIM JONES	7-21-77

amplitude as long as the input voltage exceeds a certain DC voltage value. A resistor is needed on each channel to adjust the light sensitivity. It was determined that a 10 K ohm resistor was required to permit each phototransistor channel to be used with a G.E. #47 6.3 V light bulb. Only four of the nine channels are required for the binary code. The remaining five are left unconnected (see Fig. 4.2 for circuit connection). The arrangement of the 6.3 V light bulb, the Gray shaft encoder, and the phototransistor is shown in Figure 4.3. Each channel is connected to an input of a 7475 flip-flop, which latches the value (logic 1 or 0) of the four channels on a positive device select pulse generated by the Mark 80. The data is read into the Mark 80 when a negative device select pulse is applied to the corresponding 8095 three state-buffer. The 8095 three-state buffer has three outputs, a "logic 0" state, a "logic 1" state, and a state in which the output is, in effect, disconnected from the rest of the circuit and has no influence on it. The negative device select pulse is used to enable the 8095 buffer and allows the information in the 7475 flip-flop to be transferred on to the data bus of the Mark 80 and be read into the accumulator (see Fig. 4.4 for interface connection). The 8095 chip output pins are connected to the data bus lines D0 through D3. Data bus lines D4 through D7 are set to "logic 0".

Digital Panel Meter Interface

The interface between the digital panel meter and the Mark 80

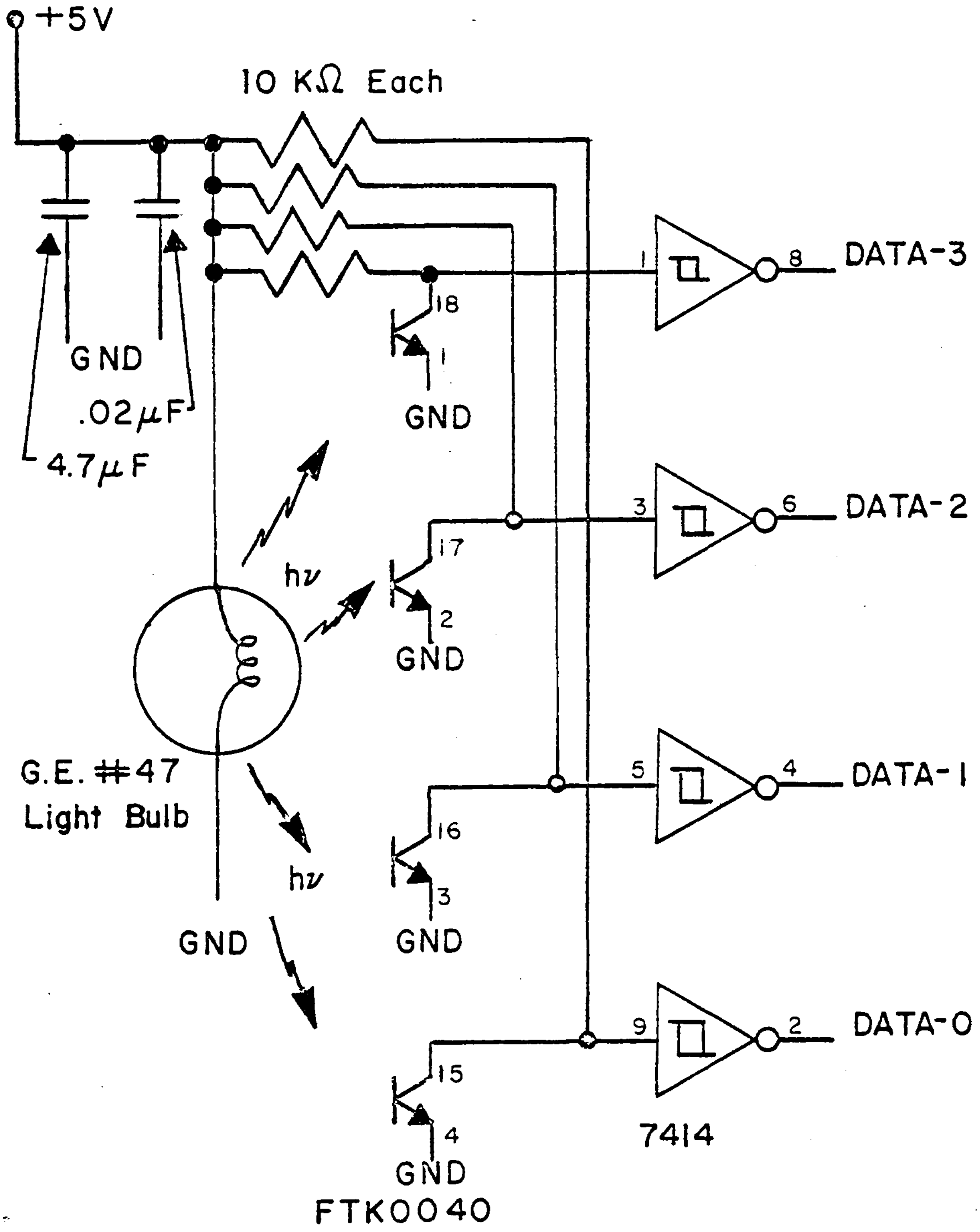


Fig. 4.2 Gray Shaft Encoder Circuit

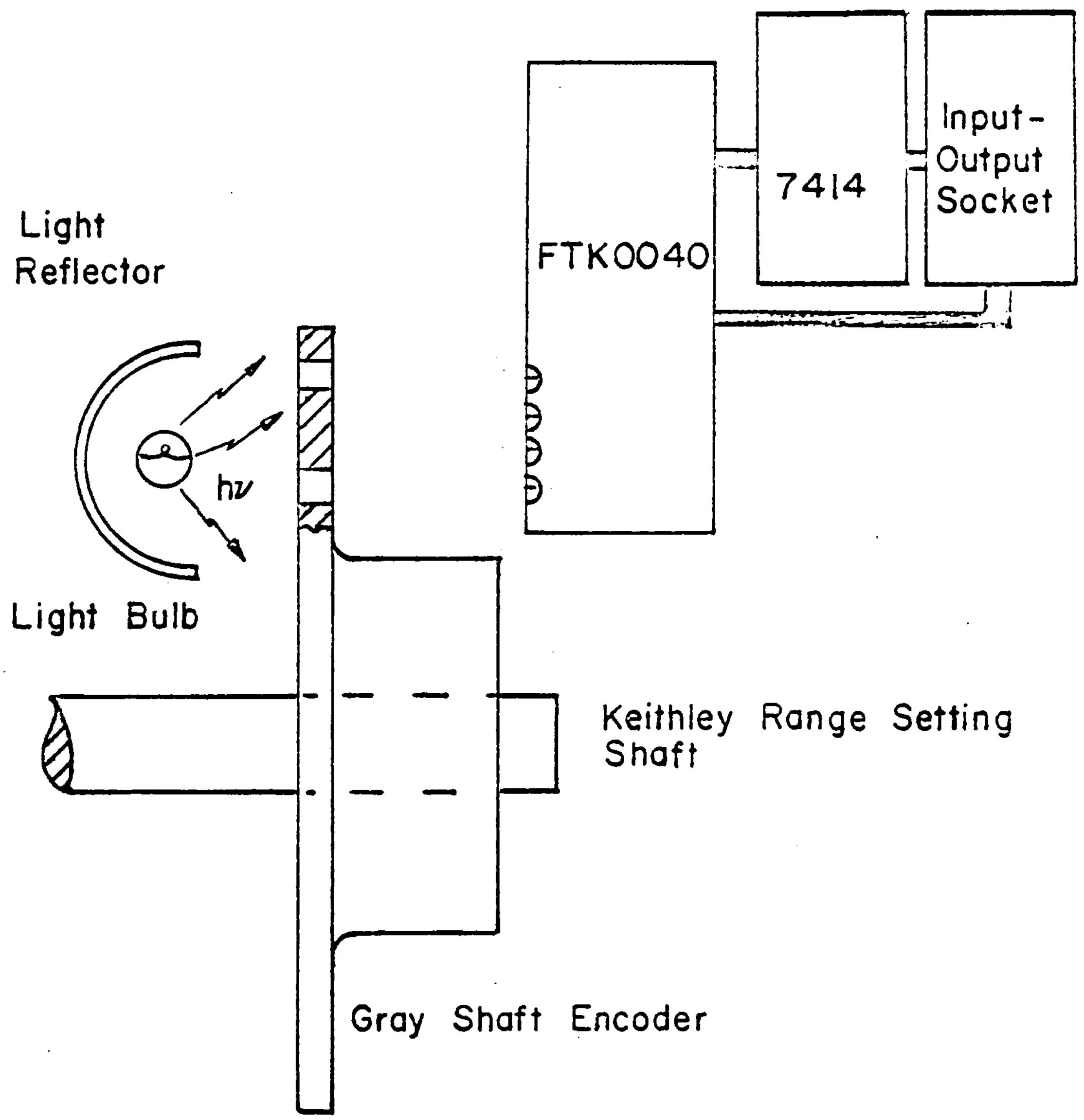


Fig. 4.3 Block Diagram of Gray Shaft Encoder

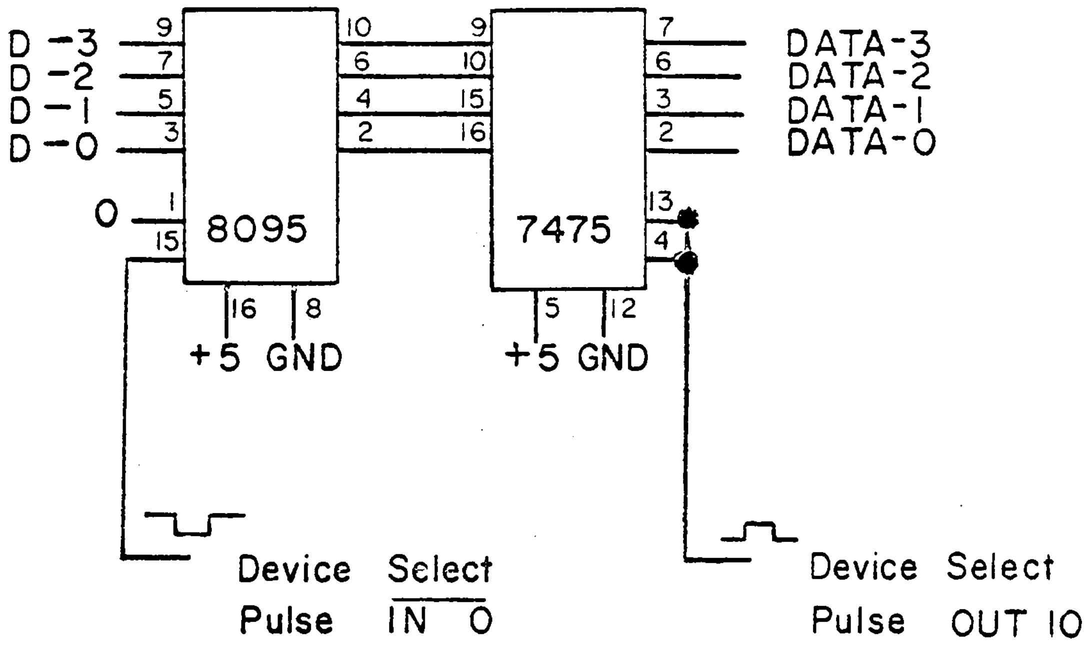


Fig. 4.4 Keithley Interface Circuit

is simple. Four bits are required for each of the three digits that are produced by the digital panel meter. Three 7475 flip-flops and three 8095 three-state buffers are required to latch the twelve bits of data (see Fig. 4.5). A positive device select pulse is used to latch the three 7475 flip-flops simultaneously as well as the 7475 flip-flop used in the Keithley meter interface. Each 8095 buffer has its own individual negative device select pulse to allow the Mark 80 to read the data from one 7475 flip-flop at a time. All three 8095 buffer outputs are connected to the data bus lines D0 through D3. Lines D4 through D7 are set to zero by using three 2-input AND gates and a 8095 three-state buffer with its inputs set at "logic 0" (see Fig. 4.6). The arrangement of the three 2-input AND gates is that of a 4-input AND gate. The purpose of creating a 4-input AND gate is to allow four devices to use the same device for the same purpose, which is the setting of the data bus lines D4 through D7 to "logic 0" while input data is being read on the other four data bus lines by the Mark 80. The inputs to the AND gates are always at "logic 1" unless a negative device select pulse is sent. Thus, only one line at a time will ever be at "logic 0". Table 4.2 summarizes the truth table for the AND gate circuit. A 4-input AND gate could be used in place of the three 2-input AND gates, but in the decoding circuit that appears later in this manuscript, a 2-input AND gate is needed. Each 7408 chip has four 2-input AND gates; three of these and one 4-input AND gate would be wasted if a 7425 chip, which has two 4-input AND gates, were used with the 7408.

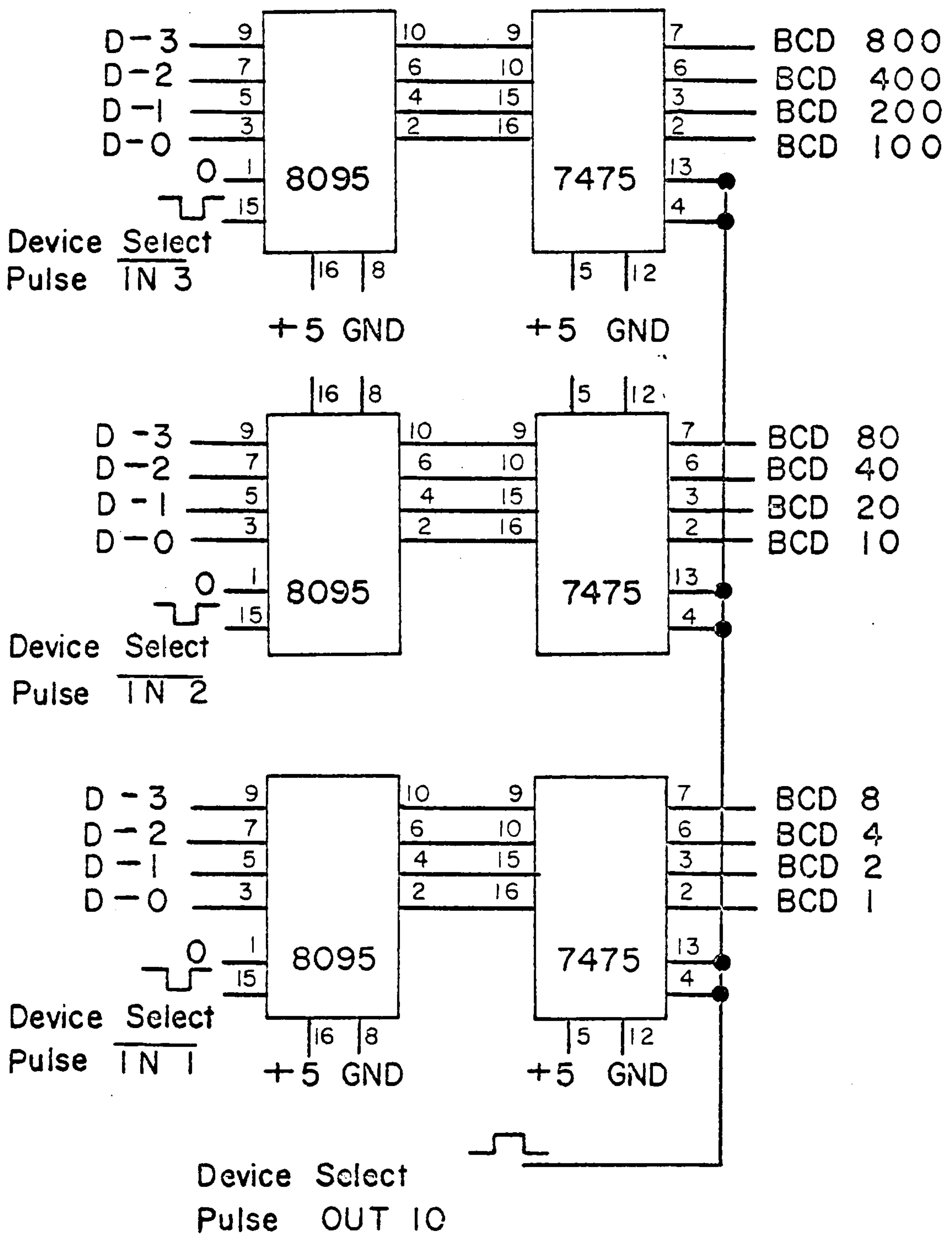


Fig. 4.5 Digital Panel Meter Interface Circuit

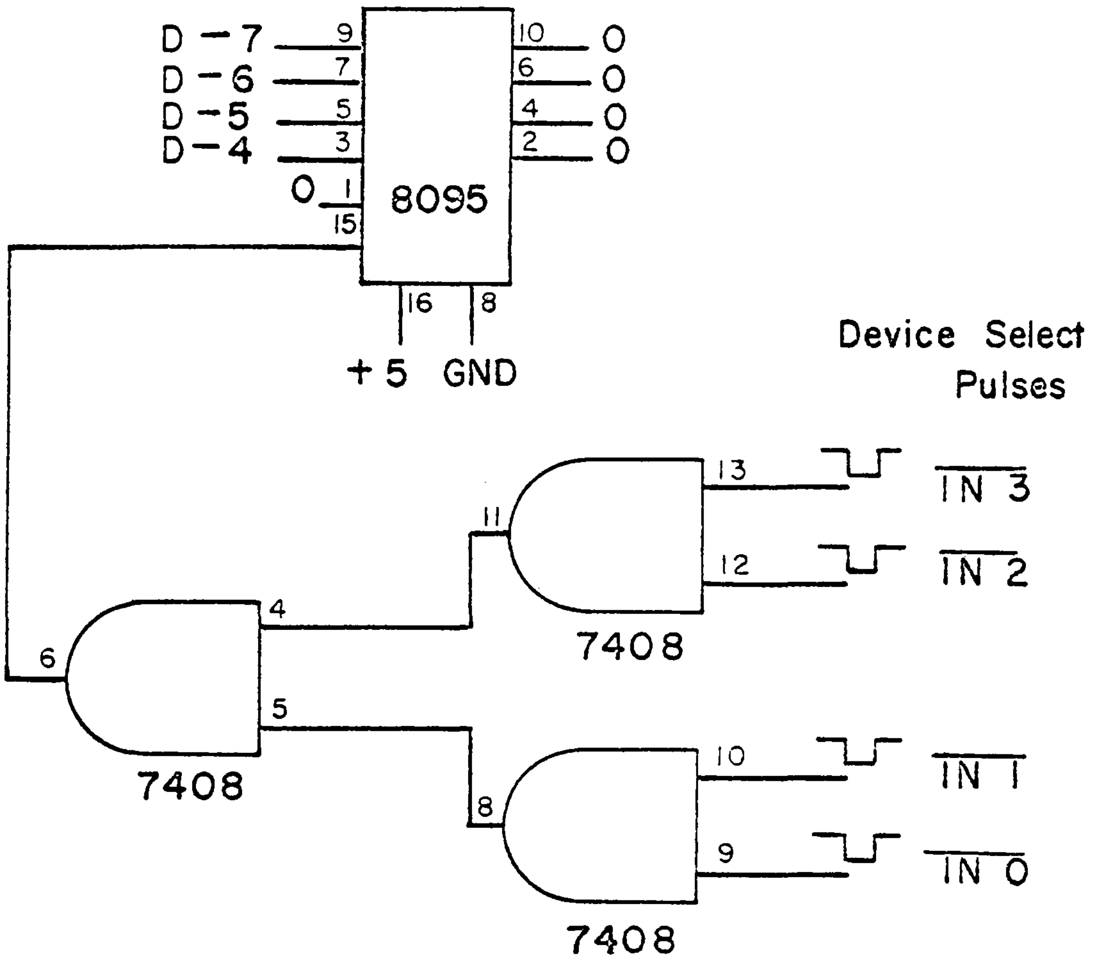


Fig. 4.6 Zero Input Interface Circuit

Table 4.2

4-Input AND Gate Truth Table for Device Select Pulses

IN 0 through IN 3

Logic State of				Logic State of
IN 0	IN 1	IN 2	IN 3	Output
1	1	1	1	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

In Figure 4.7, the connections for the digital panel meter are shown. A 110 VAC source and an earth shield are required for operation of the DPM. An external trigger is used for a conversion rate of five times per second. To trigger a new conversion, a positive clock pulse is required at pin B. A positive clock pulse is obtained by inverting the negative clock pulse generated by an IN 1 instruction, which also enables a three-state buffer. The IN 1 instruction starts a new conversion by the DPM after the previous data has been latched by an OUT 10 instruction.

Thus, while the microcomputer performs calculations with the current data, the DPM updates the reading for the next calculation. Since the analog-to-digital conversion takes less than 10 msec, the new reading is ready for the next reading before the microcomputer is done with the current calculations. The analog voltage output of the Keithley is connected to the analog voltage input of the DPM (pins 10 and 2). To ensure the correct conversion from analog to digital, the analog and digital signal grounds are connected together (pins N and 10). Only twelve out of the thirteen data lines are used because all input voltages are positive and the data line for the number sign is not required. The twelve data lines are connected to the inputs of three 7475 flip-flops (see Fig. 4.5).

Interrupt Timing Circuit

Because of a software requirement, the Mark 80 needs to know when one 0.2 second period ends and the next 0.2 second period begins.

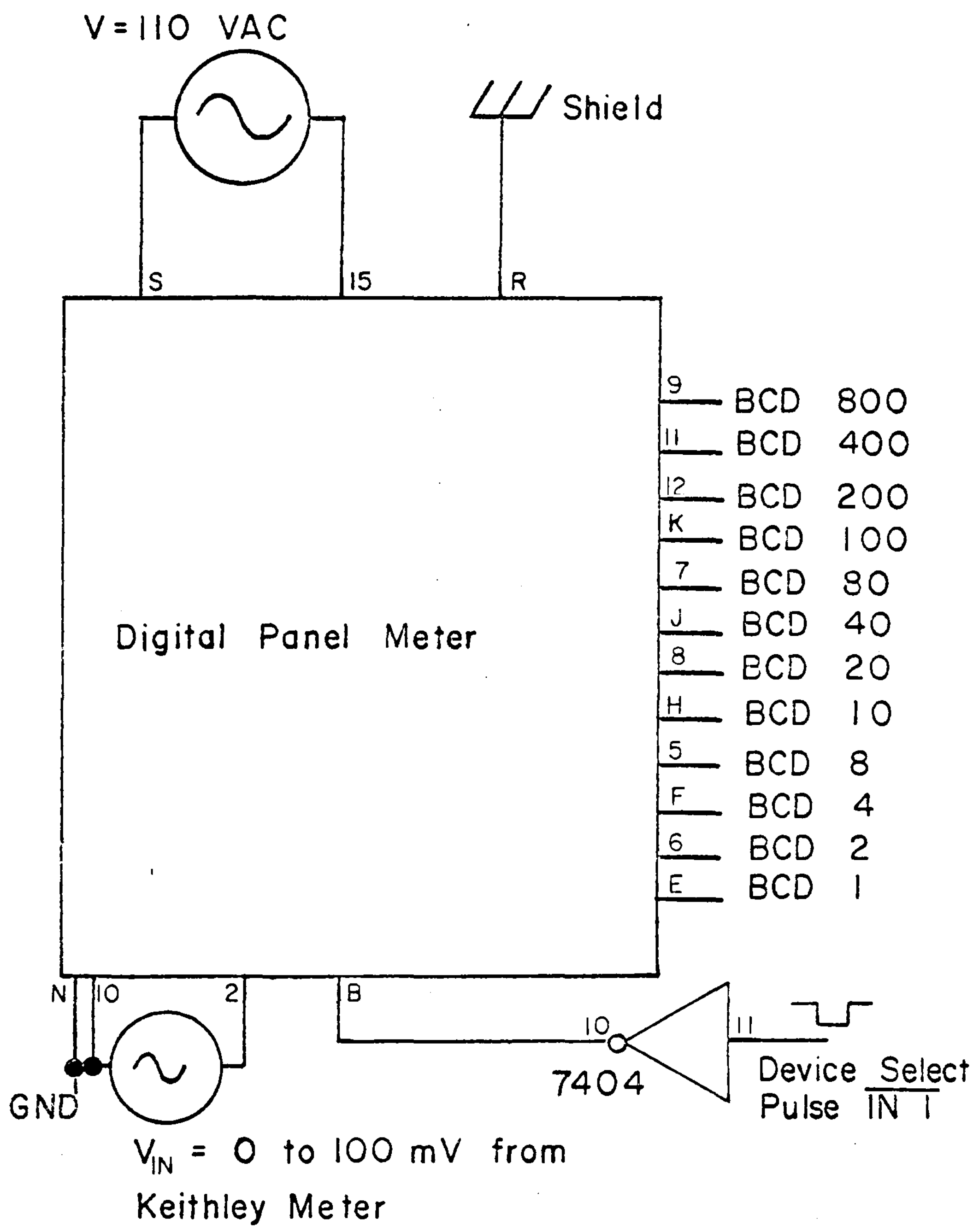


Fig. 4.7 Connection Diagram for Digital Panel Meter

The Mark 80 itself cannot keep track of the time because of the timing variations in the software system. Instead, an interrupt signal from a timing circuit is used to keep track of the time period of 0.2 seconds (see Fig. 4.8). A 555 timer IC chip is the basis for the external clock. A frequency of 40 Hz is used to generate eight clock pulses every 0.2 seconds. To generate a frequency of 40 Hz from the 555 timer chip the following equation is used,

$$\text{Hz} = 1.443/(R_A + 2R_B)\ C \tag{4.1}$$

where $R_A = 2.28\ \text{K}\Omega$

$R_B = 660\ \Omega$

$C = 10\ \mu\text{F}.$

The above value yields a frequency of 40.08 Hz, but this is of no consequence because the resistors have a tolerance of ± 5%, and the capacitor has a tolerance of ± 20%(13). Since the frequency must be as close as possible to 40 Hz, a potentiometer is used in the circuit to adjust the resistance to obtain the exact frequency.

At pin 3 of the 555 timer, a clock pulse is generated at every cycle. The clock pulses are directed to the input pin of a 7490 decade counter. Since there are eight counts every 0.2 seconds, on the eighth count pin 11 of the 7490 counter goes from a "logic 0" to a "logic 1", which is changed to a "logic 0" by an inverter. When a "logic 0" appears at the inverter output, two things happen:

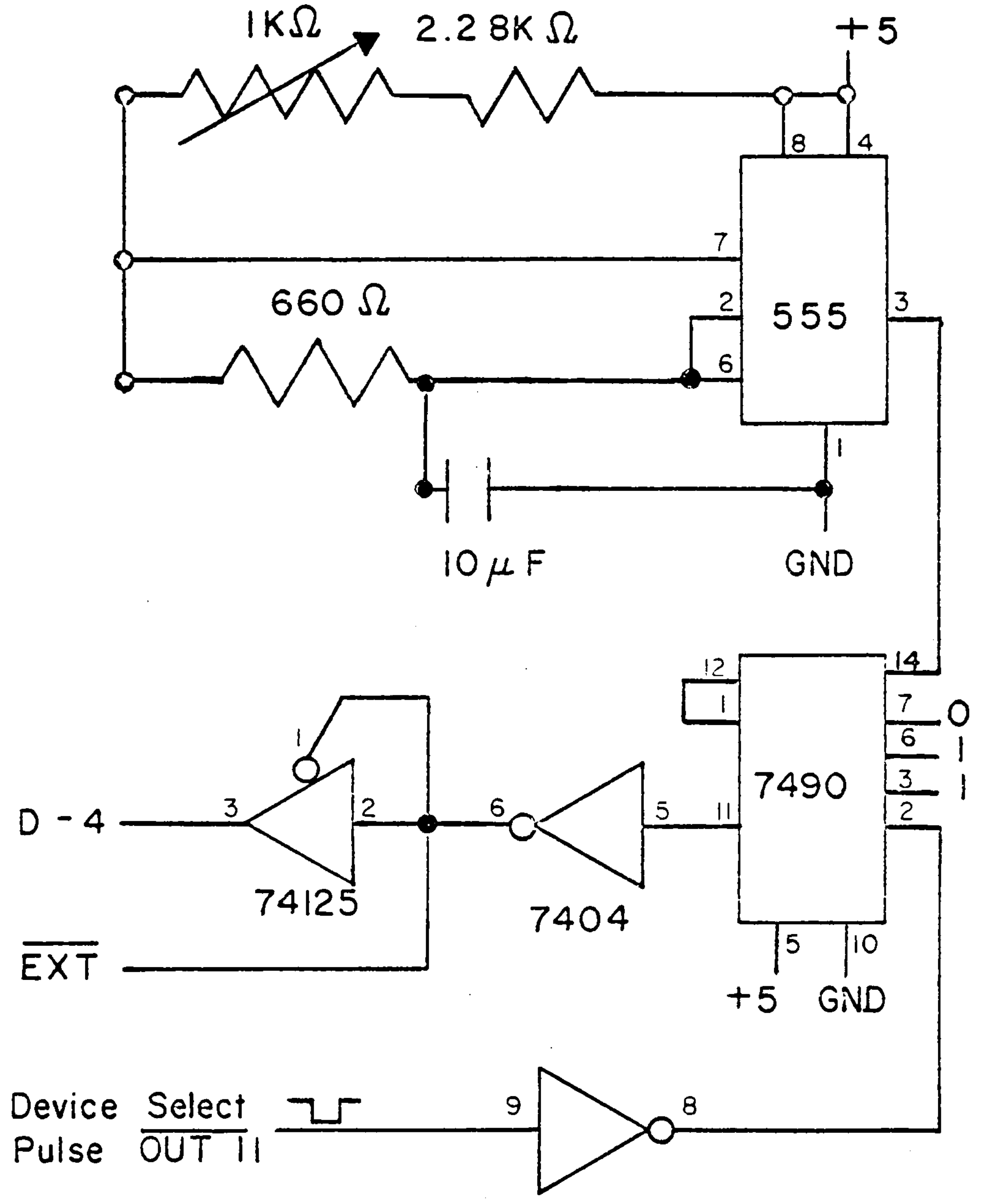


Fig. 4.8 0.2 sec. Timing Interval Circuit

(1) a "logic 0" enables a three-state buffer on a 74125 chip, which allows that same signal to be read on to the data bus, and (2) a "logic 0" acknowledges the interrupt, resets the interrupt flag of the Mark 80, and jams an instruction on the data bus. Note that a data bus line that is not connected to a peripheral device during an interrupt signal assumes a "logic 1". Thus, during the interrupt, data bus lines D0 through D3 and D5 through D7 assume a "logic 1", while D4 is set to "logic 0". The octal code that appears on the data bus, 357, has significance to the Mark 80 because it directs the Mark 80 to memory location HI = 000_8 and LO = 050_8 to restart the program. Located at that address is an OUT instruction, that is used to reset the 7490 counter back to zero to count another eight clock pulses for a new time interval. Since a positive clock pulse is required to reset the 7490 counter, an inverter is used to change the negative clock pulse from the Mark 80 to a positive clock pulse. When the 7490 counter is reset, pin 11 goes to "logic 0", which disconnects the 74125 buffer from the data bus line and returns EXT to its normal state of "logic 1". The 7490 counter begins counting pulses again for the next interrupt.

Display Interface

The display is constructed from Texas Instruments Inc. TIL309 digital displays, which contain a built in 4-bit flip-flop, display, and buffer. The TIL309 operates in the following manner: Input data from the data bus is transferred to pins 7, 6, 10, and 15.

Pin 12 is used to light the decimal point, which appears to the right of the figure. Pin 5 is the TIL309 display enable input; data at the input lines are latched during a negative clock pulse. A "logic 1" at pin 11 allows the input data to be displayed, while a "logic 0" blanks the display.

The TIL309 display with the device select pulse OUT 4 is the only display with the decimal point illuminated and is the only display that is not blanked (see Fig. 4.9A). In the Reactimeter program, the software calls for blanking the other four displays on certain occasions. Device select pulses OUT 5, 6, 7, and 8 each enable a TIL309 display. The TIL309 display can be blanked by setting pins 7, 6, and 15 to "logic 1" and pin 10 to "logic 0". Table 4.3 lists the binary representation of the numbers, 0 through 9, and the software codes for plus sign, minus sign, and blank (for blanking the display).

The sign of the number is displayed by a light-emitting diode (LED). If the LED is on, the sign is negative, whereas if the LED is off, the sign is positive (see Fig. 4.9B). A 7475 flip-flop is used to latch the data appearing on the data bus line D7. Only a single bit is needed to light the LED. The octal codes for the minus and plus signs are 200 and 000, respectively. A positive device select pulse enables the 7475 flip-flop, which latches the data at bit D7. A 330 ohm resistor is used as a current limiting resistor to ensure that the LED does not burn out.

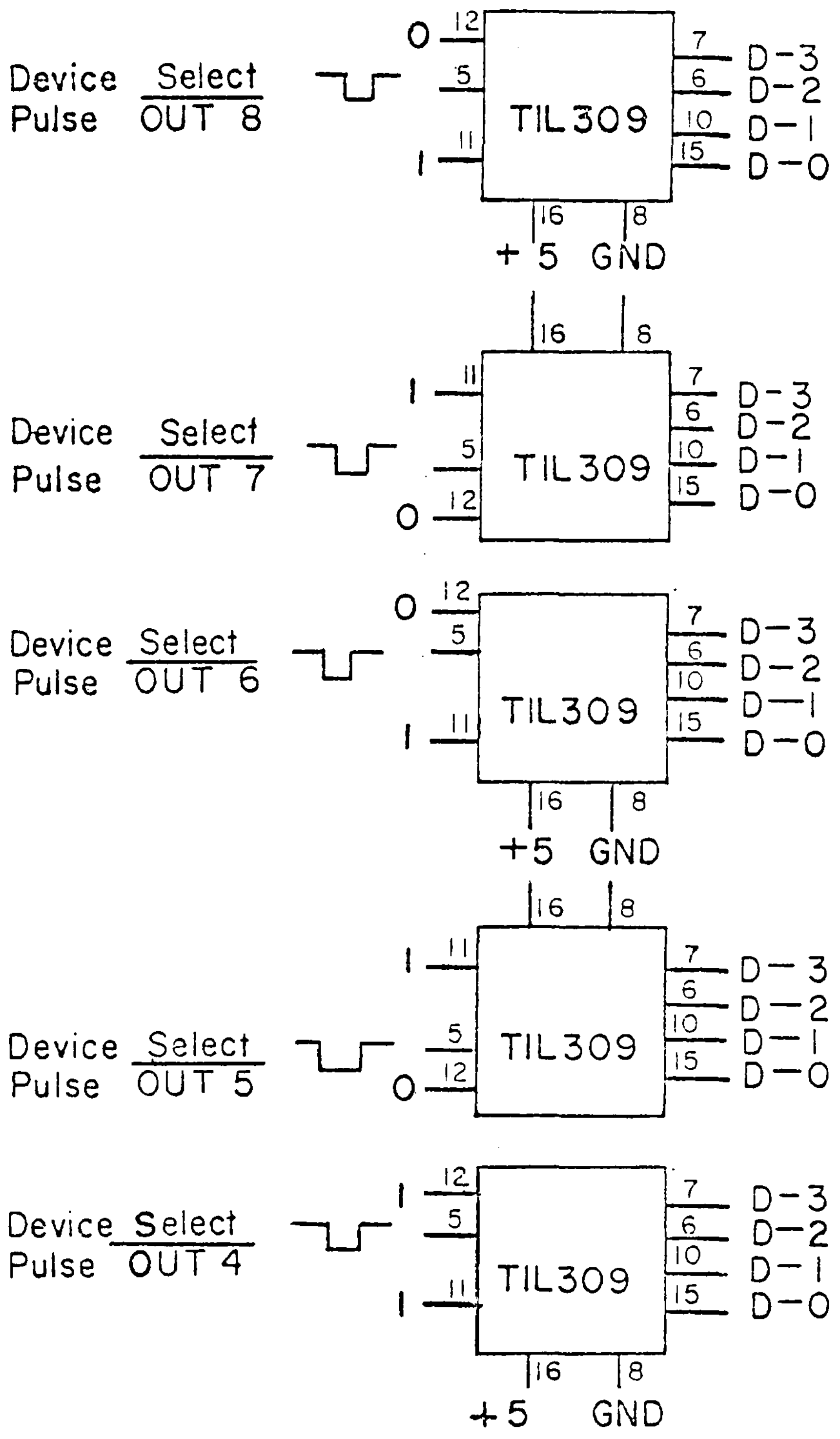


Fig. 4.9 A Output Interface Circuit Numeric Displays

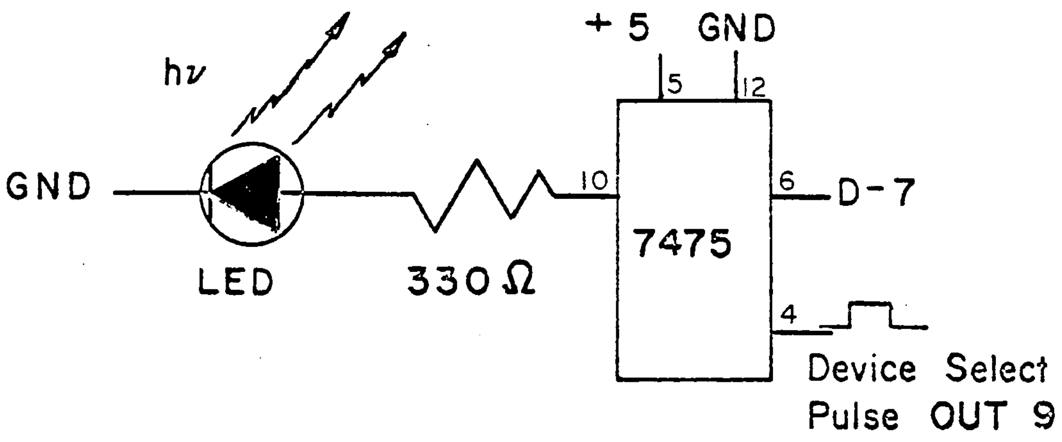


Fig. 4.9 B Output Interface Circuit Sign Display

Table 4.3

Binary Representation of Output Data

Output Data	Binary Representation
0	0 0 0 0 0 0 0 0
1	0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 1 0
3	0 0 0 0 0 0 1 1
4	0 0 0 0 0 1 0 0
5	0 0 0 0 0 1 0 1
6	0 0 0 0 0 1 1 0
7	0 0 0 0 0 1 1 1
8	0 0 0 0 1 0 0 0
9	0 0 0 0 1 0 0 1
plus sign	0 0 0 0 0 0 0 0
minus sign	1 0 0 0 0 0 0 0
blank	0 0 0 0 1 1 0 1

Decoding Circuit

Device select pulses are used as clock pulses for the DPM, 7475 flip-flops, 8095 three-state buffers, and TIL309 digital displays. Three chips are required to construct the decoding circuit: a 74154 4-line-to-16-line decoder, a 7408 AND gate, and a 7404 inverter. The 74154 input lines are connected to the address bus lines A0 through A3, which permit the device code to be read into the chip at the same time the chip is enabled by a negative clock pulse that is generated by either an IN or OUT instruction (see Fig. 4.10). All output channels of the 74154 are at "logic 1" in the standard state. When a 4-bit code is read into the address bus and the 74154 is enabled, the corresponding output channel goes to "logic 0" and remains there until the chip is disabled (see Table 4.4). The 7475 flip-flop requires a positive clock pulse, so an inverter is used to change the negative clock pulse into a positive clock pulse. To permit the use of one 74154 decoder for both IN and OUT instructions, the $\overline{\text{IN}}$ and $\overline{\text{OUT}}$ status bits are ANDed together. These bits cannot be directly connected together to the same line, since both cannot be "logic 0" at one time without causing damage. Both bits can be "logic 1" at the same time or they can have opposite logic. Table 4.5 summarizes the device codes, the peripheral devices, and the types of clock pulse sent out.

500 KHz Clock Rate

The Mark 80 microcomputer must operate at 500 KHz instead of

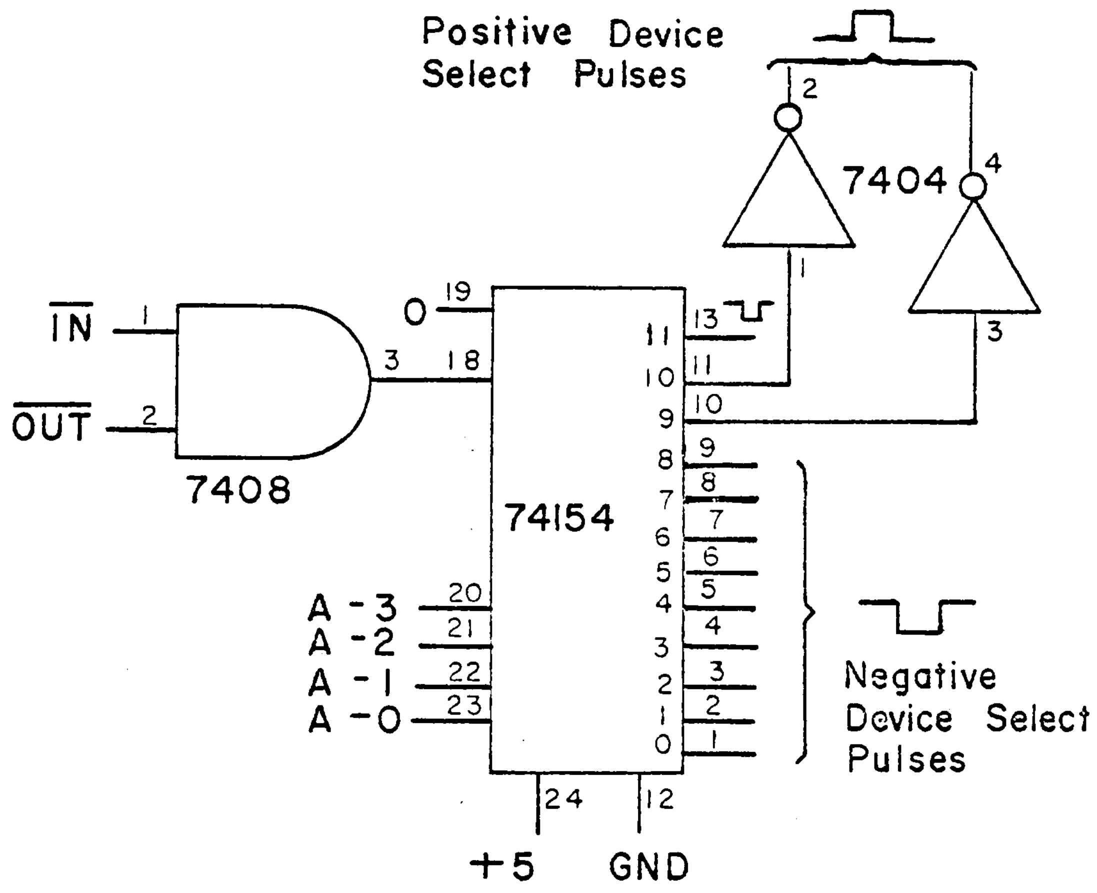


Fig.4. 10 Decoding Circuit

Table 4.4

Truth Table for a 4-Line-to-16-Line Decoder

Octal Device Code	Inputs				Outputs															
	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
000	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
001	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
002	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
003	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
004	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
005	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
006	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
007	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
010	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
011	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
012	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
013	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
014	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
015	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
016	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
017	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 4.5

Peripheral Device vs Clock Pulse

Peripheral Device	Device Code	Clock Pulse
8095 Three-state buffer	000	Negative
8095 Three-state buffer	001	Negative
8095 Three-state buffer	002	Negative
8095 Three-state buffer	003	Negative
TIL309 Display	004	Negative
TIL309 Display	005	Negative
TIL309 Display	006	Negative
TIL309 Display	007	Negative
TIL309 Display	010	Negative
7475 Flip-flop	011	Positive
Four 7475 Flip-flops	012	Positive
7490 Decade Counter	013	Negative

the normal rate of 2 MHz to allow the software to function correctly. At 2 MHz, mathematical calculations performed by the math subroutine package produces inaccurate results, but at the slower rate of 500 KHz the correct results are produced. To decrease the operating rate from 2 MHz to 500 KHz, the 2 MHz internal clock of the Mark 80 is used along with the circuit on Figure 4.11 to single step the Mark 80 through the program. The 7493 binary counter chip is used to generate one clock pulse for every four clock pulses received, and the new clock pulse is used to single step the Mark 80 through the program. The yellow switch on the top row of switches on the front panel of the Mark 80 must be in the up position to allow single step operation to proceed.

At a speed of 2 MHz, each machine cycle is executed immediately after the previous machine cycle. When the microcomputer is single stepped at a clock rate of 500 KHz, the machine cycles still operate at 2 MHz, but there is a waiting period between two consecutive machine cycles.

Conclusion

All the circuits that have appeared in this section are to be wire wrapped together on one circuit board that can be plugged directly into the Mark 80 system. Because of a lack of time this was not done and the circuits have only been tested individually. The software required to direct information in and out of these circuits is discussed in the next chapter.

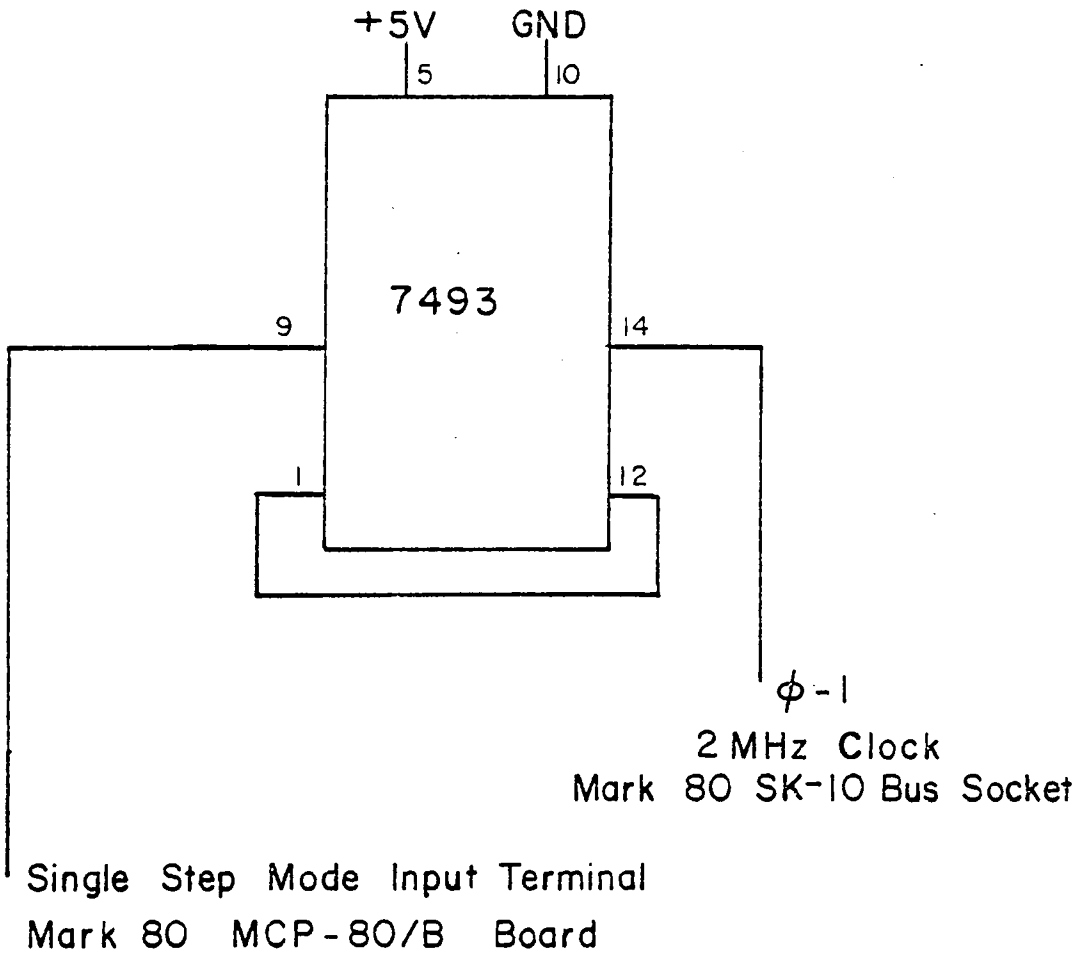


Fig. 4.11 Automatic Single Step Mode Circuit

CHAPTER V

SOFTWARE

Introduction

The programming of the microcomputer is the last step in the development of the reactimeter. The program has two major computations to perform. One computation is the interpretation of the flux from the two signals received from the Keithley micro-microammeter. The flux interpretation program is comprised of four routines: two which convert electrical current into neutron flux, one that selects one of the conversion routines and the correct conversion constant, and one that stores the calculated flux in the correct memory location. The second computation is the calculation of the reactivity from the present and past neutron flux. Simpson's Rule is used to evaluate the integral in equation (2.18). Two flux measurements must be recorded before a new reactivity calculation can be performed. Thus, if the flux is measured at each time interval, Δt, the reactivity is calculated every $2\ \Delta t$. The time interval, Δt, that is chosen must be based on the decay times of the neutron precursors and on the time requirements for the calculations of the neutron flux and reactivity.

Neutron Flux Algorithm

The neutron flux algorithm converts electrical current data received from the Keithley meter into neutron flux data. The Keithley has sixteen range settings; eight range settings have mantissas between one and three, and eight have mantissas between three and ten. On all ranges the mantissa is represented as an analog voltage between 0 and 100 mV at the auxiliary output of the Keithley meter. This signal is digitized by a digital panel meter and read into the microcomputer in binary-coded-decimal form. During the read routine, the data is changed into binary-floating-point form and stored in MANT (normalized mantissa). By using one of two equations, the actual mantissa of the electrical current measured by the Keithley is determined and is labeled ACMAN (actual mantissa). MANT is a number between 0 and 100 mV that represents a number either between one and three, or a number between three and ten. If MANT represents a number between one and three, then 0 mV corresponds to one and 100 mV corresponds to three (see Table 5.1). For a range setting with the mantissa between one and three, ACMAN is defined as follows,

$$\text{ACMAN} = 1 + \text{MANT} * (3-1)/100, \tag{5.1}$$

which simplifies to

$$\text{ACMAN} = 1 + \text{MANT} * 0.02. \tag{5.2}$$

If the range setting of the Keithley is between three and ten, 0 mV corresponds to three and 100 mV corresponds to ten. ACMAN is defined as follows,

$$\text{ACMAN} = 3 + \text{MANT} * (10-3)/100, \tag{5.3}$$

Table 5.1

Mantissa Value vs Output Voltage

Electric current range setting with the mantissa between one and three.

Value of Mantissa	Output Voltage (mV)
1.00	000.0
1.50	25.0
2.00	50.0
2.50	75.0
3.00	100.0

Electric current range settings with the mantissa between three and ten.

Value of Mantissa	Output Voltage (mV)
3.00	000.0
4.00	14.3
5.00	28.6
6.00	42.9
7.00	57.2
8.00	71.5
9.00	85.8
10.00	100.0

which simplifies to

$$\text{ACMAN} = 3 + \text{MANT} * 0.07. \tag{5.4}$$

Equations (5.2) and (5.4) are the equations that are employed to calculate the actual mantissa of the electrical current measured by the Keithley. Table 5.2 lists the binary-floating-point values for the constants and their labels for memory addressing.

To obtain the complete value for the current, ACMAN must be multiplied by the correct power of ten. The exponent is between -11 and -4. The neutron flux is obtained by multiplying the electrical current by the conversion factor, $K = 2.5 \times 10^{13}$ nv/amp, from the compensated ion chamber. The neutron flux is equal to the current times K.

$$\text{FLUX} = \text{Current} * 2.5 * 10^{13} \text{ nv/amp}, \tag{5.5}$$

which is equivalent to

$$\text{FLUX} = \text{ACMAN} \times 10^{J} * 2.5 * 10^{13} \text{ nv/amp}, \tag{5.6}$$

where J = -11, -10, -9, ... -4. For simplicity, the constants 10^{J} and K can be defined as a new constant, CONn. Let $\text{CONn} = 2.5 \times 10^{13+J}$ for n = 1 to 8, and let J be defined in terms of n such that J = n - 12. For n = 1, J = -11, $\text{CON1} = 2.5 \times 10^{2}$, and if n = 8, J = -4, and $\text{CON8} = 2.5 \times 10^{9}$. Table 5.2 lists the values of CONn and their binary-floating-point values. Equation (5.6) can be simplified to the following,

Table 5.2

Conversion Constants for Neutron Flux Algorithm

Number	Label	Binary Floating Point Representation			
0.02	CONT2	173	043	327	010
0.07	CONT7	175	017	134	050
1.00	ONE	201	000	000	000
3.00	THREE	202	100	000	000
2.5×10^2	CON1	210	171	377	320
2.5×10^3	CON2	214	034	077	340
2.5×10^4	CON3	217	103	120	010
2.5×10^5	CON4	222	164	043	270
2.5×10^6	CON5	226	030	226	270
2.5×10^7	CON6	231	076	274	100
2.5×10^8	CON7	234	156	152	370
2.5×10^9	CON8	240	025	003	060

$$\text{FLUX} = \text{ACMAN} * \text{CONn},\ n = 1 \text{ to } 8. \tag{5.7}$$

Equations (5.2), (5.4), and (5.7) comprise the neutron flux algorithm.

Reactivity Algorithm

The reactivity algorithm is based on equations (2.17) and (2.18),

$$\rho'(t) = 1 - \left(\sum_{i=1}^{6} a_i Y_i(t)\right)/\phi(t). \tag{2.17}$$

$$Y_i(t) = \phi_0 \exp(-\lambda_i t) + \lambda_i \int_0^t \phi(\tau)\exp(-\lambda_i(t-\tau))d(\tau). \tag{2.18}$$

Since the microcomputer cannot integrate, a numerical integration technique, such as Simpson's Rule, must be used to evaluate the integral in equation (2.18). Simpson's Rule requires the use of three points to calculate the area under the curve. The first point in the area calculation is the end point from the previous area calculation. Thus, two neutron flux measurements are needed before the next neutron precursor flux and reactivity can be calculated.

The precursor flux at time t is calculated from the previous precursor flux at time $t = 2\ \Delta t$. Let t_k, t_{k+1}, and t_{k+2} be three consecutive points in time when a flux measurement is recorded, and define $\Delta t = t_{k+1} - t_k$, and $2\ \Delta t = t_{k+2} - t_k$. Assume that $Y_i(t)$ is known and that a relationship for $Y_i(t_{k+2})$ can be derived from $Y_i(t_k)$. By applying Simpson's Rule to the integral, one can calculate $Y_i(t_k)$ and $Y_i(t_{k+2})$ in equations (5.8) and (5.9), respectively. (See the following page.) $Y_i(t_{k+2})$ can be rewritten as equation (5.10), which can be simplified to equation (5.11).

$$Y_i(t_k) = \phi_0 \exp(- \lambda_i t) + \frac{\lambda_i \Delta t}{3} [\phi(t_0) \exp(- \lambda_i(t_k-t_0)) + 4\phi(t_1)\exp(- \lambda_i(t_k-t_1) + 2\phi(t_2)\exp(-\lambda_i(t_k-t_2))$$

$$+ \ldots + 4\phi(t_{k-1}) \exp(- \lambda_i(t_k-t_{k-1})) + \phi(t_k)] \qquad (5.8)$$

$$Y_i(t_{k+2}) = \phi_0 \exp(-\lambda(t_{k+2})) + \frac{\lambda_i \Delta t}{3} [\phi(t_0) \exp(-\lambda_i(t_{k+2} - t_0)) + 4\phi(t_1)\exp(-\lambda_i(t_{k+2} - t_1))$$

$$+ 2\phi(t_2) \exp(-\lambda_i(t_{k+2} - t_2)) + \ldots. + 4\phi(t_{k-1})\exp(-\lambda_i(t_{k+2} - t_{k-1}))$$

$$+ 2\phi(t_k) \exp(-\lambda_i(t_{k+2} - t_k)] + 4\phi(t_{k+1}) \exp(-\lambda_i(t_{k+2} - t_{k+1})) + \phi(t_{k+2})] \qquad (5.9)$$

$$Y_i(t_{k+2}) = \exp(-2\lambda_i \Delta t) \times \{\phi_0 \exp(-\lambda_i t_k) + \frac{\lambda_i \Delta t}{3} [\phi(t_0) \exp(-\lambda_i(t_k-t_0)) + 4\phi(t_1) \exp(-\lambda_i(t_k-t_1))$$

$$+ 2\phi(t_2) \exp(-\lambda_i(t_k-t_2)) + \ldots + 4\phi(t_{k-1}) \exp(-\lambda_i(t_{k-1}-t_k)) + \phi(t_k) \exp(-\lambda_i(t_k-t_k))]\}$$

$$+ \frac{\lambda_i \Delta t}{3} [\phi(t_k) \exp(-\lambda_i(t_{k+2}-t_k)) + 4\phi(t_{k+1}) \exp(-\lambda_i(t_{k+2} - t_{k+1})) + \phi(t_{k+2})] \qquad (5.10)$$

$$Y_i(t_{k+2}) = Y_i(t_k) \exp(-2 \lambda_i \Delta t) + \frac{\lambda_i \Delta t}{3} [\phi(t_k) \exp(-2 \lambda_i \Delta t) + 4\phi(t_{k+1}) \exp(-\lambda_i \Delta t) + \phi(t_{k+2})] \qquad (5.11)$$

Equations (5.11) and (2.17) are the two equations that are used in the reactivity algorithm. Equation (5.11) is used once for each of the six neutron precursor groups, and is performed six times every two time intervals. Equation (2.17) is used only once every two time interval.

One additional modification makes it easier for the microcomputer to perform the reactivity algorithm. We define and calculate eighteen constants, three for each of the six equations. Thus, let

$$\text{CEX1i} = \exp\ (-\ 2\ \lambda_i \Delta t) \tag{5.12}$$

$$\text{CEX2i} = 4 \times \exp\ (-\ \lambda_i \Delta t) \tag{5.13}$$

$$\text{CLAMi} = \lambda_i \Delta t/3 \tag{5.14}$$

for i = 1 to 6, where $\Delta t = 0.2$ seconds. Refer to Table 5.3 for the numerical values and binary-floating-point values for each of the above constants and the relative yield fraction, Ai.

Time Interval

How often should one take neutron flux measurements? There are two criteria: (1) a time interval that is short enough so that it provides accurate results when used in the point reactor kinetics equations, (2) a time interval that is long enough to permit the microcomputer to perform all calculations in the time span of two time intervals, and (3) a time interval that is long enough to permit a significant change in power.

Table 5.3

Constants for Reactivity Algorithm

Number	Label	Binary Floating Point Representation			
0.03800	A1	174	033	245	340
0.21310	A2	176	132	066	340
0.18800	A3	176	100	203	020
0.40690	A4	177	120	125	060
0.12800	A5	176	003	022	160
0.02600	A6	173	124	375	260
3.97468	CEX21	202	176	141	050
3.93710	CEX22	202	173	371	162
3.77649	CEX23	202	161	262	002
3.42395	CEX24	202	133	041	377
1.98634	CEX25	201	176	100	140
0.57770	CEX26	200	013	344	040
0.98738	CEX11	200	174	304	360
0.96880	CEX12	200	170	003	100
0.89137	CEX13	200	144	060	321
0.73271	CEX14	200	073	222	340
0.24660	CEX15	176	174	204	260
0.02086	CEX16	173	052	342	230
0.00212	CLAM1	170	012	357	260
0.00528	CLAM2	171	055	003	330
0.01917	CLAM3	173	035	012	150
0.05183	CLAM4	174	124	113	260
0.23333	CLAM5	176	156	256	020
0.64500	CLAM6	200	045	036	270

The maximum time interval depends upon the accuracy desired in the point reactor kinetics equations. Time intervals that are larger than the half-life of any single neutron precursor group yield an inaccurate representation of the neutron precursor flux of that group as well as the total neutron flux. Ideally, the time interval for recording neutron flux measurements should be less than the smallest half-life (0.2 sec. for group six, see Table 2.1). For good accuracy, the time interval should be no larger than 0.2 seconds. If necessary, neutron flux measurements can be taken every 0.5 seconds and still provide acceptable accuracy. At 0.5 seconds, only group six would not be represented accurately. Group six comprises approximately 3% of the total neutron flux and would introduce, at most, the same amount of error. Group five has a half-life of 0.55 seconds and comprises about 13% of the total neutron flux. Any further increase in the time interval past 0.5 seconds would introduce more error. The maximum time interval should not exceed 0.5 seconds.

The Mark 80 microcomputer is single stepped at 500 kHz instead of operating at 2 MHz to allow the 1702 EPROMs, which contain the reactimeter program and the floating point math routines, to operate correctly. There is no equation to determine how much time is required to perform all the calculations at 500 kHz. An estimate of the total time required for the routines SELECT, HRFC, LRFC, DIRECT, PRECURI, TRANSFER, and REACT to perform the calculations can be obtained by implementing the following program after the last instruction in the routine REACT.

```
LXI H, COUNT            ;HL ← COUNT
DCR M                   ;(COUNT) ← (COUNT - 1)
JMP SELECT              ;If the result is not zero, jump
                         to the starting address of SELECT
HLT                     ;HALT
```

If the value of COUNT is not zero after the DCR M instruction, the microcomputer jumps to the starting address of the routine SELECT and performs the calculation over again. If the result is zero, then the microcomputer stops. It was found that the average execution time at 500 kHz for the routines SELECT to REACT was 0.130 ± 0.005 seconds. Together, the average execution time of the routines INPUT and OUTPUT are less than 0.01 second. Thus, if we choose 0.15 seconds as a conservative estimate for the entire program to execute the longest instruction set, a time interval of 0.2 seconds is adequate for calculation time and provides good results.

The reactor period is the time that it takes for the neutron flux (and power) to change by a factor of "e". During a thirty second or greater period, the thermal neutron flux changes slowly. Thus, during two consecutive neutron flux measurements, the conversion from the analog signal to digital signal may not register any significant change. The digital panel meter can only detect changes of 0.1 mV from the output of the Keithley meter. Thus, many time intervals could pass before a voltage change of 0.1 mV could be detected by the digital panel meter. The reactivity displayed during this time would be inaccurate.

Reactimeter Program Introduction

There are ten main routines used to calculate the reactivity from the electrical current in addition to the math routines in the 8080 Intel Floating Point Package. Figure 5.1 contains the main flowchart of the reactimeter program. A discussion and flowchart of each of the ten routines follow. To help the reader better interpret the flowcharts, we have summarized below the symbols that we have used.

Registers

Symbols	Comments
A	Register A, accumulator
B	Register B
C	Register C
D	Register D
E	Register E
H	Register H
L	Register L
HL	Register pair HL, usually an address with H = HI and L = LO
(HL)	The data located at the memory location in H and L

Labels

MANT	The 16-bit address of MANT
(MANT)	The 8-bit value of MANT

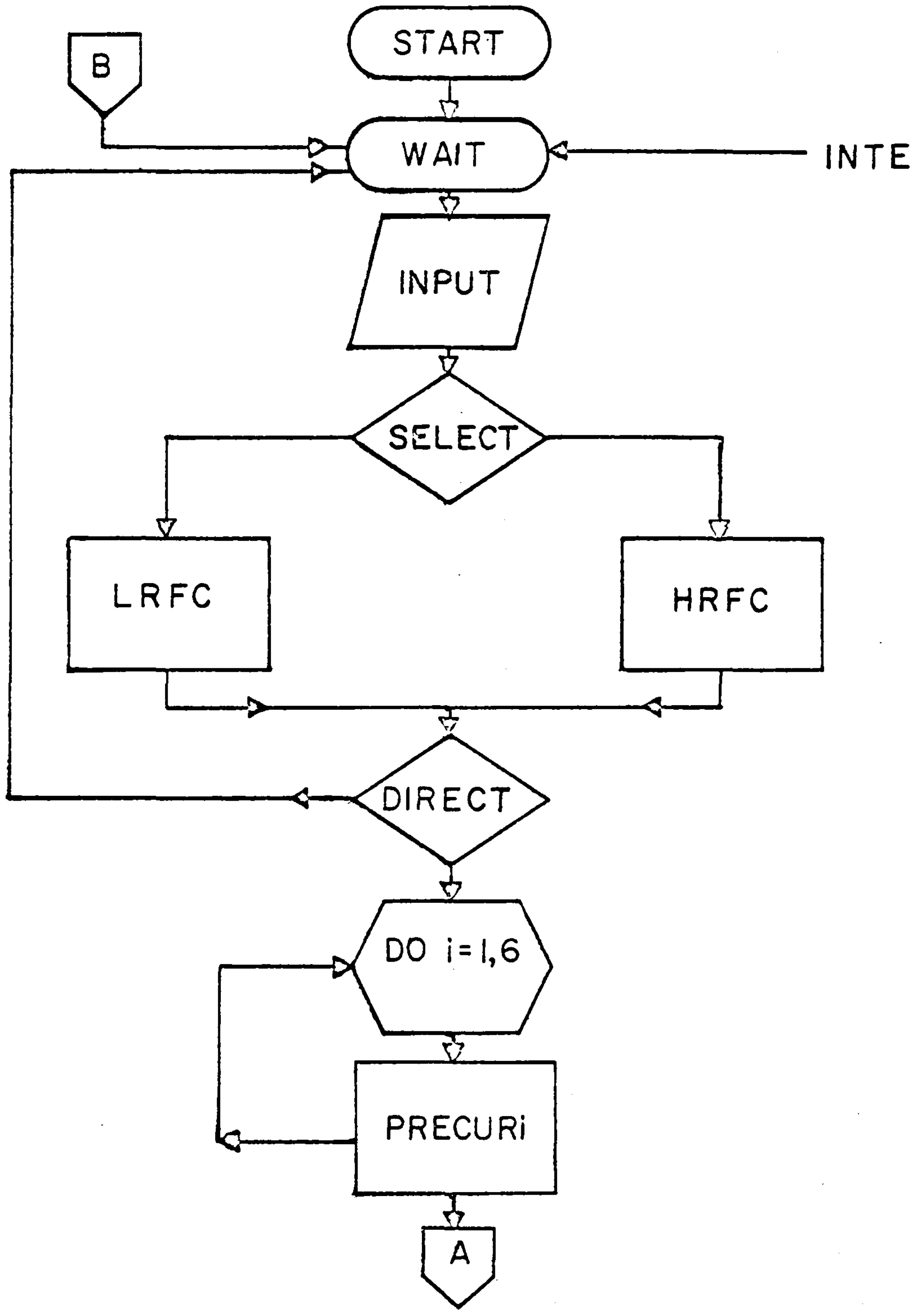


Fig. 5.1 A Main Flowchart

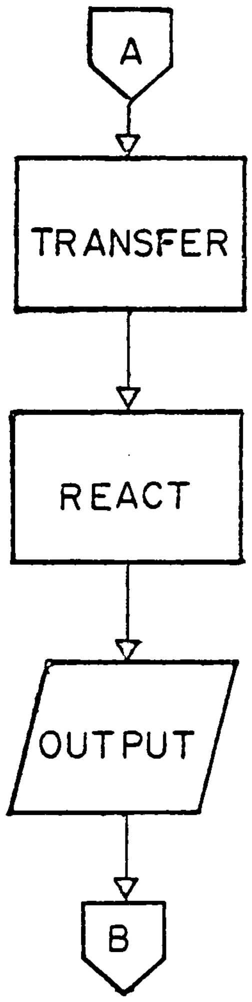


Fig. 5.1 B Main Flowchart (cont.)

Numbers

10	10 (base 10)
10 H	10 (base 16)
FE H	FE (base 16)

Expressions

HL ← MANT	The address of MANT is loaded into the HL register pair
H ← (HL)	The value of the memory location, whose address is in the HL register is moved into register H
OUT 10	Generate a device select pulse to device 10
A ← 7C H	Load A with 7C (base 16)
HI = 13 H	HI address is 13 (base 16)

INPUT

The purpose of INPUT is to read the BCD signals from the Keithley meter and the digital panel meter, convert the input data from the digital panel meter into an appropriate form, and store the results in the correct memory locations for later use by other routines (see Fig. 5.2 for flowchart). The first task of INPUT is to latch the two bytes of input data simultaneously. An OUT 10 instruction is used to generate a device select pulse to four 7475 flip-flops. Each flip-flop latches four bits. Three 7475 chips latch the tenths, ones, and tens digits from the digital panel meter, and the fourth 7475 chip latches the binary coded signal from the range dial of the Keithley meter. The memory storage location, DMANT + 4, for the tenths digit is loaded

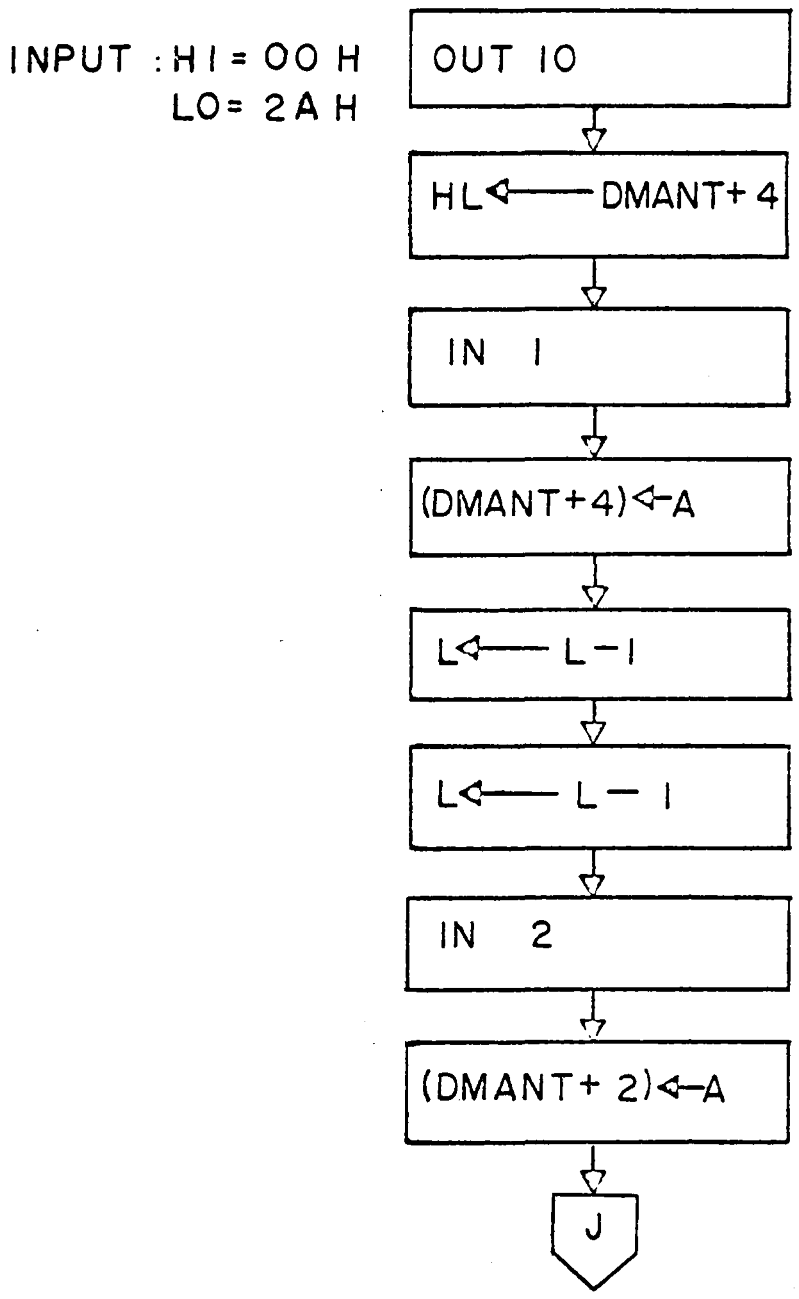


Fig. 5.2 A INPUT Routine

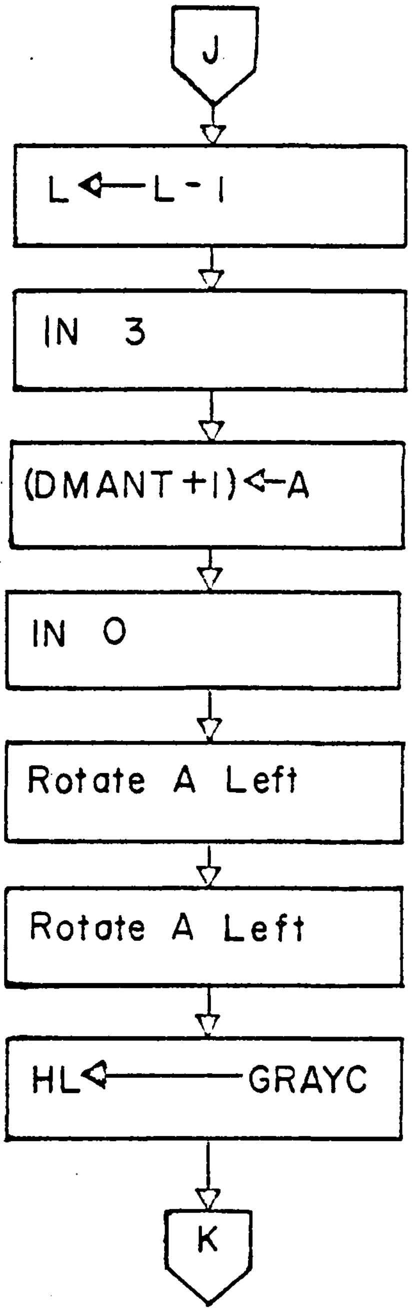


Fig. 5.2 B INPUT Routine (cont.)

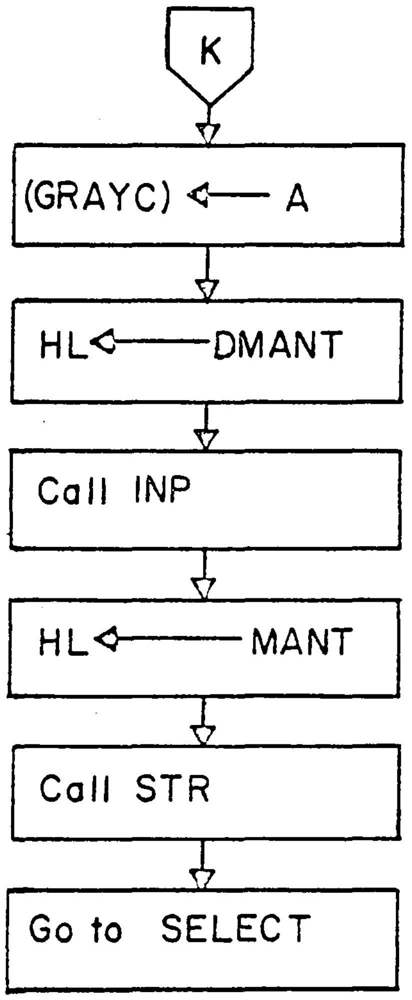


Fig. 5.2 C INPUT Routine (cont.)

into the HL register pair and an IN 1 instruction generates a device select pulse to input device one (see Table 5.4 for peripheral device and codes). Input device one is an 8095 three-state buffer, which when enabled allows the data stored in the 7475 chip to be read into bits D0 to D3 of the accumulator, whose contents are then moved to the address given by the HL register pair. Register L is decremented twice, and the memory storage location for the ones digit, DMANT + 2, appears in the HL register pair. One byte is skipped to leave space to hold the ASCII code for the decimal point, which must appear for the conversion subroutine INP. Input device two is an 8095 three-state buffer for the ones digit, and behaves the same way as input device one. Register L is decremented and the memory storage location (DMANT + 1) for the tens digit appears in the HL register pair. Input device three is an 8095 three-state buffer for the tens digit and functions the same way as input devices one and two.

Input device zero is an 8095 three-state buffer for the binary code from the range dial of the Keithley. After the bits are read into bits D0 to D3 of the accumulator, two RCL instructions are used to move the four bits to bits positions D2 to D5. The accumulator byte corresponds to the low address byte in the high address block 01 H. The data of the accumulator is stored in the memory location GRAYC. Initially, DMANT, the three digit BCD string number is converted into a binary floating-point number by calling the subroutine INP. The result is stored in MANT, and control is passed to SELECT.

Table 5.4

Device Codes and Peripheral Devices

Octal Device Number	Peripheral Device
000	8095 Three-state buffer from Gray shaft encoder from Keithley
001	8095 Three-state buffer from tenths digit of the DPM
002	8095 Three-state buffer from ones digit of the DPM
003	8095 Three-state buffer from tens digit of the DPM
004	Numeric Display-latch for ones digit
005	Numeric Display-latch for tens digit
006	Numeric Display-latch for hundreds digit
007	Numeric Display-latch for thousands digit
010	Numeric Display-latch for ten-thousands digit
011	7475 flip-flop for sign
012	Four 7475 flip-flops that latch data from the Keithley and DPM
013	7475 flip-flop for 7490 decade counter

SELECT

The purpose of SELECT is to direct the microcomputer to one of two electrical-current-to-neutron-flux conversion routines and select the corresponding constant, CONn, n = 1 to 8, for the range setting of the Keithley micro-microammeter. Each range setting has a binary value between zero and fifteen assigned to it. This value has been stored in bits D2 through D5 in memory location GRAYC and is the low address byte of the first byte of a four byte section of memory that contains the starting address of either HRFC (High Range Flux Conversion) or LRFC (Low Range Flux Conversion), and the address of the constant CONn. Since four bytes are required to store the two addresses, the binary code of each range setting of the Keithley is stored in bits D2 through D5 instead of D0 through D3, thus making it possible to address every fourth byte. Sixty-four bytes of memory must be reserved at the beginning of a 1K memory block to store the four bytes for each of the sixteen range settings.

The routine, SELECT (see Fig. 5.3 for flowchart), begins by loading the address of GRAYC into the HL register pair and then moves the value of GRAYC into the L register. The high address of the first byte of four bytes is loaded into the H register. The HL register pair contains the address of a byte that contains the starting low address of either HRFC or LRFC. This value is moved from memory to register E; register L is incremented. The HL register pair now contains the address of the second byte that

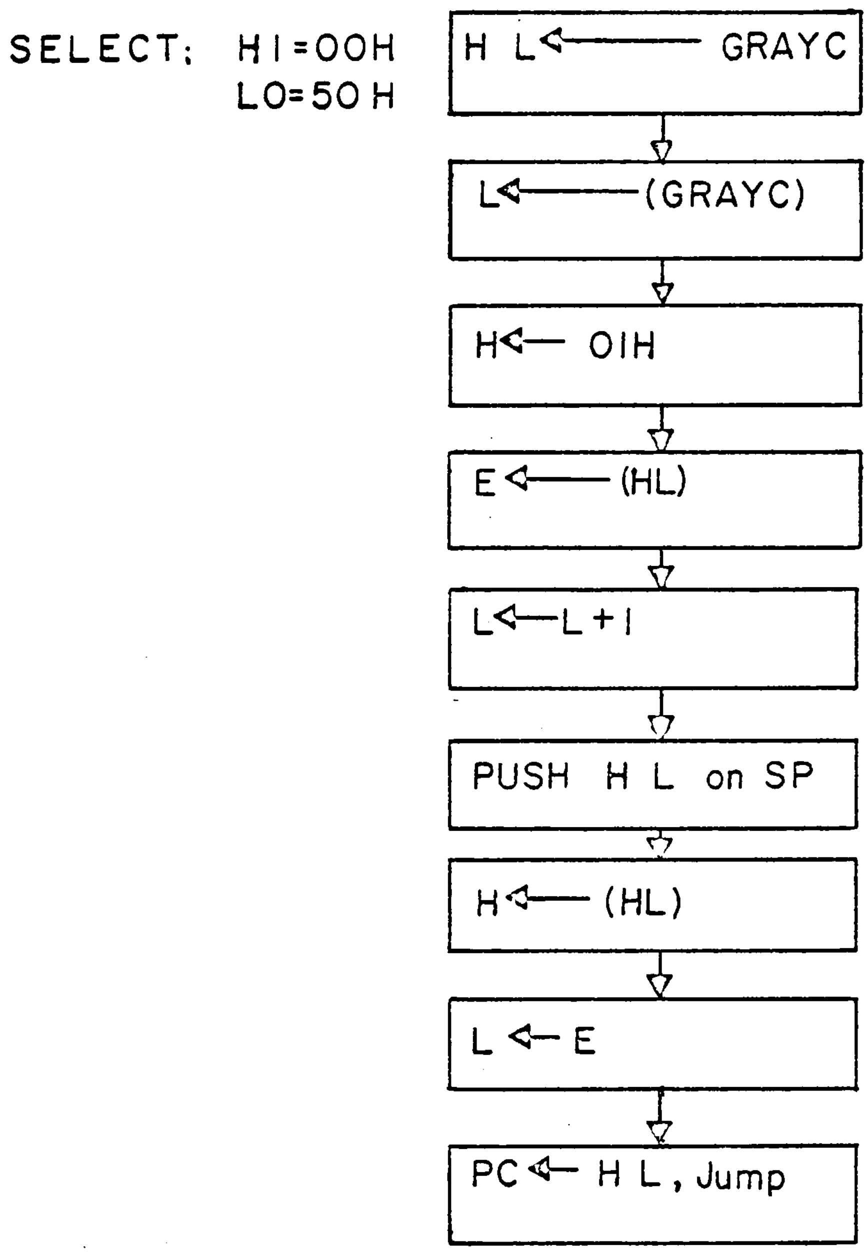


Fig. 5.3 SELECT Routine

contains the high address of either HRFC or LRFC. This address in the HL register pair is stored on the microprocessor stack for later reference by LRFC or HRFC to determine the address of the first byte of CONn. Durin a PUSH instruction, the contents of the HL register pair do not change, so the value of the byte addressed by the HL register pair can still be moved into the H register. The value in register E is moved to the L register. The HL register pair now contains the starting address of HRFC or LRFC. A jump to that address is initiated by a PCHL instruction, which moves the data in the HL register pair to the program counter and causes the microcomputer to jump to that address.

HRFC-LRFC

HRFC (High Range Flux Conversion) and LRFC (Low Range Flux Conversion) convert the electrical current input data into the corresponding neutron flux measured in the reactor core. Depending on the range setting on the Keithley meter, the mantissa of the electrical current is either between one and three or between three and ten. The normalized mantissa is read by the microcomputer and stored in MANT (normalized mantissa) in binary floating-point form. If HRFC (see Fig. 5.4 for flowchart) is chosen, equation (5.4) is used for the first half of the calculation. MANT is loaded into the floating-point-accumulator and multiplied by CONT7 (0.07); ONE (1.00) is added to obtain the actual mantissa (ACMAN). ACMAN is a dummy name because no memory space is allocated for the result,

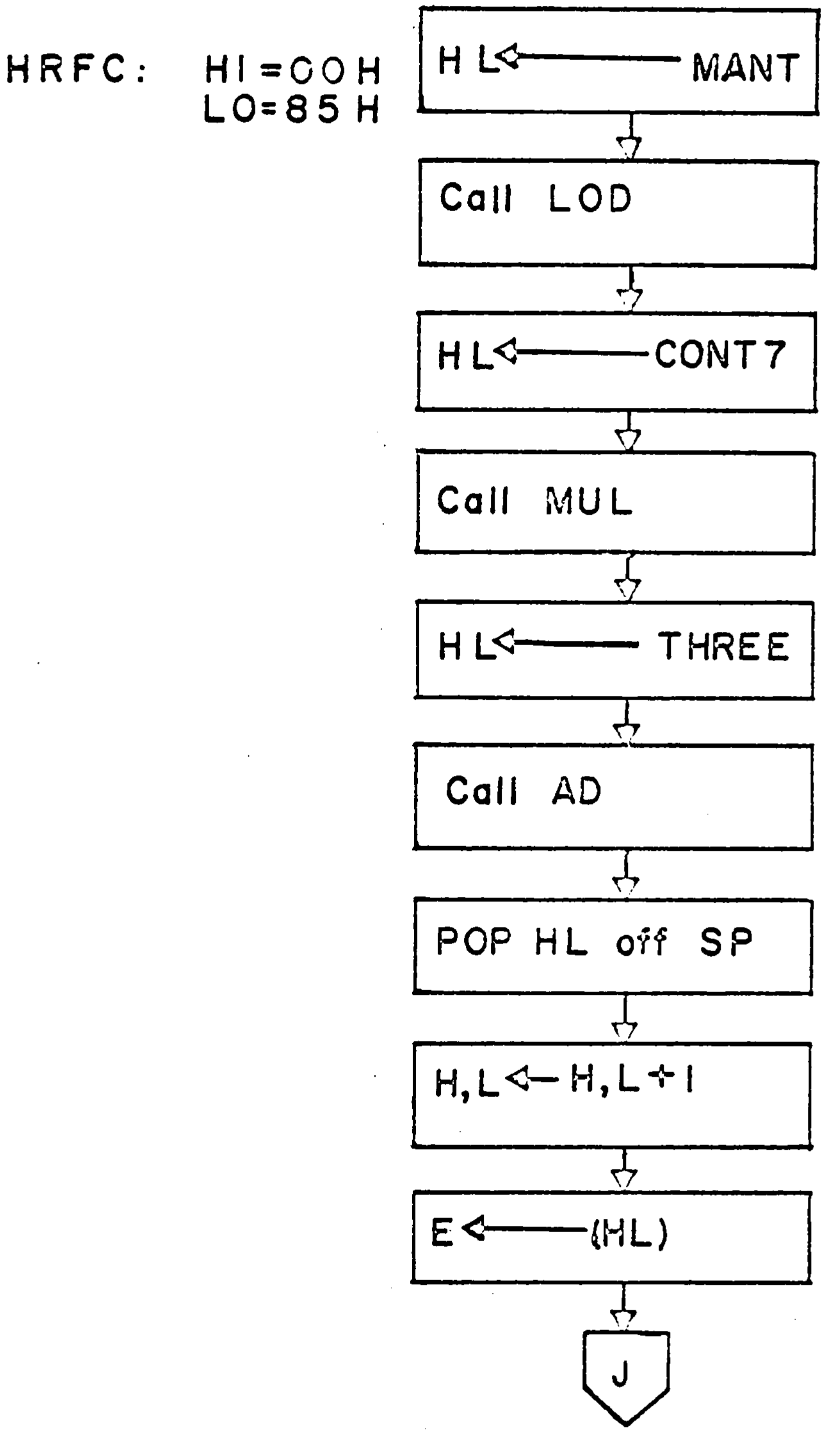


Fig. 5.4 A HRFC Routine

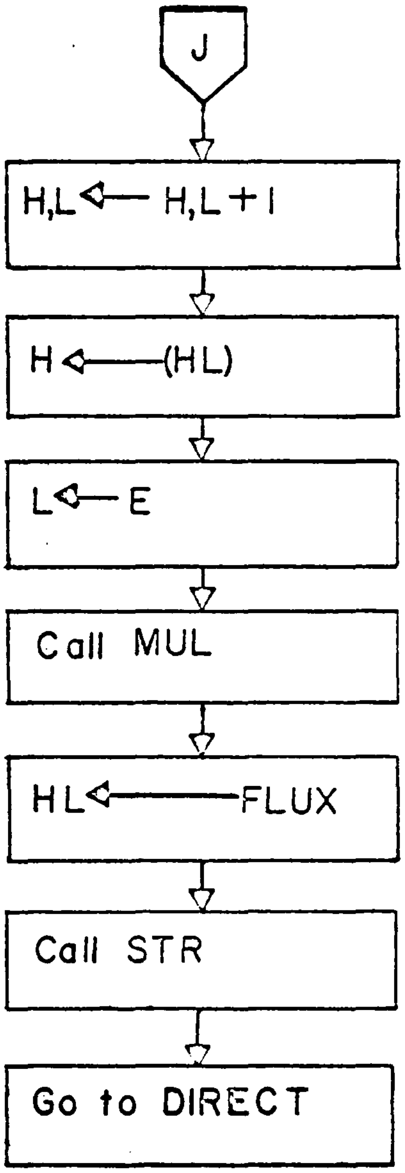


Fig. 5.4 B HRFC Routine (cont.)

which remains in the floating-point accumulator until it is multiplied by CONn. If LRFC (see Fig. 5.5 for flowchart) is chosen, equation (5.2) is used instead of equation (5.4). The procedure is the same with only the constants differing; CONT2 (0.02) and THREE (3.00) replace CONT7 and ONE. From here to the end, both routines are exactly alike and are based on equation (5.7).

So far, only two of the four bytes that the routine SELECT has previously addressed have been used. The address of the second byte is stored on the microprocessor stack and is available for retrieval. Using a POP instruction, that address is moved back into the HL register pair and then register L is incremented by one. The address of the byte in the HL register pair contains the value of the low address for the constant CONn. A multiplication is to be performed on ACMAN, which is in the floating-point-accumulator, by CONn, whose address must be loaded into the HL register pair. The low address of CONn is moved to register E from memory and register L is incremented. The HL register pair contains the memory location for the high address of CONn. This value is moved to register H, and the value of register E is moved to register L. The address in the HL register pair is the address of CONn and the multiplication between ACMAN and CONn can be performed. The result of the multiplication is stored in the memory location FLUX. Control is passed to the routine DIRECT, which assigns the value of FLUX to either FLUX2 or FLUX3.

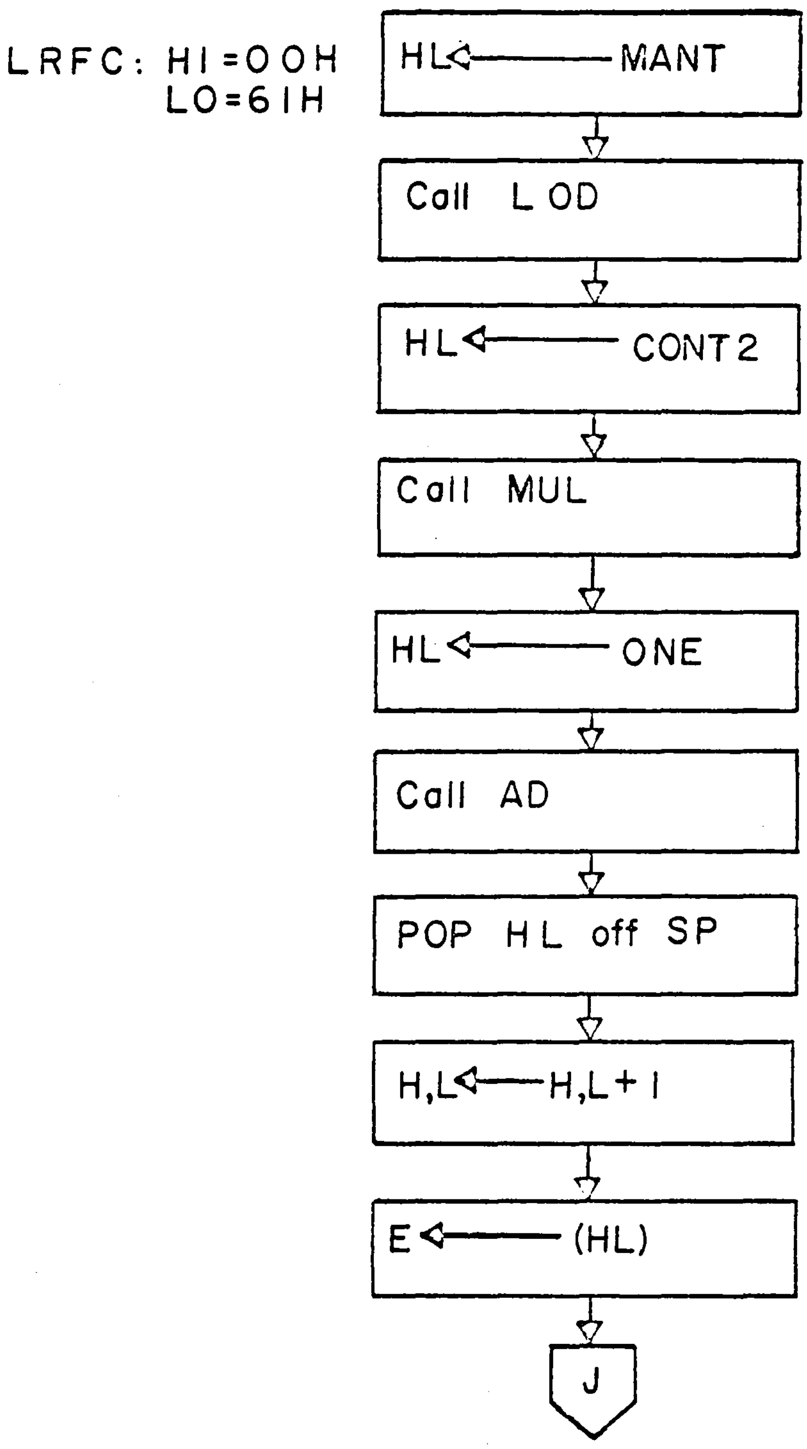


Fig. 5.5 A LRFC Routine

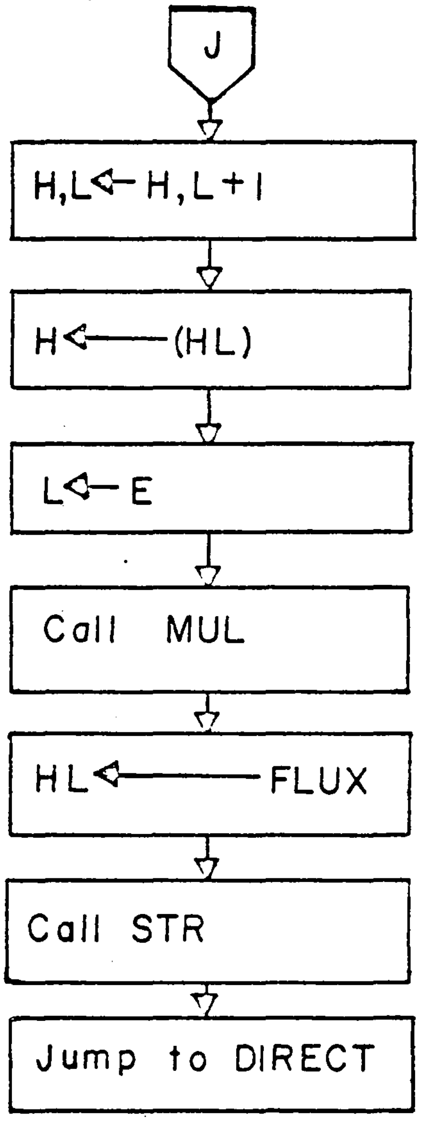


Fig. 5.5 B LRFC Routine (cont.)

DIRECT

DIRECT receives control from either HRFC or LRFC, and stores the calculated value of the most recent neutron flux measurement in either FLUX2 or FLUX3. If the neutron flux measurement is the first measurement after the previous reactivity calculation, then it represents $\phi(t - \Delta t)$ and is stored in FLUX2. If it is the second measurement after the previous reactivity calculation, it represents $\phi(t)$ and is stored in FLUX3.

The method of directing the neutron flux measurement to the proper storage address is simple (see Fig. 5.6 for flowchart). One byte of memory, MEM, is set aside with the initial value of two. When control is given DIRECT, it decrements MEM by one, and tests the result for a zero value by using a conditional jump instruction. If MEM is not equal to zero, the value of FLUX is loaded in the floating-point-accumulator and stored in the address of FLUX2 and control is given to WAIT. If the value of MEM is zero, the operation is the same, with FLUX being stored in the address of FLUX3. After FLUX3 has been stored, MEM is reset to the value of two and control is passed to PRECURi.

PRECURi

PRECURi calculates the neutron precursor flux at time t from the previous neutron precursor flux at time $t - 2\ \Delta t$ with the aid of equation (5.11). There are six PRECURi routines, one for each of the six delayed neutron groups. In the six PRECURi routines

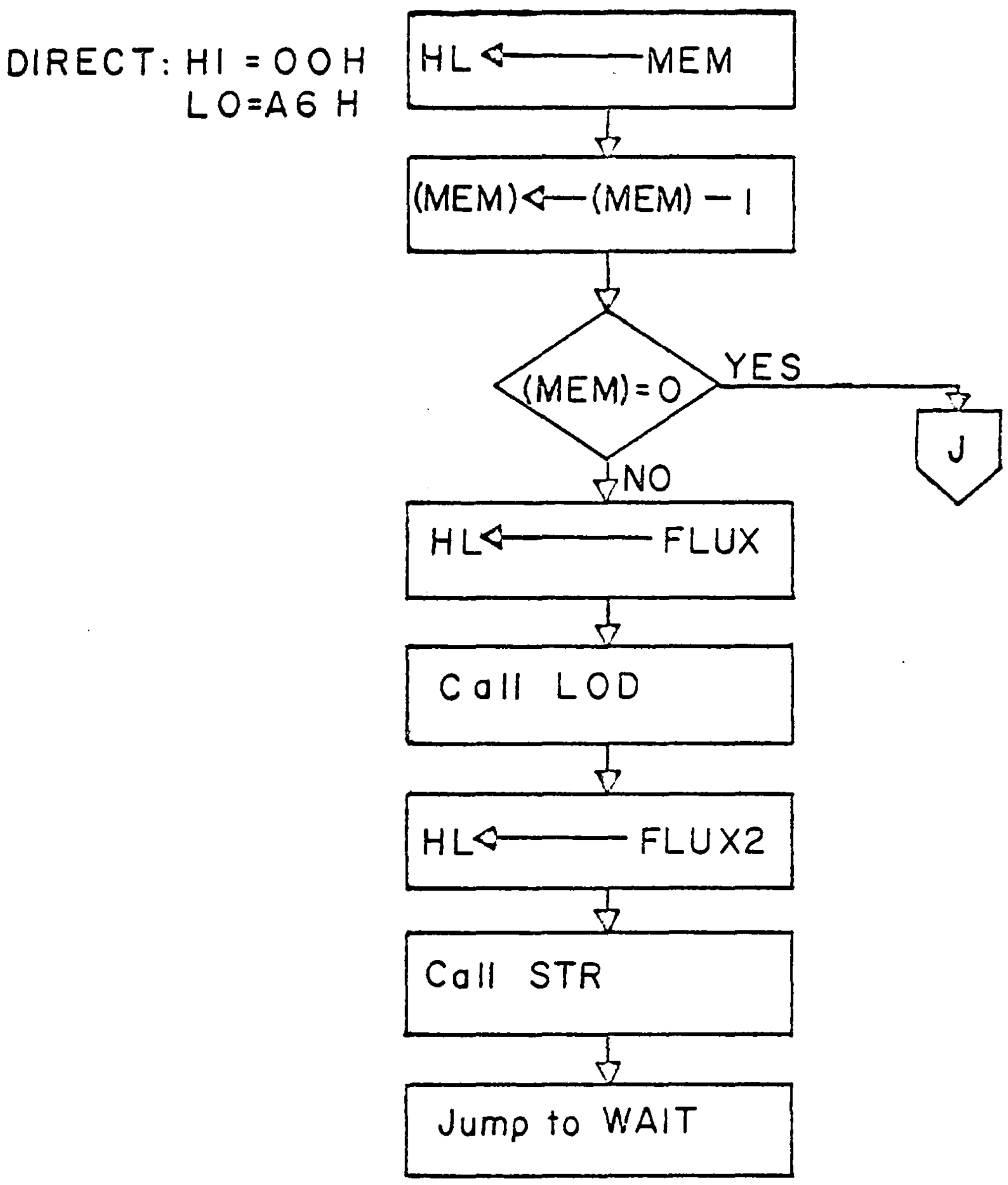


Fig. 5. 6 A DIRECT Routine

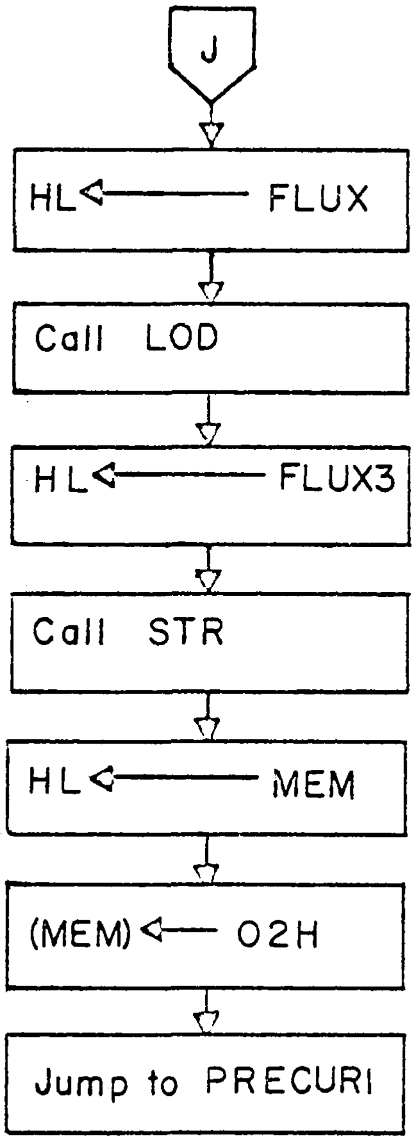


Fig. 5.6B DIRECT Routine (cont.)

(see Fig. 5.7 for flowchart), a temporary four-byte section of memory is required to store a result from a multiplication operation. This section in memory is labeled HOLD and is used twice by each PRECURi routine to store the result from one multiplication operation while another multiplication operation is being performed. The two results are added later.

The first half of the calculation is based on the use of Simpson's Rule to determine the value of the new neutron flux introduced since the last reactivity calculation. FLUX1, $\phi(t - 2\ \Delta t)$, is loaded into the floating-point accumulator and multiplied by CEX1i, $\exp(-\ 2\ \lambda_i \Delta t)$. This result is stored in the address HOLD to permit the next multiplication to be performed. FLUX2, $\phi(t - \Delta t)$, is loaded into the floating-point-accumulator and is multiplied by CEX2i, $4\ \exp(-\ \lambda_i \Delta t)$. The result of the multiplication appears in the floating-point-accumulator. HOLD, FLUX3, ($\phi(t)$), and CLAMI ($\lambda_i \Delta t/3$) are each loaded as operands, and two additions and one multiplication are performed, respectively [CLAMI * (FLUX1 * CEX1i + FLUX2 * CEX2i + FLUX3)]. The final result of the above calculation is stored in HOLD while the multiplication of Y2Ti, the previous neutron precursor flux, and CEX1i is performed. HOLD is added to Y2Ti to obtain the new neutron precursor flux at time t. The result is stored in memory location Y2Ti for the next neutron precursor calculation (for the reactivity calculation at time $t + 2\ \Delta t$). The present neutron precursor flux is still in the floating-point-accumulator. A multiplication is performed with Ai,

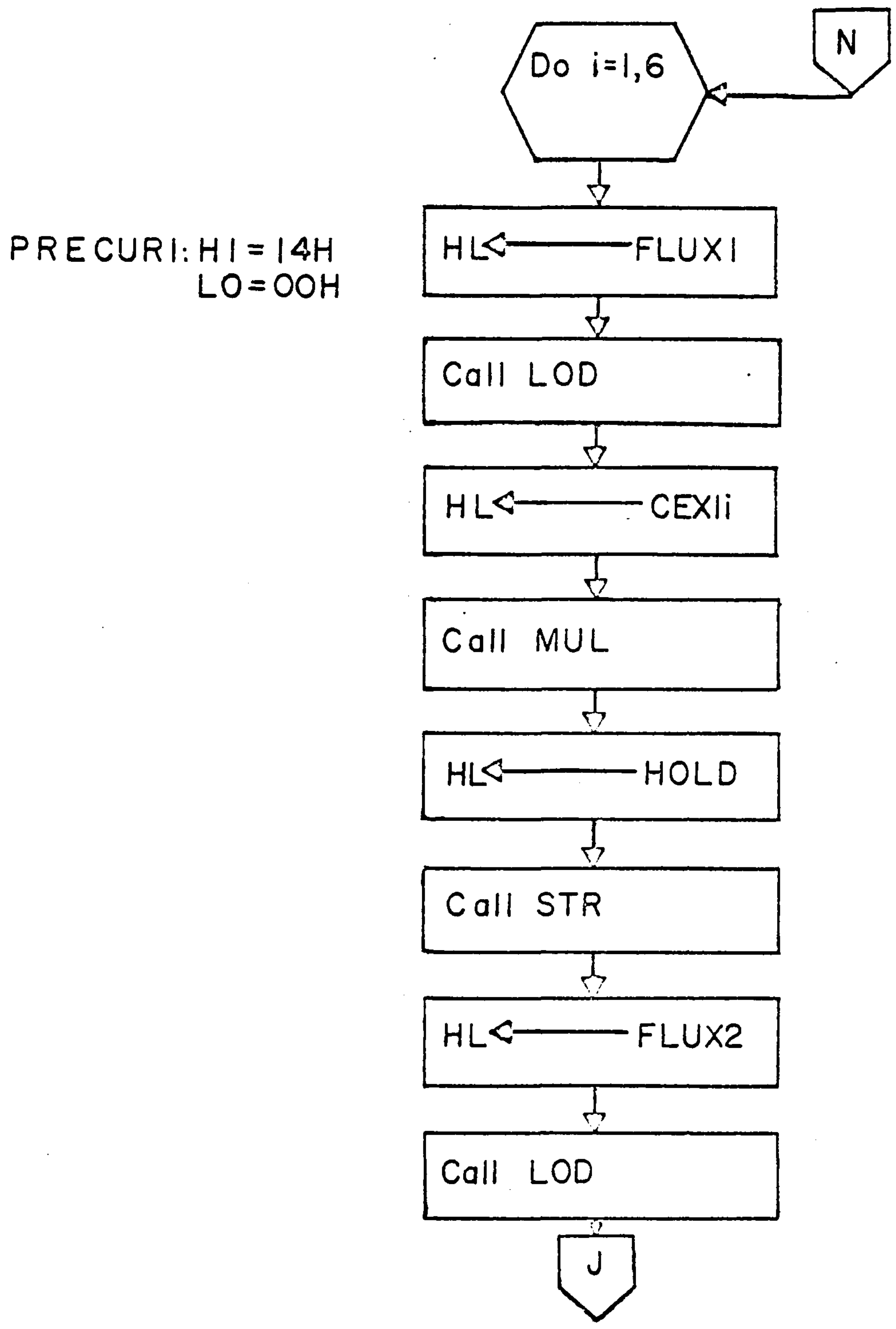


Fig. 5.7A PRECURi Routines

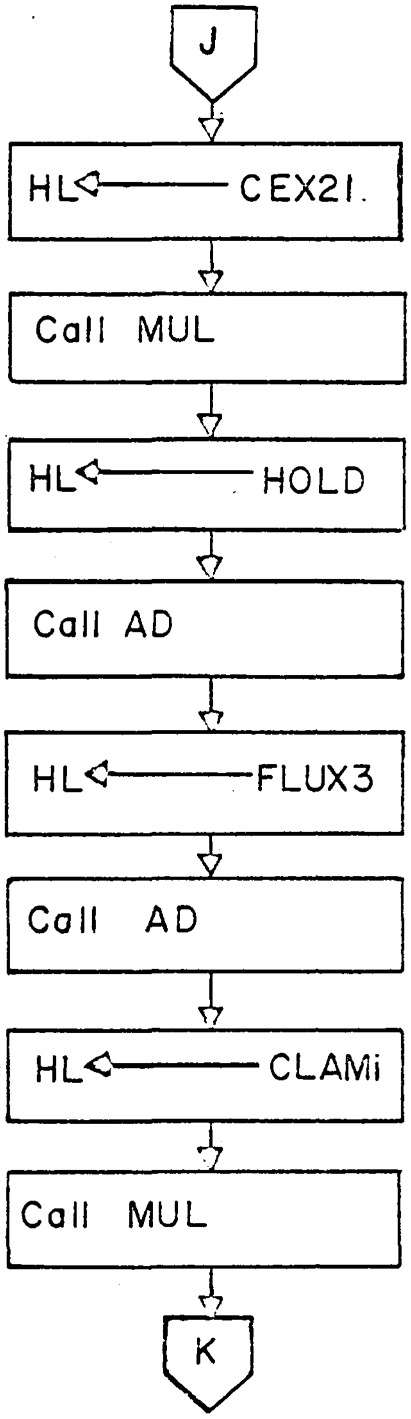


Fig. 5.7 B PRECURi Routines (cont.)

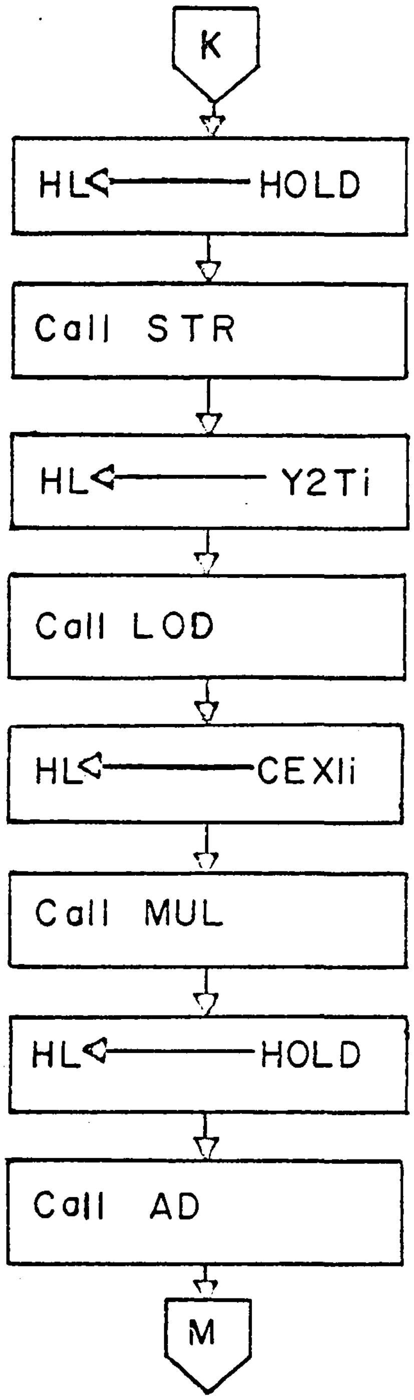


Fig. 5.7C PRECURi Routines (cont.)

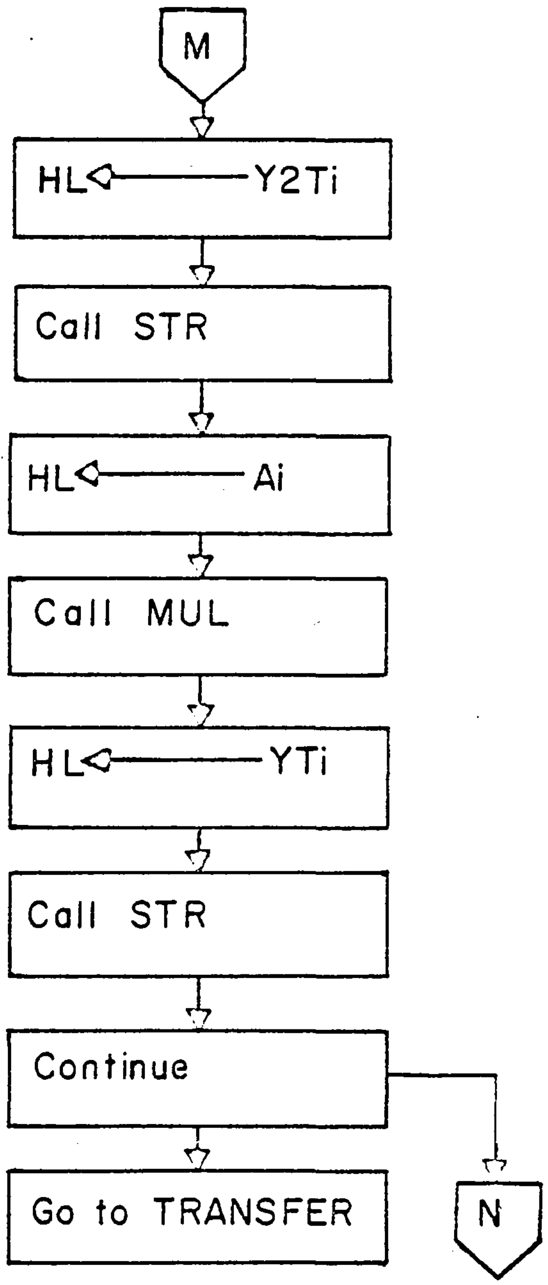


Fig. 5.7 D   PRECURi Routines   (cont.)

the relative fractional yield, with the result being stored in YTi. The YTi's are summed together in REACT, which calculates the reactivity.

TRANSFER

TRANSFER (see Fig. 5.8 for flowchart) is a small routine that assigns the last flux measurement, FLUX3, to FLUX1 in preparation for a new calculation in each of the six PRECURi routines. In the reactivity algorithm, an integral is evaluated by using Simpson's Rule, which requires three points. The last point of one area section becomes the first point of the next successive area section. This program is very simple since all that is done is to load FLUX3 into the floating-point-accumulator and output it to FLUX1.

REACT

REACT performs the final calculation for determining the reactivity of the reactor (see Fig. 5.9 for flowchart). In order to sum the relative neutron precursor yields $a_i Y_i(t)$, labeled in memory as YTi, YT1 is loaded into the floating-point-accumulator and YT2 is loaded into the operand. The floating-point AD subroutine is called, with the result of the operation appearing in the floating-point-accumulator. Similarly, YT3, YT4, YT5, and YT6 are added one at a time in the same way through the use of the proper address for the operand in the HL register pair. The address of FLUX3 is loaded and the floating-point DIV subroutine is called, with the result

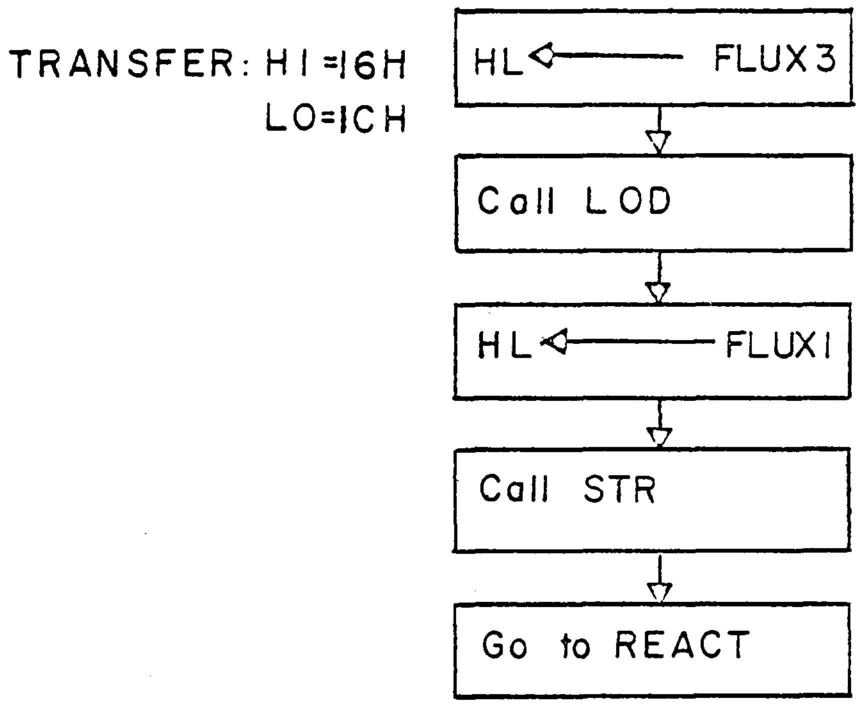


Fig. 5.8 TRANSFER Routine

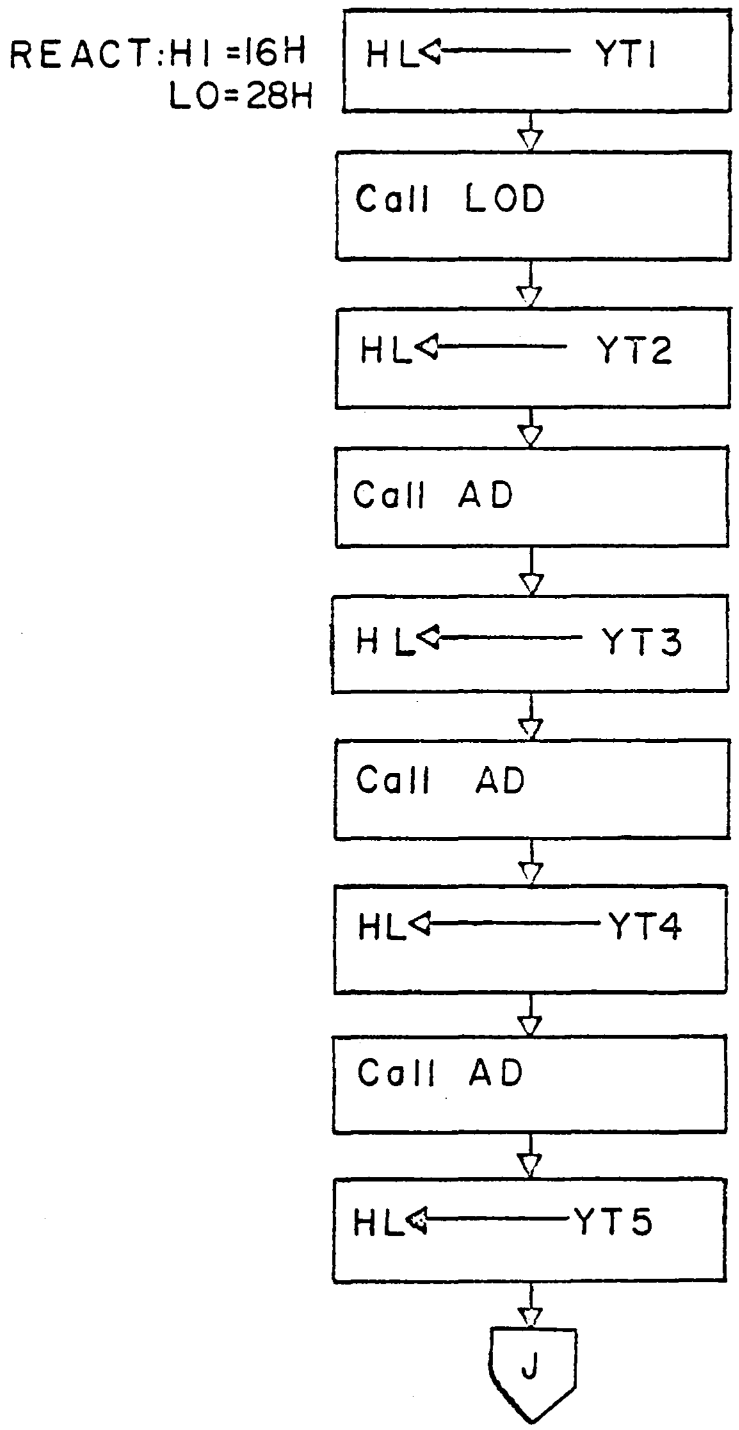


Fig. 5.9 A REACT Routine

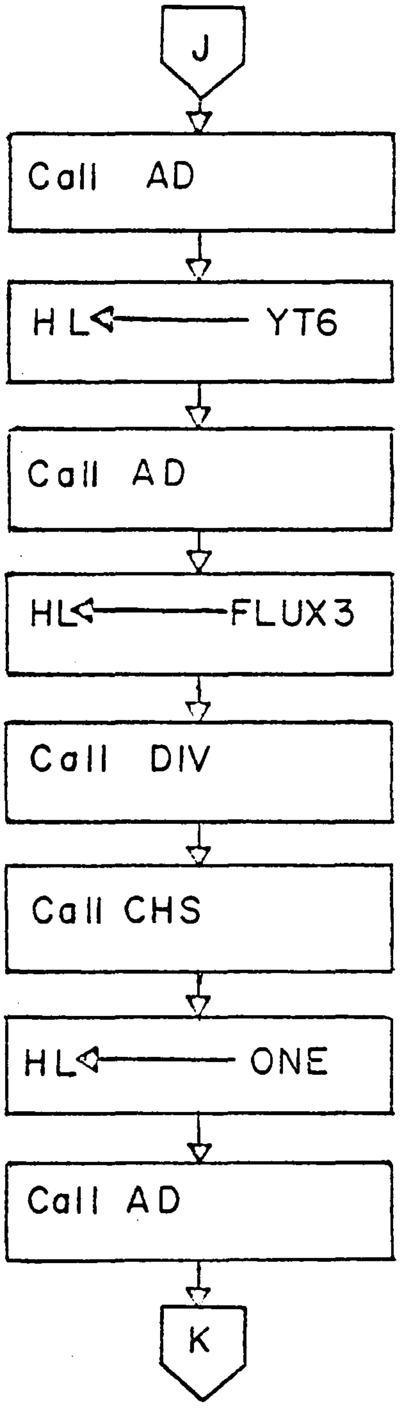


Fig. 5.9B REACT Routine (cont.)

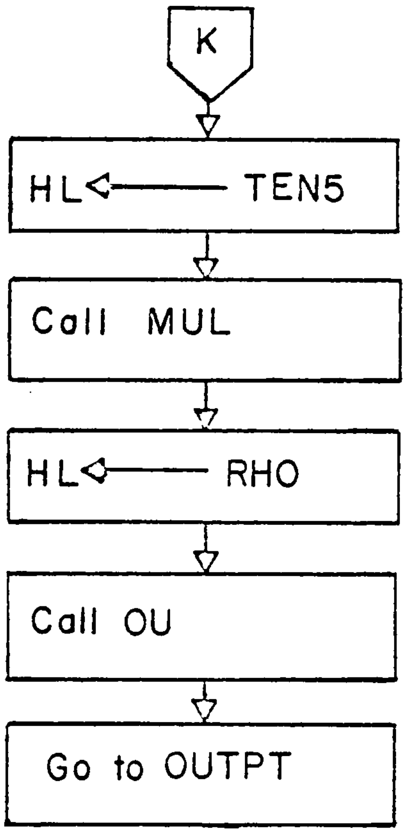


Fig. 5.9 C REACT Routine (cont.)

being stored in the floating-point-accumulator. The result is the sum of the relative yields of each neutron precursor divided by the neutron flux at time t. This result must be subtracted from one to obtain the final answer which is the reactivity. One method to calculate the reactivity would be to output the result to a storage place in memory, load the number one into the floating-point-accumulator, and call the floating-point SB subroutine to subtract it from one. This operation requires 24 bytes of memory. A second method is to leave the result in the floating-point-accumulator, change the sign of the result with the floating-point CHS subroutine, and then use the AD subroutine to add one to obtain the reactivity. This operation requires nine bytes of memory, which is a savings of greater than 50% in memory space and a couple of milli-seconds of time. Of the two methods mentioned above, the latter is more efficient and is employed in REACT. The units of RHO are dollars, which is a very large quantity to measure reactivity. The more common units are per cent mills. One dollar equals 650 per cent mills. Since RHO is desired in per cent mills, after one is added, RHO is multiplied by 650. REACT stores the final result in RHO in a binary-coded-decimal form that requires thirteen bytes of memory for the decimal representation. RHO is the final derived value. REACT passes control to OUTPUT, which transfers RHO to a digital display.

OUTPUT

The purpose of OUTPUT is to locate the decimal point in the computed reactivity and transfer it to the digital display. Most reactor operators prefer to detect changes in reactivity of 1 to 10 pcm, but no smaller. OUTPUT does not immediately transfer the computed reactivity to the display latches because it has to make a decision and a search (see Fig. 5.10 for flow chart). On entry into OUTPUT, the decision that must be made is whether or not the absolute value of the computed reactivity is less than 1 pcm. If so, the value that is transferred to the digital display is zero. If the value is not less than 1 pcm, then a search for the decimal point is performed. Only numbers whose absolute value are less than 0.1 or greater than or equal to 1×10^7 are written in scientific notation by the floating point subroutine OU. The subroutine OU converts binary floating-point numbers into a 13-byte BCD character-string representation (see Appendix B). The first byte is the sign of the number. The next eight bytes are seven significant digits and the decimal point. The last four bytes represent the exponent; if it exists, 025_8 (ASCII code for E) appears in the first byte. The last three bytes are the sign and a two digit exponent. If there is no exponent, all four bytes are set to 360 (ASCII code for space) and the number is not written in scientific notation. Table 5.5 illustrates the different forms of the 13-byte floating point representation. The subroutine OU is used in the routine REACT, but the results are interpreted by OUTPUT.

All bytes are referenced from the first byte in the 13-byte representation of decimal numbers. For instance, the computed

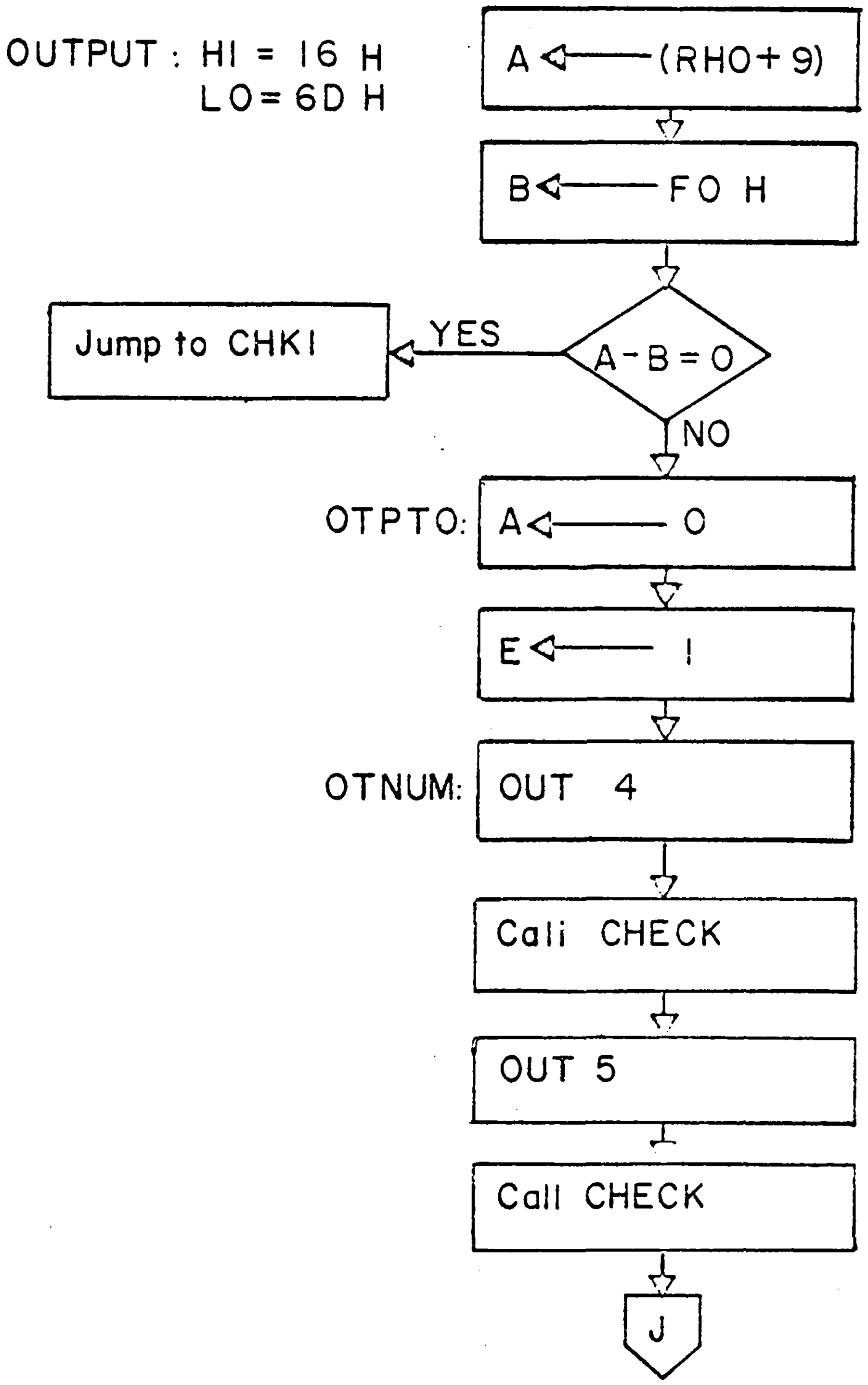


Fig. 5.10 A OUTPUT Routine

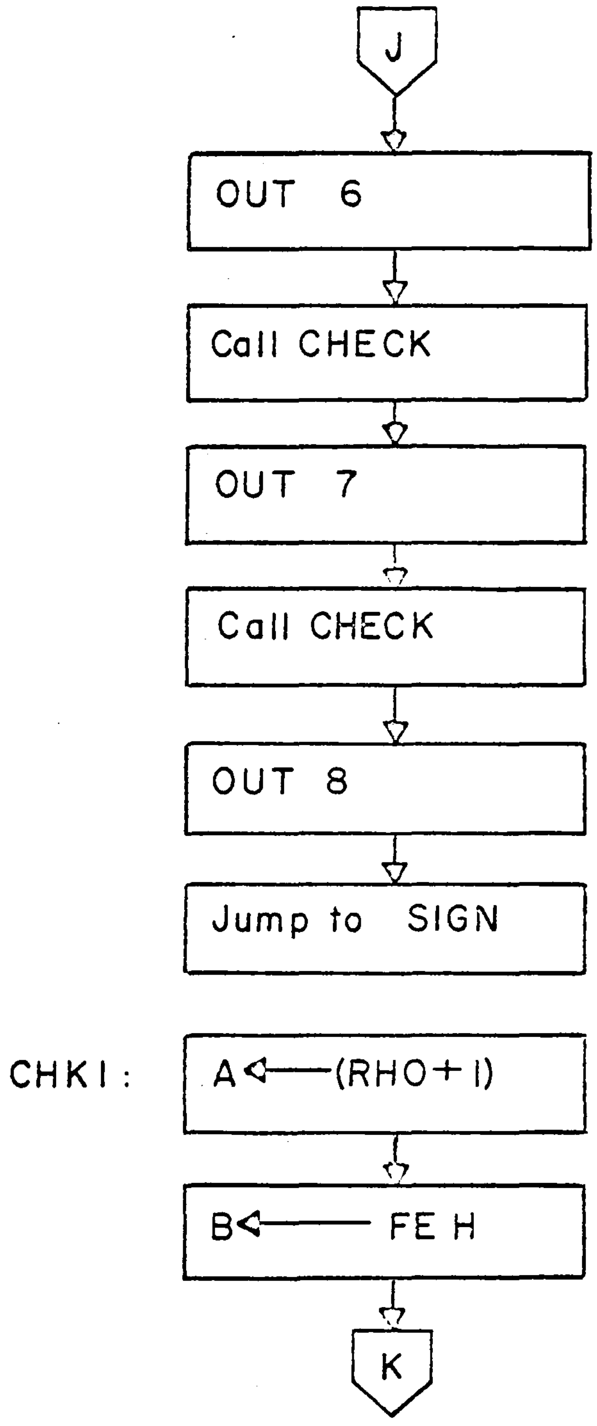


Fig. 5.10 B OUTPUT Routine (cont.)

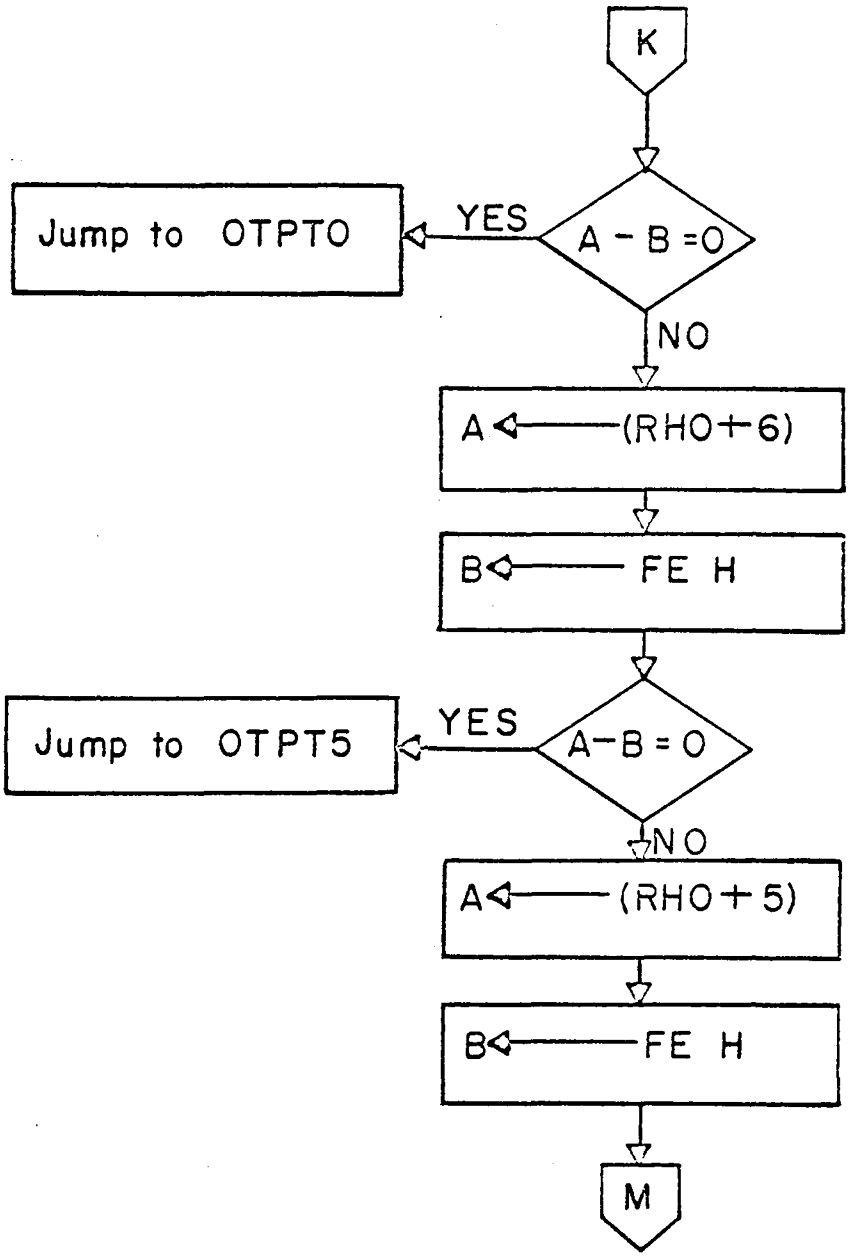


Fig. 5.10 C OUTPUT Routine (cont.)

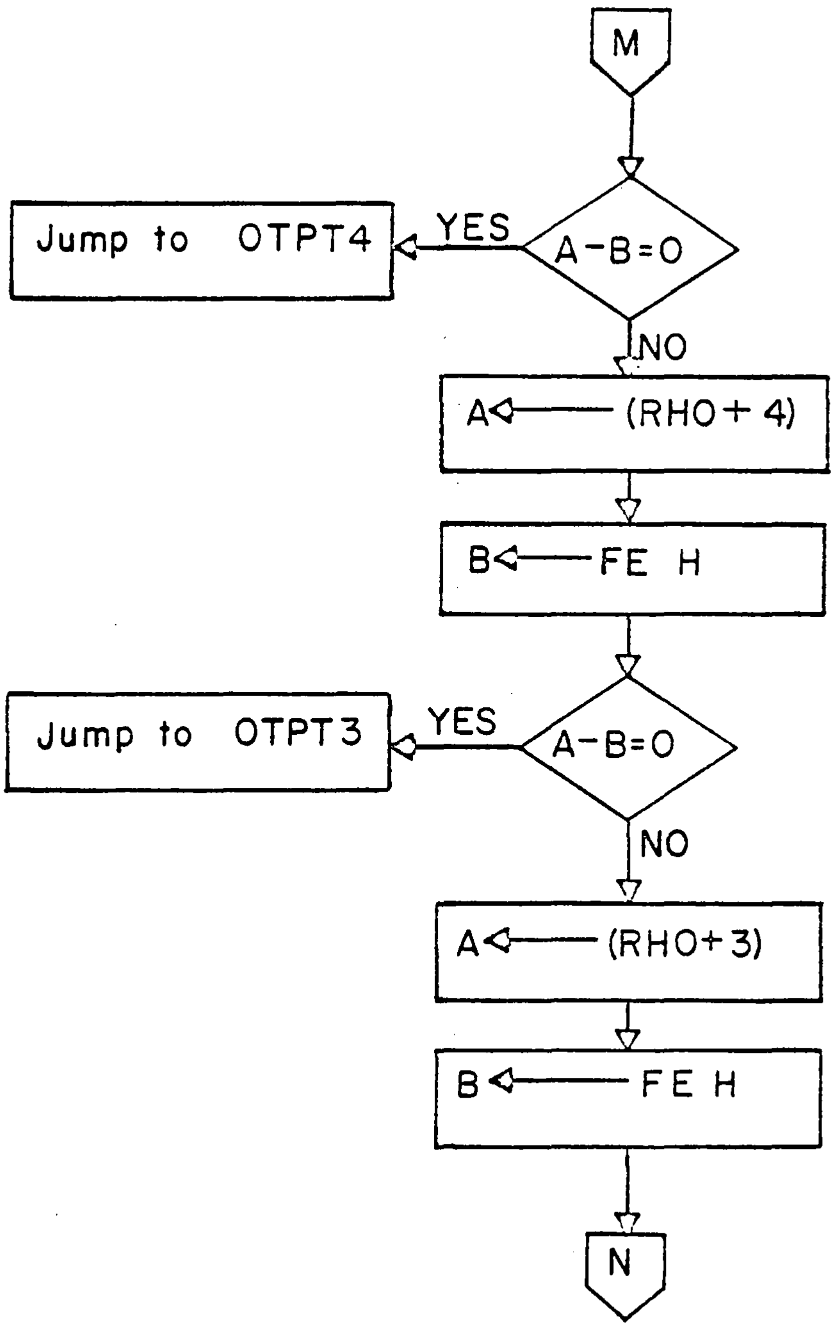


Fig. 5.10 D OUTPUT Routine (cont.)

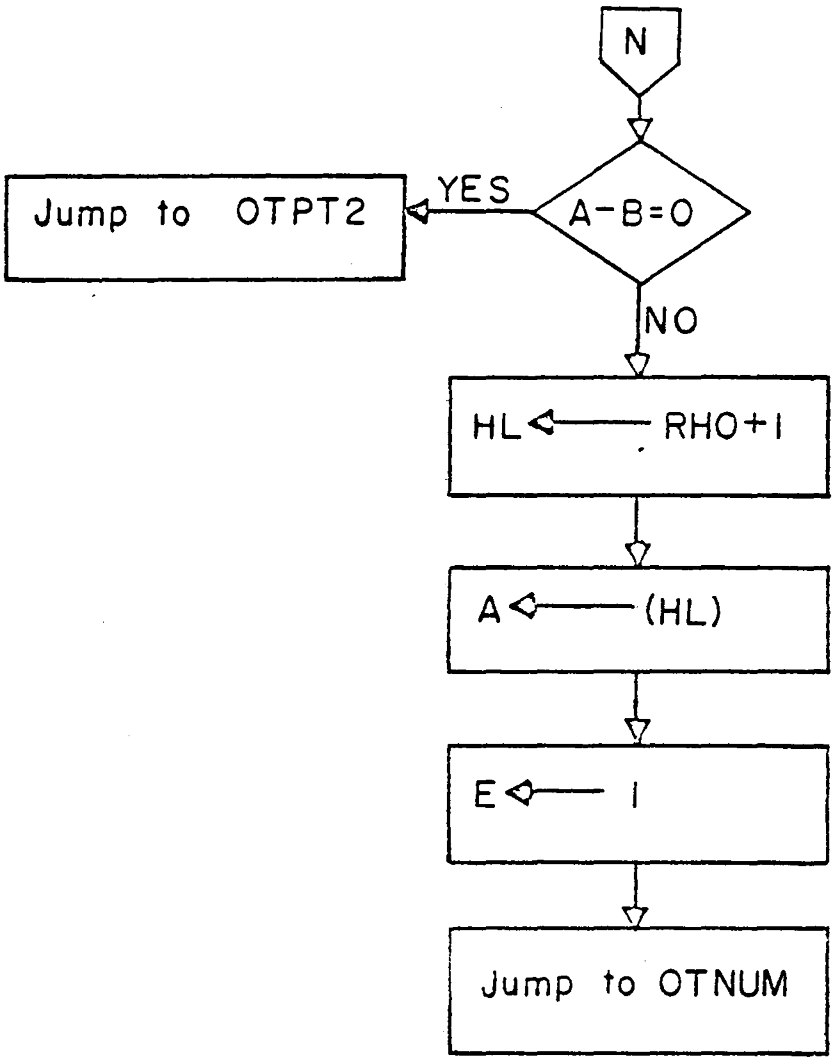


Fig. 5.10 E OUTPUT Routine (cont.)

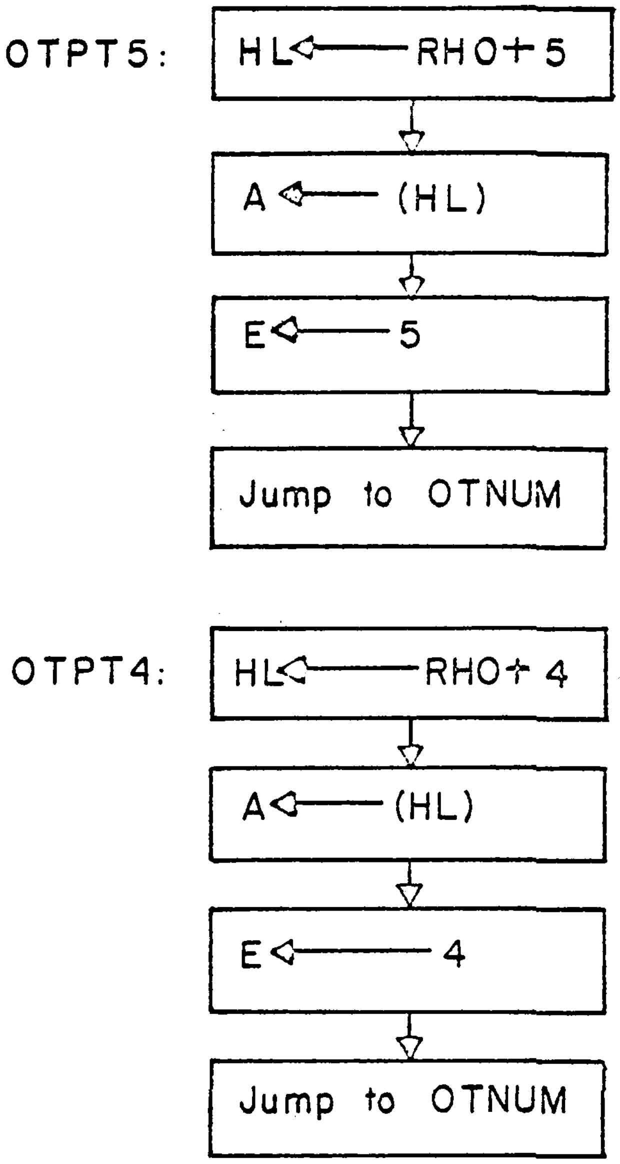


Fig. 5.10F OUTPUT Routine (cont.)

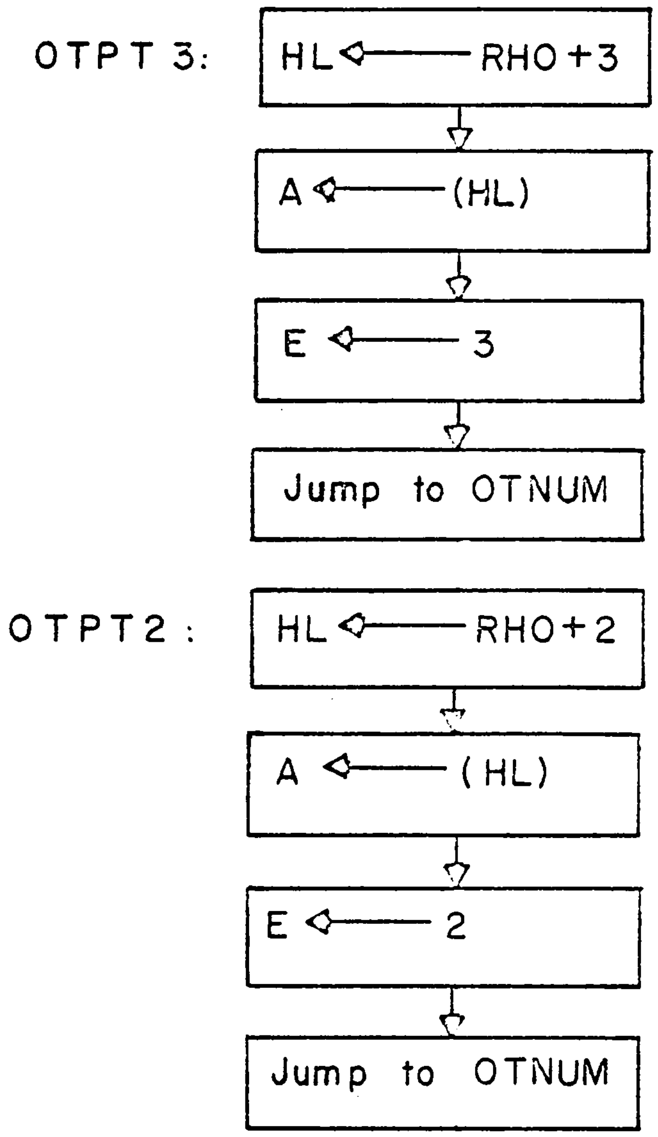


Fig. 5.10 G OUTPUT Routine (cont.)

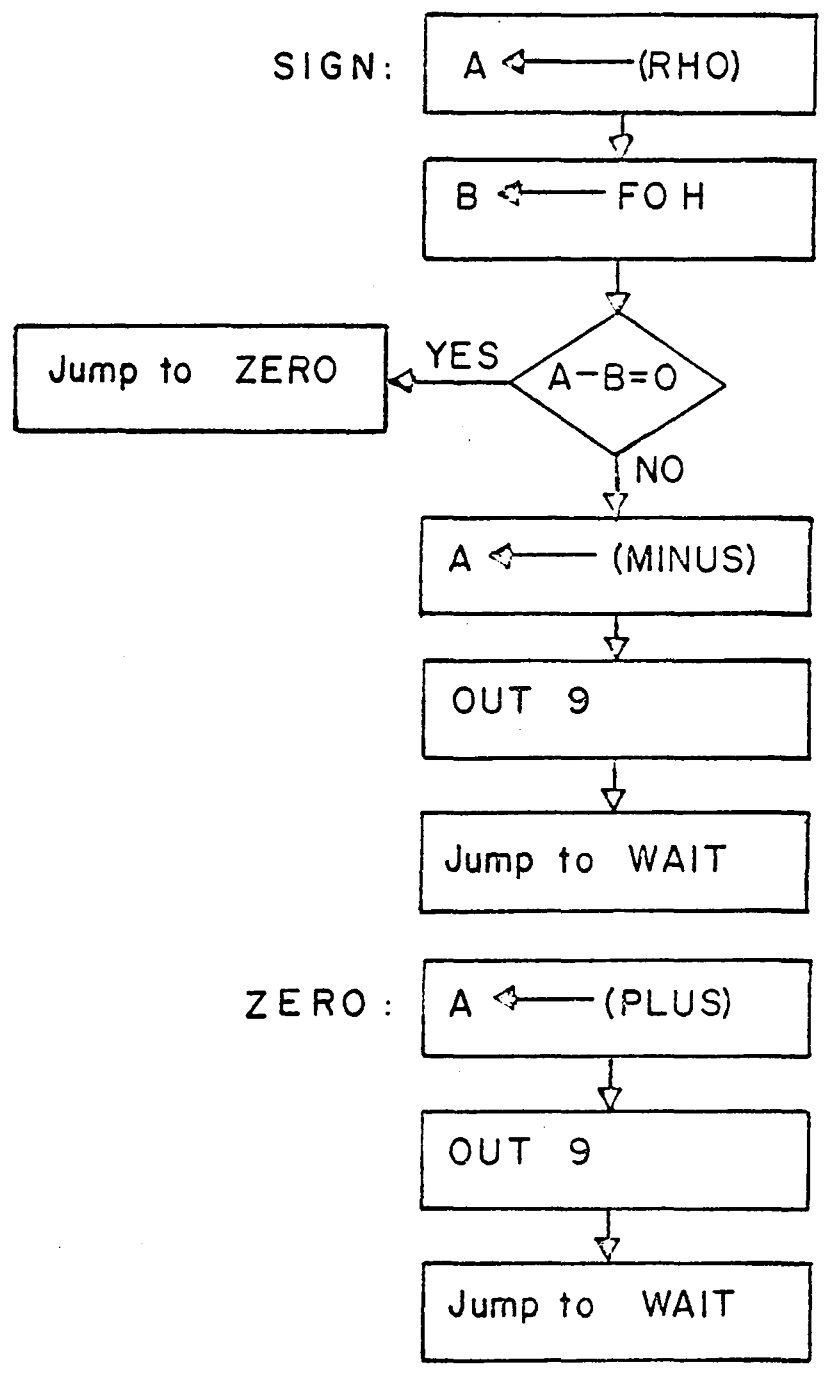


Fig. 5.10 H OUTPUT Routine (cont.)

Table 5.5

ASCII Coded 13-Byte Representation of Floating-Point-
Decimal Numbers

ASCII Code in Octal	Definition
000	0
001	1
002	2
003	3
004	4
005	5
006	6
007	7
008	8
009	9
360	space
374	plus
375	minus
376	decimal point
025	E

Table 5.5

ASCII Coded 13-Byte Representation of Floating-Point-Decimal Numbers (continued)

Floating Point Number	13-Byte Representation												
1.0×10^{-3}	360	001	376	000	000	000	000	000	000	025	375	000	003
1.5×10^{-2}	360	001	376	005	000	000	000	000	000	025	375	000	002
2.75×10^{-1}	360	376	002	007	005	000	000	000	000	360	360	360	360
4.5×10^{0}	360	004	376	005	000	000	000	000	000	360	360	360	360
-4.5×10^{0}	375	004	376	005	000	000	000	000	000	360	360	360	360
1.2×10^{1}	360	001	002	376	000	000	000	000	000	360	360	360	360
2.9×10^{2}	360	002	011	000	376	000	000	000	000	360	360	360	360
5.85×10^{3}	360	005	010	005	000	376	000	000	000	360	360	360	360
6.789×10^{4}	360	006	007	010	011	000	376	000	000	360	360	360	360
8.0×10^{5}	360	010	000	000	000	000	000	376	000	360	360	360	360
1.5×10^{6}	360	001	005	000	000	000	000	000	376	360	360	360	360
1.75×10^{7}	360	001	376	007	005	000	000	000	000	025	374	000	007
1.05×10^{12}	360	001	376	000	005	000	000	000	000	025	374	001	002

reactivity is converted into a 13-byte BCD string and is labeled RHO in the reactimeter program. To reference byte 10 of RHO, the exponential byte, one refers to it as RHO + 9. This is the first of two bytes that are checked to determine if the computed reactivity is less than 1 pcm in absolute value. The largest reactivity magnitude will never exceed 1000 pcm. Therefore, any value of RHO that has an exponent, 025_8 appearing in byte RHO + 9 is a number that is less than one. In such a case, the routine OUTPUT transfers zero to the digital display. If a space instead of an E is found in RHO + 9, RHO + 1 is checked as the second part of the test. A decimal point in RHO + 1 signifies a number greater than or equal to 0.1 and less than 1.0. Again the value of zero is transferred to the digital display (see Fig. 5.10 for flowchart).

If the test determines that the value of the computed reactivity is not less than 1 pcm in absolute value, then a search is conducted for a decimal point in bytes RHO + 6 through RHO + 3. RHO + 6 is the first byte checked. If one is found, then RHO + 5 through RHO + 1 are transferred to the ones through ten-thousands digit displays, respectively. If no decimal point is found, then RHO + 5 is checked. If RHO + 5 contains a decimal point, then RHO + 4 through RHO + 1 are transferred respectively to the ones through thousands digit displays, and the ten-thousands digit display is blanked. If no decimal point is found, RHO + 4 is checked, then RHO + 3. If no decimal point is found, it is assumed to be in RHO + 2. The decimal point cannot be located in RHO + 7 or RHO + 8, because the numbers represented by a decimal point in these locations are beyond the actual values of reactivity capable of being produced in the VPI & SU reactor. Table 5.6 lists the location of the decimal point, the digital displays with numbers, and the displays that are blanked.

After the test for zero and the decimal point search have been completed, the data is ready for transfer to the digital display. If the value to be transferred is zero, OUTPUT starts OTPTO, which loads the accumulator with zero and proceeds to OTNUM. If the value is not zero, OUTPUT starts at OTNUM. The first value to be transferred has been previously loaded into the accumulator immediately after the location of the decimal point was determined. The number of digits to be transferred had also been stored in Register E at this same time. The data are transferred to the digital display by generating a device select pulse with an OUT instruction and calling the subroutine CHECK after each OUT instruction. Table 5.4 summarizes the peripheral devices and their respective codes.

The purpose of the subroutine CHECK (see Fig. 5.11 for flowchart) is to determine if the next value moved into the accumulator is a number or the code word to blank the display. Register E, which was loaded earlier with the number of significant digits to be latched, is decremented on each entry into the subroutine. If the result is zero, the subroutine calls the subroutine WOUT; if the result is not zero, register L is decremented and the next number is loaded into the accumulator. The subroutine WOUT (see Fig. 5.12 for flowchart) loads the address of BLANK + 1 in the HL register pair and the value

Table 5.6

Decimal Point Location	Location of Decimal Digit	Digit
RHO + 6	RHO + 5	Ones
	RHO + 4	Tens
	RHO + 3	Hundreds
	RHO + 2	Thousands
	RHO + 1	Ten-thousands
RHO + 5	RHO + 4	Ones
	RHO + 3	Tens
	RHO + 2	Hundreds
	RHO + 1	Thousands
	Blank	Ten-thousands
RHO + 4	RHO + 3	Ones
	RHO + 2	Tens
	RHO + 1	Hundreds
	Blank	Thousands
	Blank	Ten-thousands
RHO + 3	RHO + 2	Ones
	RHO + 1	Tens
	Blank	Hundreds
	Blank	Thousands
	Blank	Ten-thousands
RHO + 2	RHO + 1	Ones
	Blank	Tens
	Blank	Hundreds
	Blank	Thousands
	Blank	Ten-thousands
RHO + 1	Zero	Ones
	Blank	Tens
	Blank	Hundreds
	Blank	Thousands
	Blank	Ten-thousands

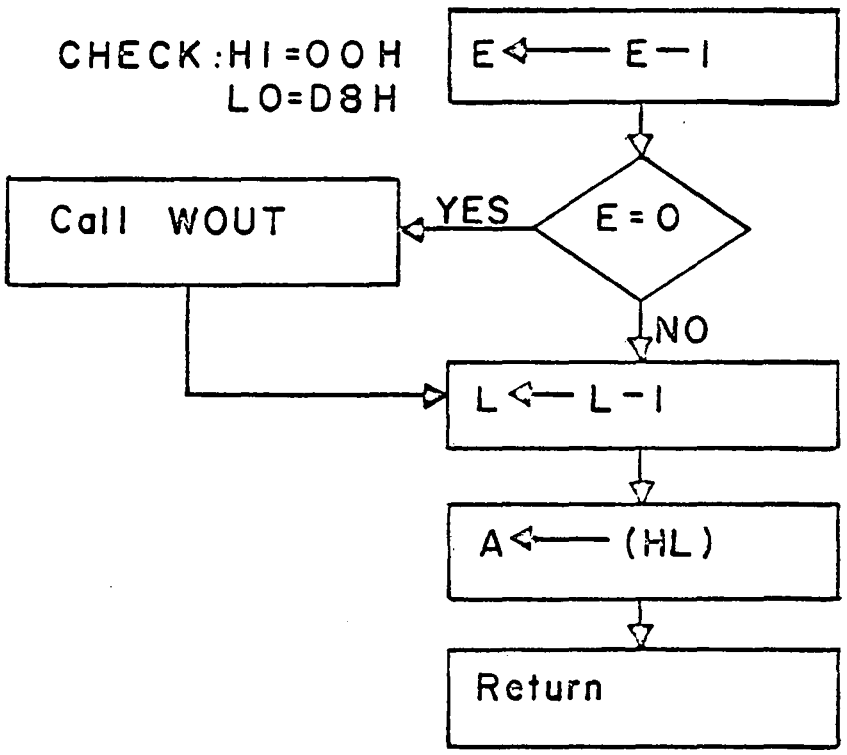


Fig.5.11 CHECK Subroutine

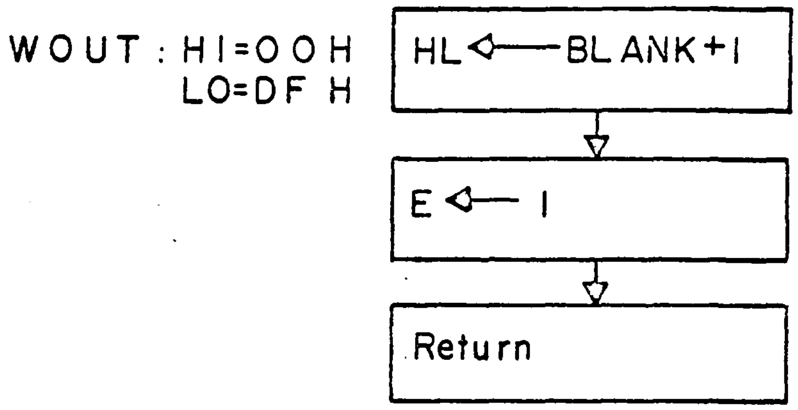


Fig. 5.12 WOUT Subroutine

of one in register E before returning to CHECK. The address directly above BLANK is loaded because, on returning to CHECK, register L is decremented and the address of BLANK is now in the HL register pair. Register E is set to one so subsequent displays can be blanked if required. CHECK is called four times by OTNUM, once after each of the first four OUT instructions.

WAIT

The purpose of WAIT is to let the microcomputer idle until an interrupt signal is received from the external clock (see Fig. 5.13 for flowchart). The external clock generates an interrupt signal every 0.2 seconds and restarts the microcomputer at the address RESTR, which provides a device select pulse to reset the 7490 decade counter that is used to generate the interrupt signal (see Chapter 4).

WAIT receives control from DIRECT if no reactivity calculation is performed, from OUTPUT if a reactivity calculation has been performed, and from START after a power termination. WAIT is a three instruction loop. The first instruction enables the interrupt that permits the external clock to signal the start of a new time interval. The third instruction is a jump to the address occupied by the first instruction. The second instruction is a NOP. The Mark 80 cycles through this loop until an interrupt signal along with a RST 5 is received. The RST command restarts the Mark 80 at HI = 000_8 and LO = 050_8, which is the starting address of RESTR. After clearing the decade counter, the routine INPUT regains control.

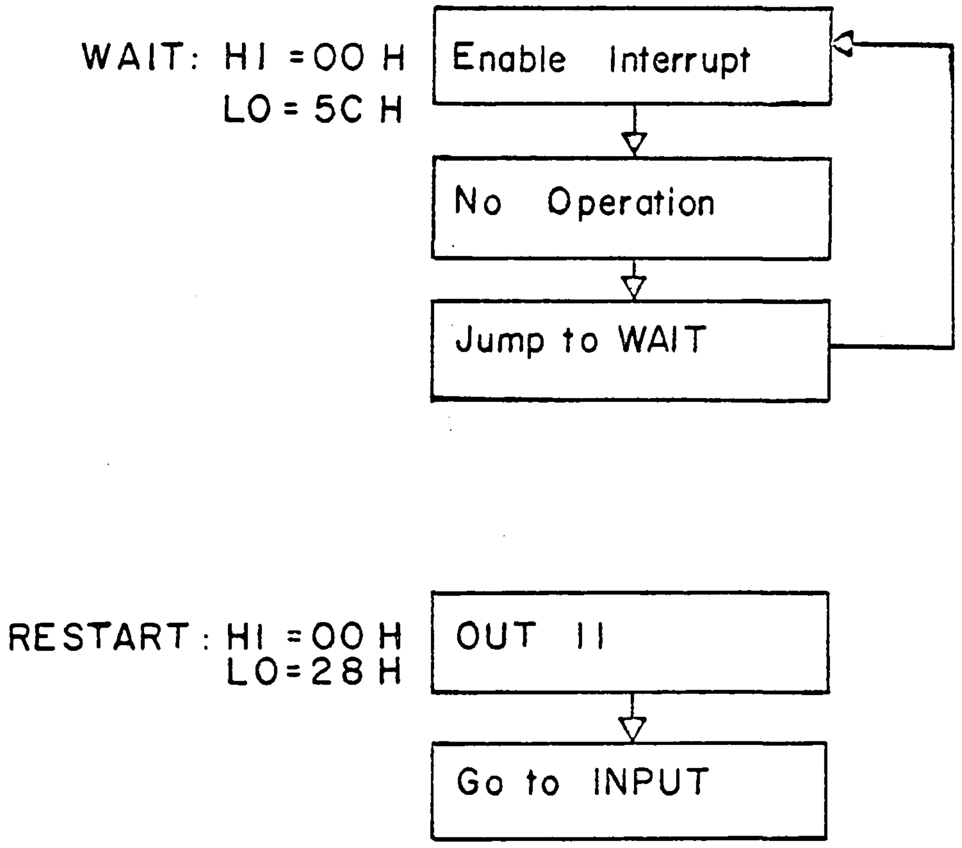


Fig. 5.13 WAIT Loop

START

The START routine is used only once, when the power is first turned on. The entire program can be stored in EPROM, but 256 bytes of R/W memory are required for temporary data: 64 bytes for scratch pad use by the math routines, 64 bytes for use by the Reactimeter program to store input data and results from calculations, and sixteen bytes for stack pointer use. The purpose of START (see Fig. 5.14) is to locate the stack pointer at the high end of R/W memory, insert the ASCII codes for decimal point, space, and end-of-BCD string into the storage section of DMANT, and initialize FLUX1 and Y2Ti (i = 1 to 6) to zero for the first calculation. After this is done, START gives control to WAIT to begin the first calculation.

Conclusion

The development of the software system is based on two algorithms and the constraints governing the hardware system for the input and output of data. The program was assembled using MAC80, a cross assembler on the VPI & SU IBM 360 computer. The MAC80 reads mnemonic symbols and labels typed into a source program and generates a listing program that assigns an address and the correct code to each mnemonic symbol. In conjunction with the reactivity program, the Intel Floating Point Package of basic mathematical subroutines is used to perform mathematical operations. The Intel Floating Point Package is listed in Appendix E. The Reactimeter program is listed in Appendix D. In assembling the Reactimeter program, dummy

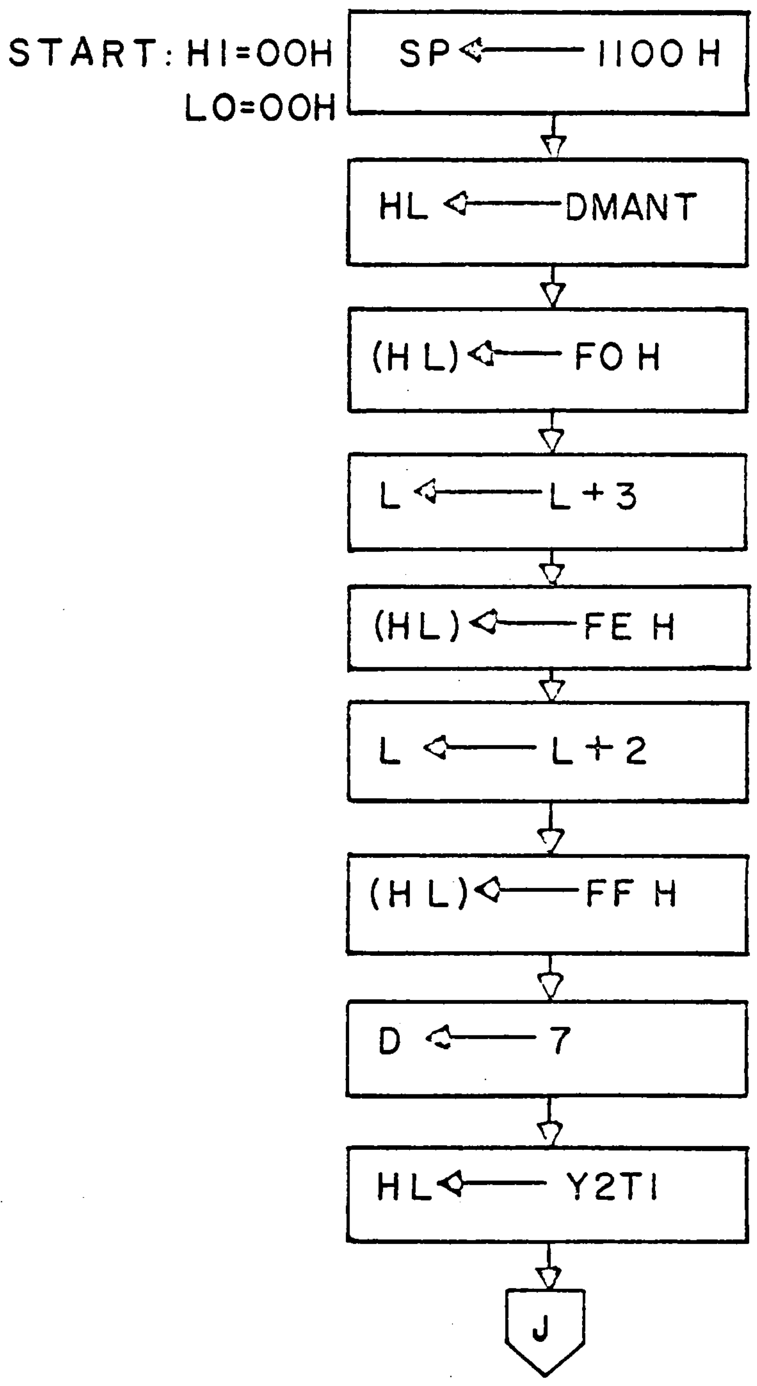


Fig. 5.14 A START Routine

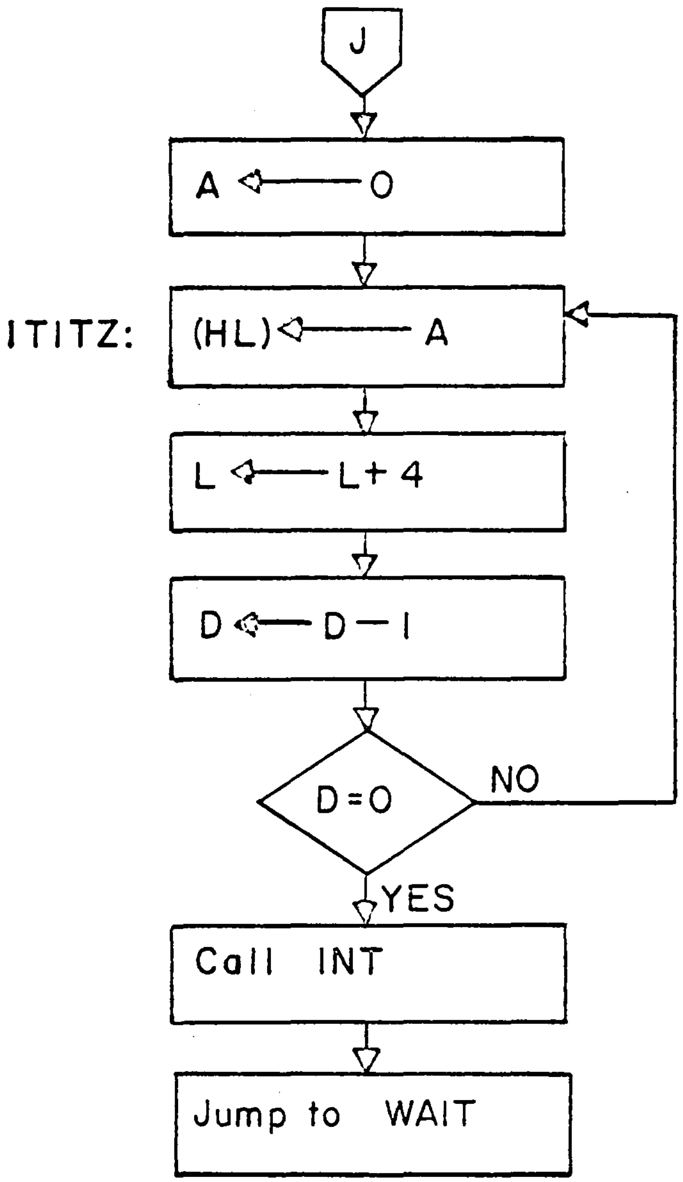


Fig. 5.14 B START Routine (cont.)

routines were placed under the floating-point subroutine names and addresses so that the cross assembler could reference them. The cross assembler generates the address and the code in hexadecimal notation. Once the listing program is free of errors, it can be stored in the Mark 80 and testing can begin. Results of testing are discussed in Chapter 6.

CHAPTER VI

CONCLUSIONS

Conclusion

Figure 6.1 compares the reactor period versus the actual reactivity for the VPI & SU reactor. The point reactor kinetics algorithm developed does not calculate the correct reactivity for certain reactor periods. Table 6.1 summarizes the actual and the calculated reactivity for different reactor periods, the per cent change in flux during a 0.2 second interval, and the change in the digital panel meter reading in 0.4 seconds. These calculations were performed at 10%, 50%, and 90% of scale readings for the 0 to 100 mV range. At 10% of scale, reactor periods greater than 50 seconds have a calculated reactivity of zero. Reactor periods greater than 300 seconds have a calculated reactivity of zero at 90% of scale. It is desired that reactor periods up to 1000 seconds (16 minutes) be detected by the reactimeter. The reason for the inaccurate reactivity calculation for reactor periods longer than thirty seconds is the lack of sensitivity of the digital panel meter in detecting changes of voltage from the Keithley meter. The method developed to calculate reactivity is not wrong, but is as only accurate as the instrumentation supplying the information.

Two solutions are offered to solve the problem. One solution is to change the instrumentation, and the other is to change the method of calculation. Changing the instrumentation involves

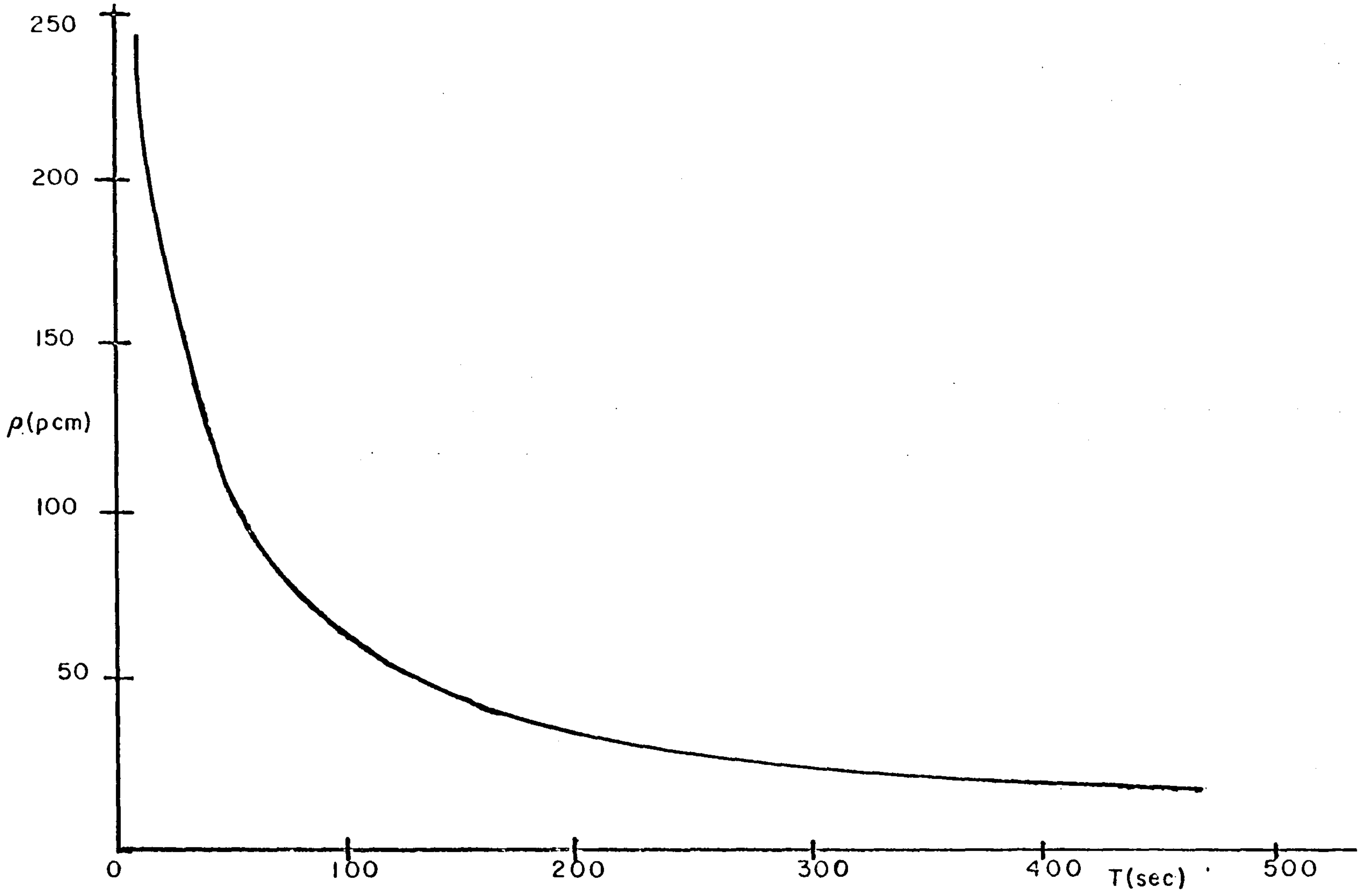


Fig. 6.1 Reactivity versus Reactor Period

Table 6.1

Instrumentation Calibration

ρ_{act} (pcm)	T (sec)	% ϕ in. 0.2 sec	Reading Change of DPM in 0.4 sec			$\rho_{calculated}$		
			10% scale reading	50% scale reading	90% scale reading	10% scale reading	50% scale reading	90% scale reading
337	5	4.1	.8	4.0	7.4	335	334	336
211	15	1.3	0.2	1.3	2.3	180	215	205
144	30	0.7	0.1	0.7	1.2	118	148	145
104	50	0.4	0	0.4	0.7	0	103	102
78	75	0.3	0	0.3	0.5	0	83	80
60	100	0.2	0	0.2	0.3	0	62	55
53	125	0.16		0.1			35	55
45	150	0.10		0.1	0.2		35	39
35	200	0.10		0	0.1		0	21
29	250	0.08			0.1			21
28	300	0.04			0			0

obtaining a more sensitive digital panel meter and is the more costly of the two methods. A more sensitive digital panel meter, capable of detecting a microvolt, could be used to interpret the signal from the Keithley meter. This would allow measurements to be taken every 0.2 seconds and the correct reactivity calculated by the algorithm for reactor periods of 1000 seconds at 10% of scale. With the high sensitivity meter, background noise on the signal line becomes a major concern. If the noise level is greater than the voltage change, it is useless to employ a more sensitive instrument.

The second solution is to change the calculation method. The inhour equation, equation (2.12), provides acceptable results for reactor periods of thirty seconds or greater.

$$\rho_0 = \Lambda\omega + \sum_{i=1}^{6} \frac{\beta_i}{\omega + \lambda_i} \tag{2.12}$$

One of the seven solutions to the inhour equation is $\omega = 1/T$, where T is the reactor period. The inhour equation can be written in the following form,

$$\rho_0 = \frac{\Lambda}{T} + \sum_{i=1}^{6} \frac{\beta_i}{1 + \lambda_i T} \tag{6.1}$$

This algorithm is simpler than the first and is employed in the reactimeter program in Appendix F.

Reactimeter Program

The following circuits discussed in Chapter IV can still be used.

Figure 4.5 Digital Panel Meter Interface Circuit -- No change

Figure 4.6 Zero Input Interface Circuit -- Device select pulse INO is not used. Therefore pin 9 of the 7408 AND gate is set to "logic 1"

Figure 4.8 Digital Panel Meter -- Two changes (see Fig. 6.2) (1) Pin B, the external trigger, is connected to a 100 Hz timing circuit (shown in Fig. 6.3), which generates a conversion pulse to the DPM every 10 msec; (2) Pin 3, Status ($\overline{\text{Print}}$), is used to generate an interrupt instruction (RST 5) to the Mark 80, that signals an update in data (see Fig. 6.4)

Figure 4.9 A Output Interface Circuit displays -- No change

Figure 4.9 B Not required

Figure 4.10 Decoding Circuit -- Channels 0 (pin 1), 9 (pin 10), and 11 (pin 13) are not needed and are unconnected

The inhour reactimeter program works in the following manner. A master flowchart of the inhour program is shown in Figure 6.5. The first measurement on each range setting is used as initial power reference. There is no cross referencing between range settings of the Keithley meter. If a power change of 10% is detected, the previous measurement becomes the new reference. The routine INPUT that transfers data into the Mark 80, has two modifications (see

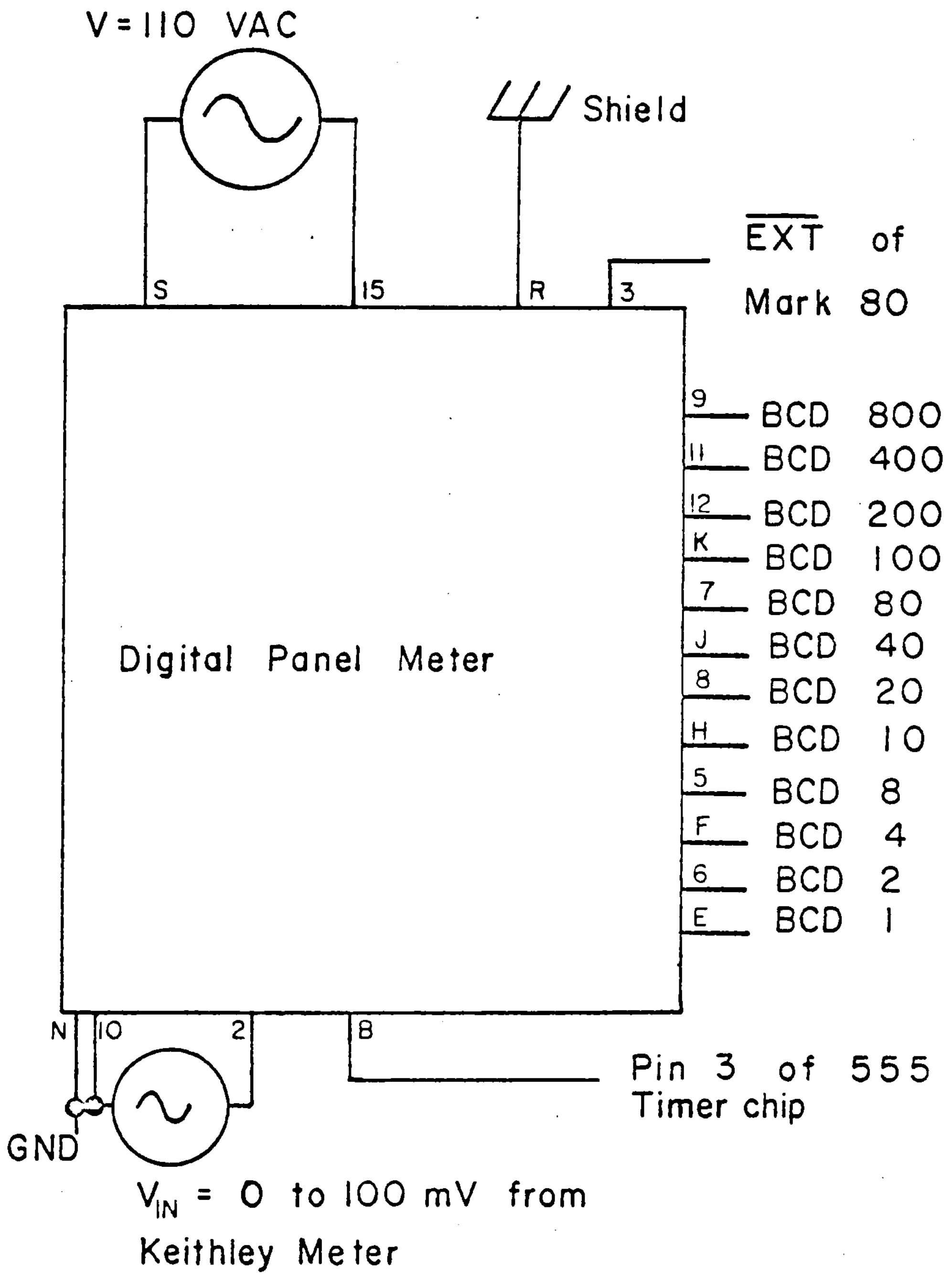


Fig. 6.2 Connection Diagram for Digital Panel Meter - Modified Version

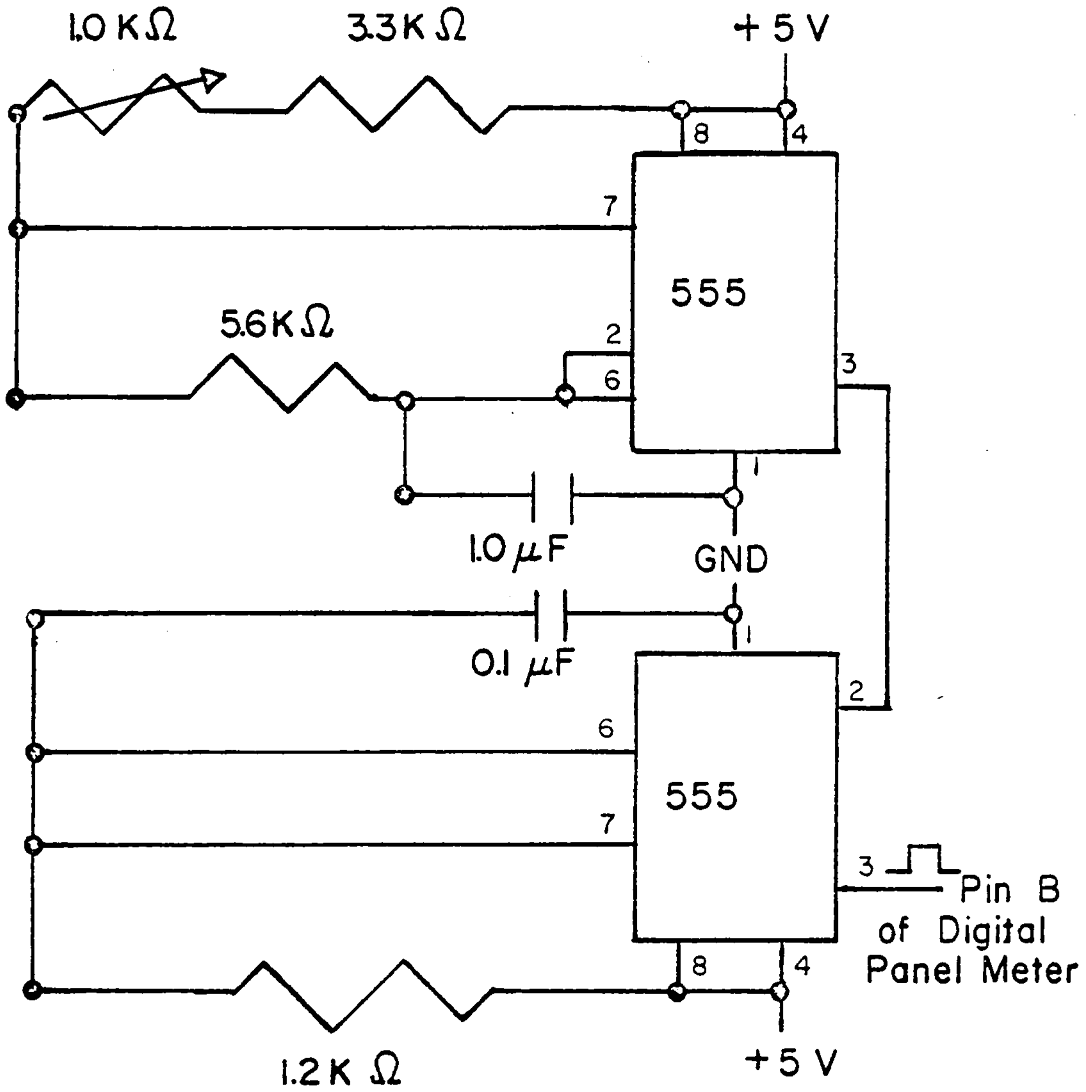


Fig. 6.3 100 Hz Timing Circuit for the Digital Panel Meter

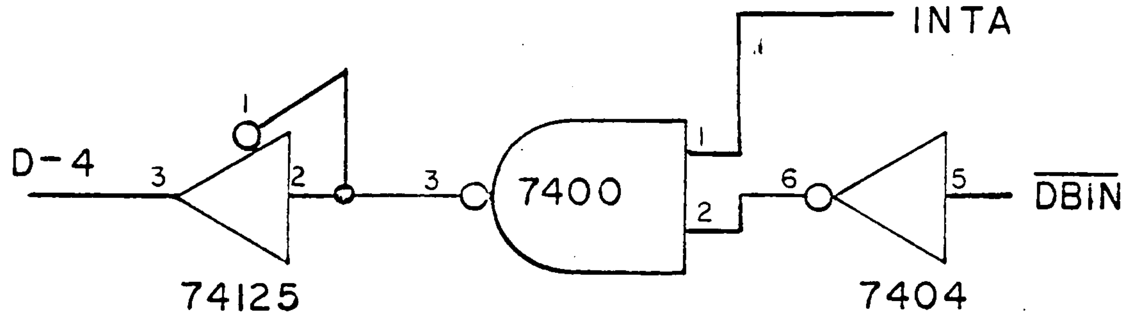


Fig. 6.4 Interrupt Circuit for the Digital Panel Meter

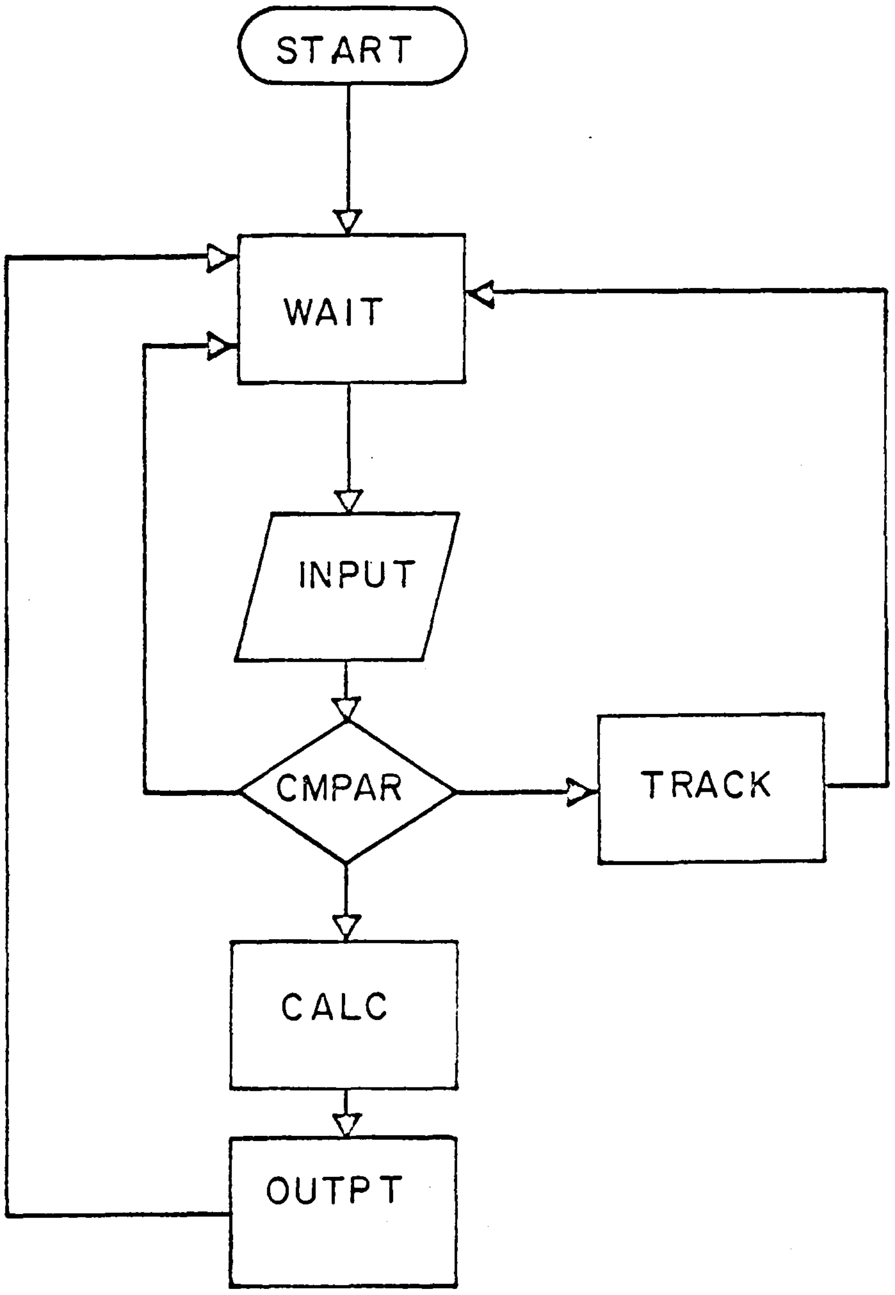


Fig. 6.5 Main Flowchart - Inhour Program

Fig. 6.6). The OUT 11 instruction is not required since the decade counter is not used, and the IN 0 instruction is omitted since no data is transferred from the range dial of the Keithley meter. The purpose of INPUT is to change the input data from BCD form to binary-floating-point form and store it in FLX2.

Measurements are recorded every 10 msec and compared to a reference, FLX1. The purpose of the routine CMPAR (compare) is to determine the per cent power change with respect to the reference power, FLX1 (see flowchart Fig. 6.7). If the power change is less than 1% with respect to the reference power, CMPAR passes control to TRACK. If the power change is greater than 10%, the subroutine EXCH (exchange) is called. On return from the subroutine CMPAR passes control to WAIT (no change, see Chapter V). If the power change is between 1 and 10%, control is passed to CALC (calculate). A change of 10% between two consecutive readings from the DPM implies one of two things; a 10% change in the reactor power, or the range setting of the Keithley meter has been changed. The 10% change in power change can be ruled out, because for a 10% change in power during 10 msec (the time between two consecutive measurements) implies the reactor is on a 0.1 second period, which is an impossible situation. The per cent change is calculated by subtracting FLX1 from FLUX2, and dividing the absolute value of the result by FLX1. After FLX1 is subtracted from FLX2, the subroutine FLAG is called to determine and store the sign of the subtraction result in SGST (sign storage). If a period calculation is required, this is

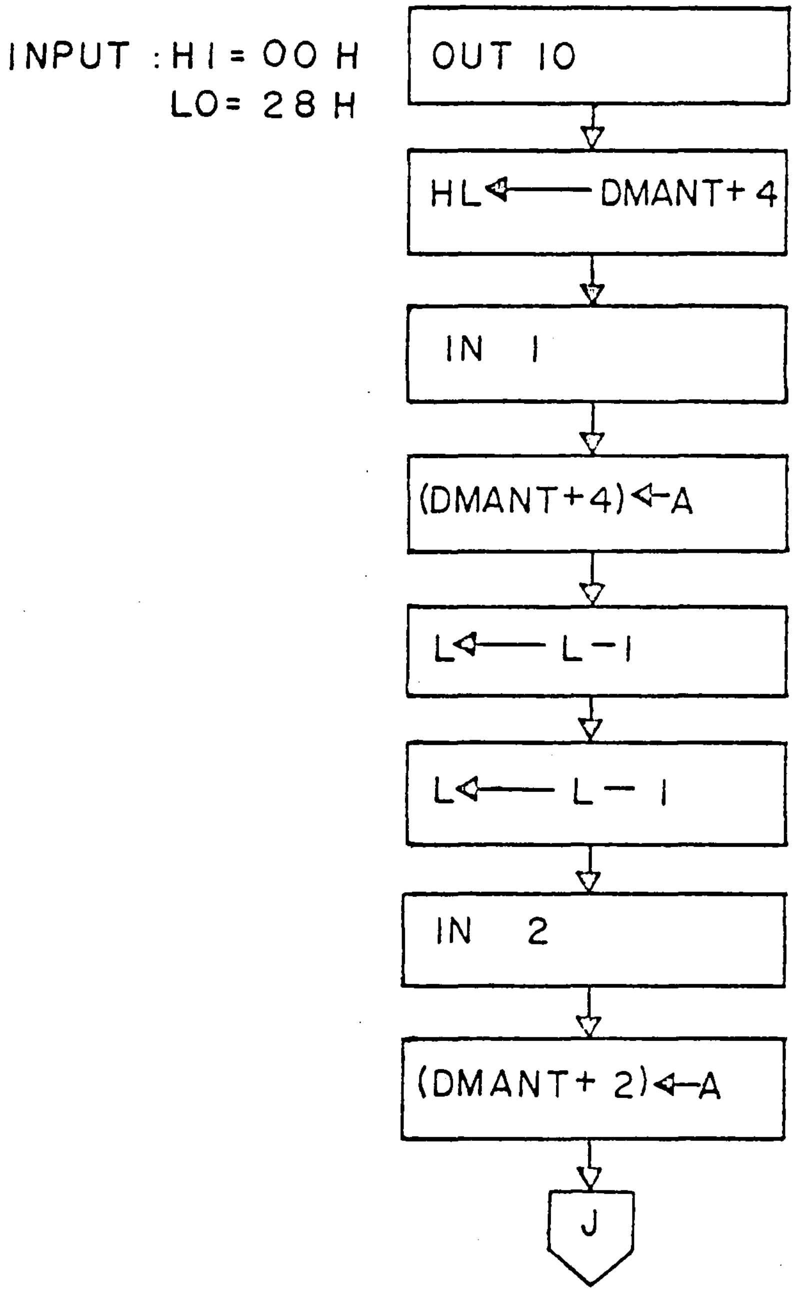


Fig. 6.6 A INPUT Routine

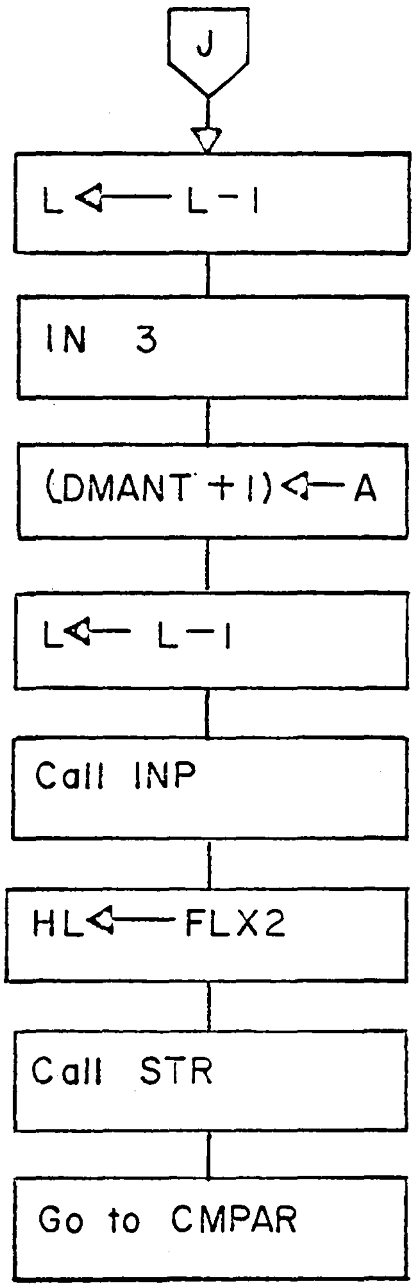


Fig. 6.6 B INPUT Routine (cont.)

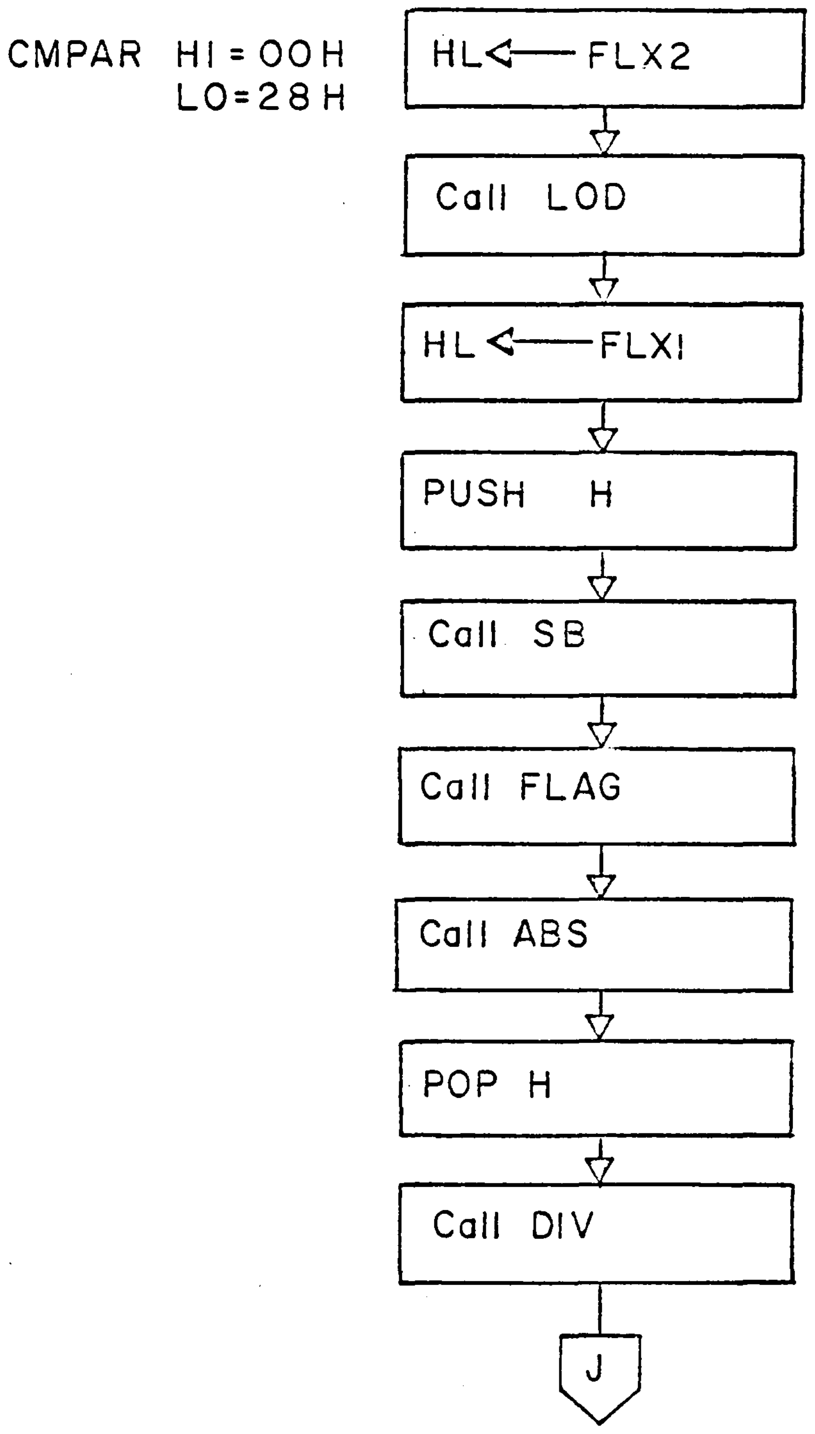


Fig. 6.7 A CMPAR Routine

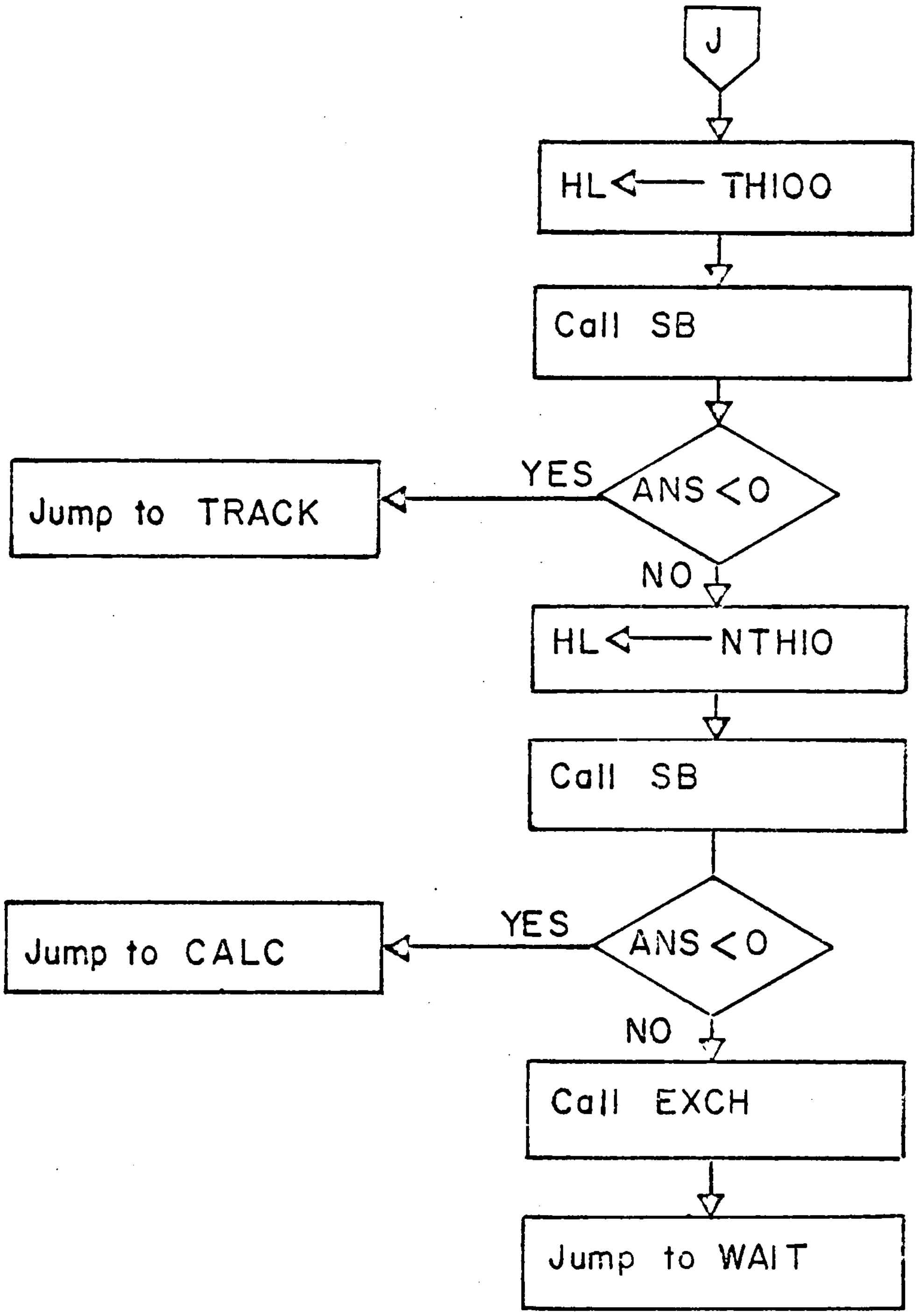


Fig. 6.7 B CMPAR Routine (cont.)

recalled later by CALC. Two tests are performed by subtracting 0.01 and 0.1, respectively, from the absolute value of the per cent change in power. If the result of the subtraction of 0.01 is negative, then control is passed to TRACK. If the result of the subtraction of 0.1 is negative, control is passed to CALC. If neither result is negative, control is passed to EXCH.

The purpose of TRACK (see Fig. 6.8) is to increment a two-byte section of memory, MEM1 and MEM, every time the power change is less than 1%. The number of counts stored in MEM1 and MEM is equal to the reactor period in seconds. Thus if MEM1 equals 0000010_2 and MEM equals 10001001_2, the reactor period is equal to $2^9 + 2^7 + 2^3 + 2^0 + 1 = 650$ seconds. A maximum limit is imposed on the number of counts recorded before a 1% change is detected. If there is no 1% change after 1024 counts (1024 seconds or 17 minutes), TRACK passes control to OUTPT, which transfers the value of zero to the digital displays. OUTPT is the data output routine from Chapter V. Reactor periods longer than 15 minutes have a reactivity that is essentially zero, and this is the reason for the upper limit.

The purpose of EXCH (see Fig. 6.9) is to transfer the most recent measurement, FLX2, into the reference location, FLX1. With a new reference power, the two-byte period counter, MEM1 and MEM, is set to zero. EXCH is called by either CMPAR, if a 10% power change is detected, or by CALC after a reactivity calculation is made.

The routine CALC (see Fig. 6.10) performs only when a reactivity calculation is required. Before calculating the reactivity, CALC

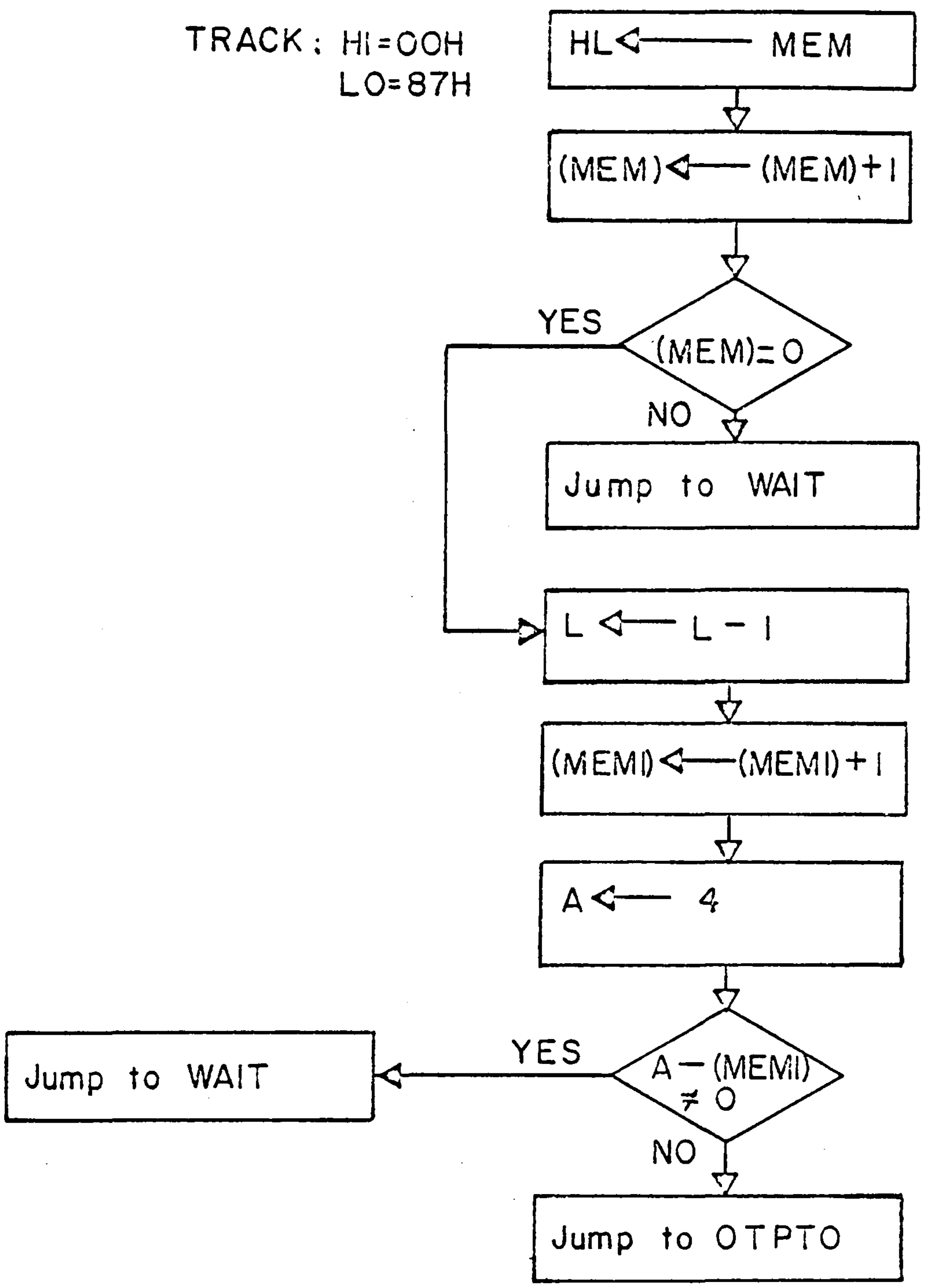


Fig. 6.8 TRACK Routine

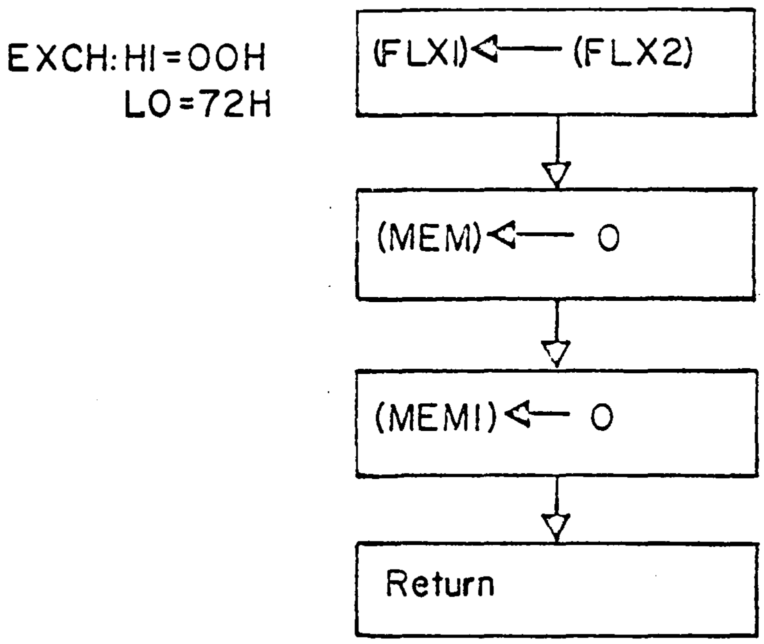


Fig. 6.9 EXCH Subroutine

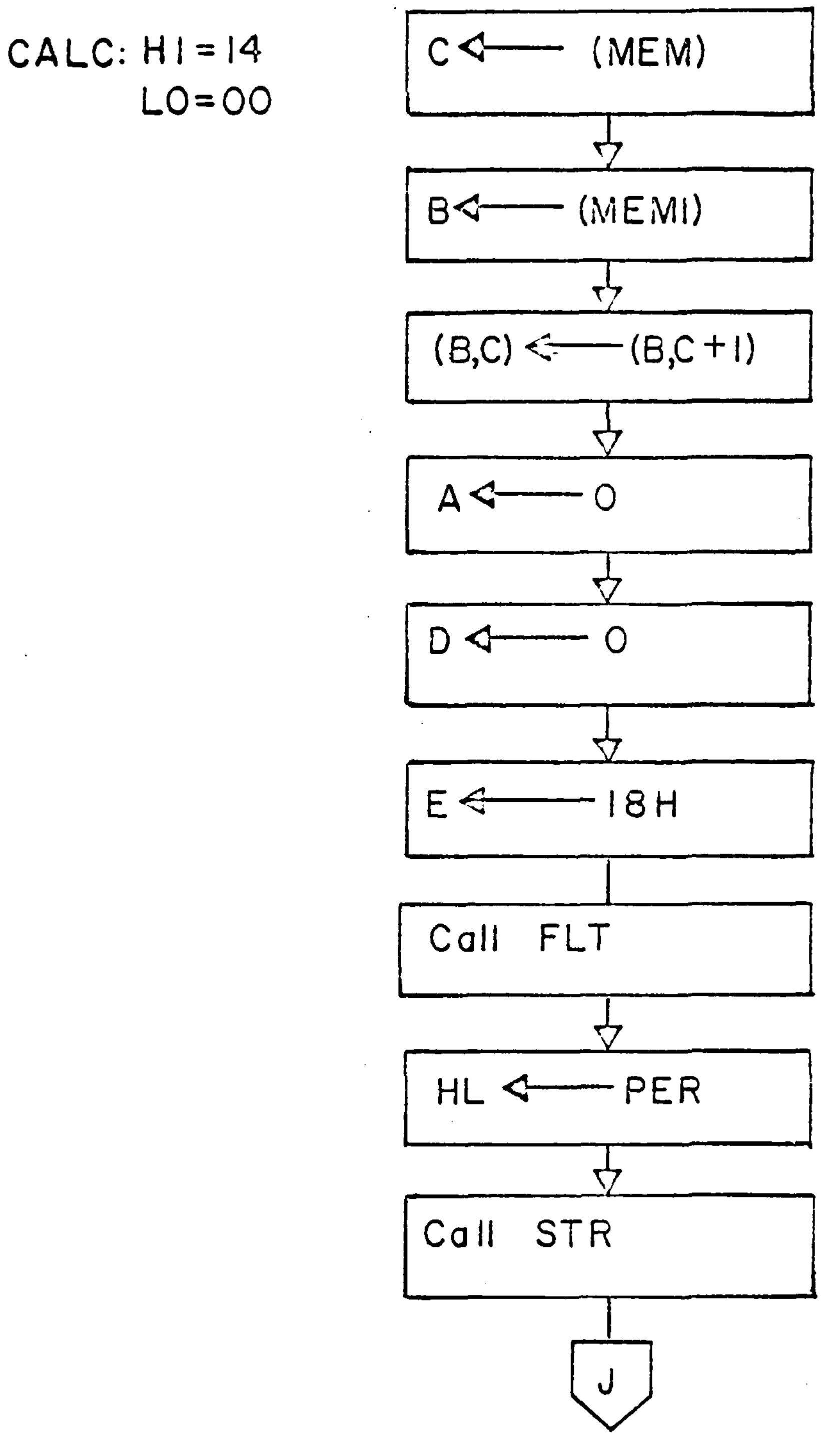


Fig. 6. 10 A CALC Routine

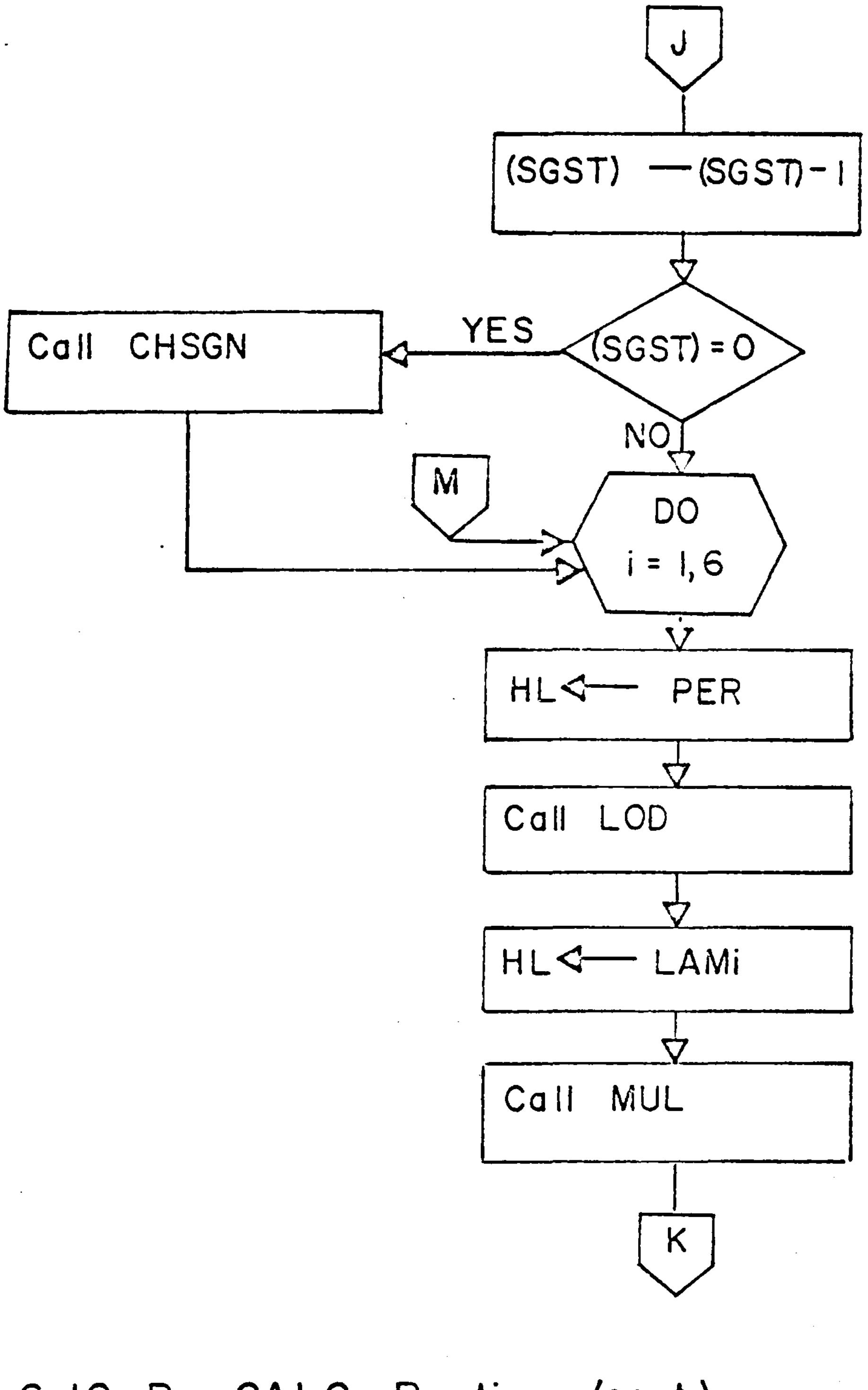


Fig. 6.10 B CALC Routine (cont.)

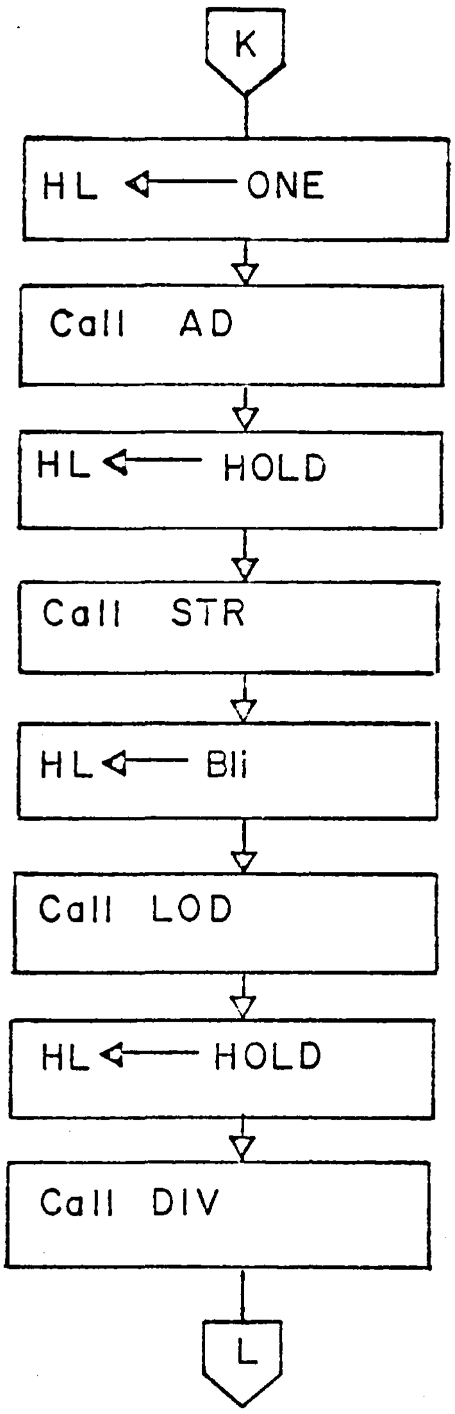


Fig. 6.10 C CALC Routine (cont.)

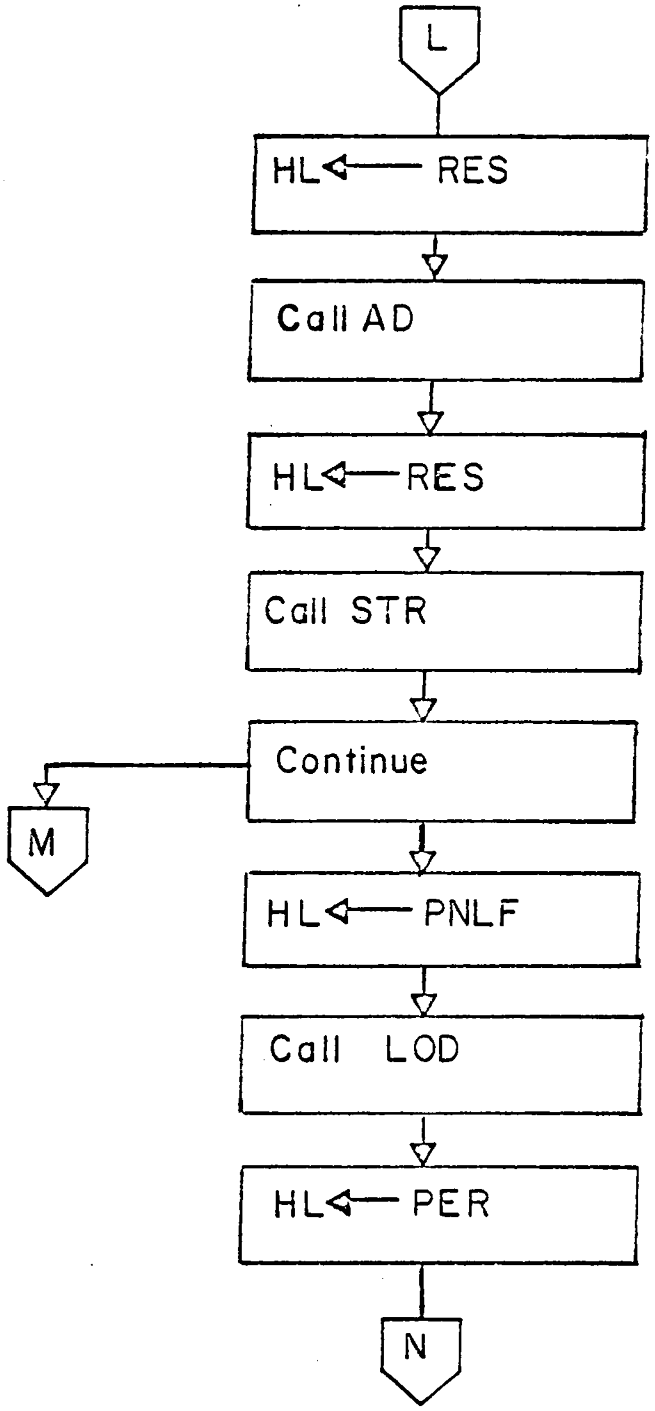


Fig. 6. 10 D CALC Routine (cont.)

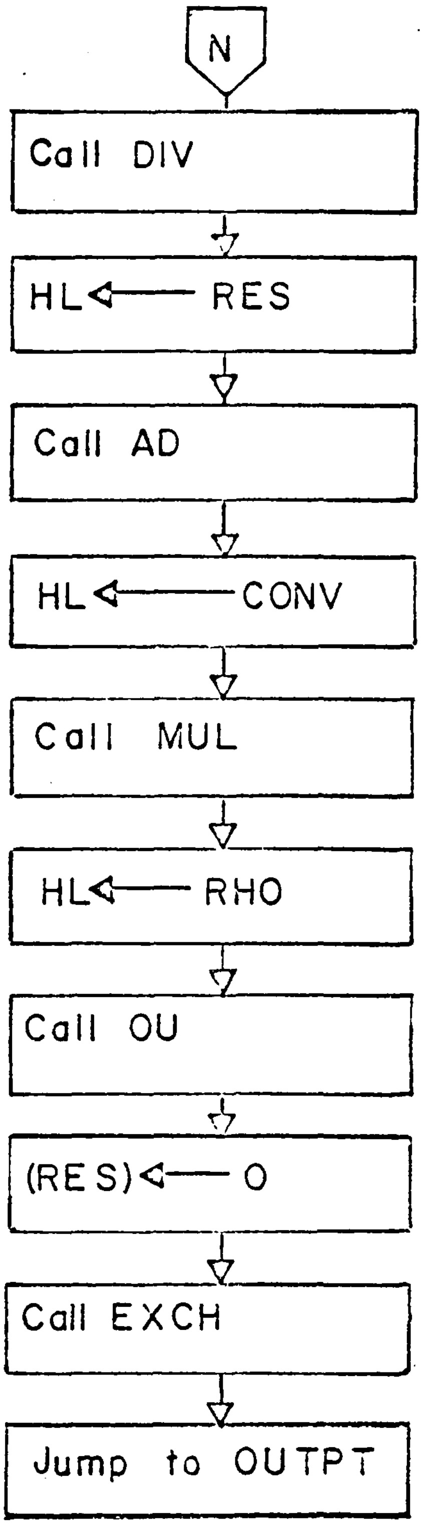


Fig. 6.10E CALC Routine (cont.)

determines the reactor period. MEM1 and MEM are loaded into registers B and C, respectively. Registers A and D are set to zero. Register E, which is the binary-fixed point scaling factor is set to 030_8, (see Appendix B for further information). The subroutine FLT is used to change the number of counts into a binary-floating-point number. The memory location SGST is checked to determine the sign for the reactor period. SGST was previously set to one for a negative number and to zero for a positive number by FLAG. SGST is decremented, and if the result is zero the subroutine CHSGN is called. CHSGN (Fig. 6.11) changes the sign of the reactor period from positive to negative.

After the period is determined, the reactivity is calculated using the inhour equation. LAMi (λ_i, the decay constant) and PER (reactor period) are multiplied together. One is added to this quantity which is divided into BIi (β_i, the delayed neutron fraction). This value is added to RES, which had previously been set to zero before CALC started the calculation. This calculation is performed six times, once for each delayed neutron group. Finally, the neutron generation time is divided by the period and added on. The reactivity is converted from absolute units to pcm by multiplying by $1 * 10^5$. Control is passed to OUTPT.

OUTPT is the routine developed in Chapter V with the following modifications. After RHO + 1 is checked for a decimal point, RHO + 5 is checked instead of RHO + 6. This also eliminates the section in OUTPT labeled OTPT5. The OUT 8 instruction and the Call CHECK instruction just before it, located in the section OTNUM, are omitted.

In the subroutine SIGN, the OUT 9 instructions are changed to OUT 8. Other than these small changes, the routine OUTPT remains the same as discussed in Chapter V. After the data have been transferred to digital displays, RES is initialized to zero and the subroutine EXCH is called. EXCH sets the period counter to zero and stores a new reference in FLX1. Control is passed to WAIT.

The routines WAIT and START perform the same functions as discussed in Chapter V. The only difference is the number of initializations. Instead of seven, there are now only three; RES, MEM, and MEM1. All of these corrections can be found in the program appearing in Appendix F.

CHAPTER VII

RECOMMENDATIONS

Recommendations for Completing the System

Originally the reactimeter calculations were to be performed by a KIM 1 microcomputer instead of the Mark 80. The Mark 80 was used because KIM accessories needed for the reactimeter were unavailable. The hardware circuits presented in Chapters IV and VI will work with most microcomputers with slight modifications. The programs presented in Appendices D, E, and F will only work on 8080 systems.

The following steps are recommendations for completing the reactimeter using the KIM system. Parts have been ordered to complete the system using the KIM 1. When these parts arrive, assembly and testing is required. If a system other than a KIM is used, the following recommendations hold.

I. Study interface techniques of the KIM system
   - A. Data transfer between peripheral devices and KIM
     1. Connect the three 8095 three-state buffers to the data bus (Fig. 4.5)
     2. Connect input pins of the five TIL309 latch displays to the data bus (Fig. 4.9)
     3. Connect the 74154 decoder (Fig. 4.10) to the address bus and the chip enable to the device select pulse clock.

B. Connect the vector interrupt from the DPM (Fig. 6.4)

C. Note corrections required for the software routines INPUT, OUTPT, and WAIT

II. Study the math package that is to be used

A. Math routines that are required

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Absolute Value
6. Change of Sign

B. Extra routines required for a binary-floating-point math package

1. Binary-floating-point to BCD conversion
2. BCD to binary-floating-point conversion
3. Binary-fixed-point to binary-floating-point conversion

C. For a BCD math package, only a binary-fixed-point to BCD conversion routine is needed

D. If these routines are not available, refer to Appendix E and use the Intel 8080 Floating Point Package to aid in the development for the needed routines.

III. Become familiar with software of the KIM system

A. Study the experiments in the lab manual left by the author

B. Practice using the 6502 cross assembler located on the VPI & SU IMP 360 computer

C. Rewrite the software in the KIM system using the flow charts and the 8080 programs as guides; assemble the program on the cross assembler.

It is the recommendation of the author that the use of the KIM system be abandoned and that further development of the reactimeter be based on the use of 8080 A/8085/Z-80 family of chips.

REFERENCES

REFERENCES

1. Gernd Punch and Eckart Schwiegger, "A Digital Reactimeter," Kerntechnik, 17 (1975), 537.

2. John R. Lamarsh, Introduction to Nuclear Reactor Theory, Addison-Wesley, Reading, Mass. (1966).

3. James J. Duderstadt and Louis J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, New York (1976).

4. A. Z. Akcasu, G. S. Lellouche, and L. M. Shotkin, Mathematical Methods in Nuclear Reactor Dynamics, Academic, New York (1971).

5. Jeffery Lewins, "The Use of the Generation Time in Reactor Kinetics," Nuclear Science and Engineering, 7 (1960), 122.

6. VPI & SU Reactor Technical Specifications.

7. David L. Hetrick, Dynamics of Nuclear Reactors, University of Chicago Press, Chicago (1971).

8. J. G. Tyror and R. I. Vaughan, An Introduction to the Neutron Kinetics of Nuclear Power Reactors, Pergamon Press, Oxford, England (1970).

9. Ronald J. Onega, An Introduction to Fission Reactor Theory, University Publications, Blacksburg, Va. (1975).

10. Samuel Glasstone and Alexander Sesonske, Nuclear Reactor Engineering, Van Nostrand Reinhold, New York (1967).

11. David G. Larsen, Peter R. Rony, and Jonathan A. Titus, The Bugbook III, E&L Instruments, Derby, Conn. (1975).

12. Thomas P. Sifferlen and Vartan Vartanian, Digital Electronics with Engineering Applications, Prentice Hall, Englewood Cliffs, N.J. (1970).

13. David G. Larsen and Peter R. Rony, The Bugbook I, E&L Instruments, Derby, Conn. (1974).

APPENDIX A

Calculation of β_{eff}

β_{eff} for the VPI & SU reactor is a slide rule calculation. A known reactivity change, in per cent mills, is measured along with the corresponding period in seconds, and then are aligned as inputs on a Babcock and Wilcox Reactrirule sliderule. β_{eff} is read off a scale. The following reactivity and period measurements, from which β_{eff} was determined, were made by Robert Stone, Reactor Supervisor, on April 4, 1977.

ρ (per cent mills)	T (seconds)	β_{eff}
179	20.4	.00680 ± .00001
170	48.0	.00682
244	11.1	.00676

To determine β_{eff} for the reactor, the average of the three β_{eff}'s is used. The average value for β_{eff} is .00679 ± .00003 which is the value that is used in the algorithm.

APPENDIX B

Intel 8080 Floating Point Package

The following copyrighted material is duplicated here with the permission of Dr. Peter R. Rony, Professor of Chemical Engineering, Virginia Polytechnic Institute and State University.

INTEL 8080 FLOATING POINT PACKAGE

INTRODUCTION

The following pages represent an attempt to describe the Intel 8080 floating point package.

The Intel 8080 floating point package, which was written by O. C. Juelich of Rockwell International Corporation, has its origin in an earlier 8008 floating point package written by C. E. Ohme that consists of a 768-byte ROM program and 63 bytes of scratch pad read/write memory.

The documentation of the 8080 floating point package is poor. To understand what Juelich is doing, you must seek clues in Ohme's much better documentation of the 8008 floating point package, after which the 8080 package is patterned.

The first thing that you must learn is what is meant by a "floating point number." This is discussed below.

FLOATING POINT NUMBER REPRESENTATION

Floating point numbers are represented by four consecutive 8-bit bytes in memory. According to Ohme, they should be in the same bank of memory (you might test whether or not this is true). The interpretation of these four bytes is as follows:

First byte: If this byte is 000_8, the number represented is zero and the remaining bytes are meaningless.

If this byte is non-zero, then it is the floating point exponent (base 2) plus a bias of 200_8. The exponent indicates the power of 2 by which the fraction is multiplied to obtain the represented value. Examples will be given later.

Second byte: Bit 7 indicates the sign of the floating point number. If bit 7 is logic 0, the number is positive; if bit 7 is logic 1, the number is negative.

Bits 6 through 0. These bits, plus an assumed logic 1 in bit 7, are the eight most significant bits of the fraction that is multiplied by 2 to an exponent. The fraction is stored in absolute form

	(unsigned) with the radix point positioned to the left of bit 7. The value of the fraction is thus less than 1.000000 and equal to or greater than 0.500000.
Third byte:	This byte contains the second most significant eight bits of the fraction.
Fourth byte:	This byte contains the eight least significant bits of the fraction.

EXAMPLES OF FLOATING POINT NUMBERS

The significance of the representation does not become clear until you examine some decimal numbers that are represented in floating point notation. Once you get the floating point package in operation on an 8080-based system, you will enjoy converting from decimal numbers to floating point numbers. It takes less than 1 ms to do so on a microcomputer operating at 2 MHz.

a. First byte

It is most appropriate to list the values of 2^n that correspond to the first byte in the floating point representation. Thus, consider the table below:

First byte	n	2^n
000	$-\infty$	$2^{-\infty} = 0.00000000$
170	-8	$2^{-8} = 0.00390625$
171	-7	$2^{-7} = 0.0078125$
172	-6	$2^{-6} = 0.015625$
173	-5	$2^{-5} = 0.03125$
174	-4	$2^{-4} = 0.0625$
175	-3	$2^{-3} = 0.125$
176	-2	$2^{-2} = 0.25$
177	-1	$2^{-1} = 0.5$
200	0	$2^{0} = 1$
201	1	$2^{1} = 2$

202	2	2^2 =	4
203	3	2^3 =	8
204	4	2^4 =	16
205	5	2^5 =	32
206	6	2^6 =	64
207	7	2^7 =	128
210	8	2^8 =	256

The largest possible exponential value of 2^n is $2^{128} = 3.4028 \cdot 10^{38}$.
The smallest possible exponential value of 2^n is $2^{-128} = 2.9386 \cdot 10^{-39}$.
This is sufficient range for almost any type of floating point calculation.

b. Second byte

The first rule involves bit 7, the sign bit for the floating point number. If the second byte is between

000_8 and 177_8

the sign is positive. If the second byte is between

200_8 and 377_8

the sign is negative.

The next thing is to consider the weighting factor for the remaining bits in the second byte. This is shown in the table below:

Bit	Weighting Factor
7	1/2 = 0.5 [always assumed to be present]
6	1/4 = 0.25
5	1/8 = 0.125
4	1/16 = 0.0625
3	1/32 = 0.03125

2	1/64 = 0.015625
1	1/128 = 0.0078125
0	1/256 = 0.00390625

If the indicated bit is logic 1, you add the weighting factor to that for bit 7, which is always assumed to be at logic 1. You proceed bit by bit until you obtain the sum of the fractions for the eight bits in the second byte. To this sum, you add the sum of the fractions for the eight bits in both the third and fourth bytes in the floating point representation.

As a practical matter, we have found it convenient to stop after the second byte and assume the third and fourth bytes do not add much to the floating point number. Whenever we use a hand calculation to check a floating point value, we ignore the third and fourth bytes.

c. Third byte

The procedure is the same as for the second byte. Consider the following table:

Bit	Weighting Factor
7	1/512 = 0.001953125
6	1/1024 = 0.0009765625
5	1/2048 = 0.00048828125
4	1/4096
3	1/8192
2	1/16384
1	1/32768
0	1/65536

d. Fourth byte

More of same. Consider the following table:

Bit	Weighting factor
7	1/131,072
6	1/262,144

5	1/524,288
4	1/1,048,576
3	1/2,097,152
2	1/4,194,304
1	1/8,388,608
0	1/16,777,216

The error inherent in the four-byte floating point representation is one part in 16,777,216, or $5.96 \cdot 10^{-8}$ parts in unity. The precision is quite acceptable for most engineering applications.

e. Examples of floating point numbers

Let us now consider some actual numbers and how they are calculated. The four bytes are listed in sequence in octal code.

200 000 000 000 = $2^0 \times [0.50 + \ldots] = 0.50$

201 000 000 000 = $2^1 \times [0.50 + \ldots] = 1.0$

202 000 000 000 = $2^2 \times [0.50 + \ldots] = 2.0$

203 000 000 000 = $2^3 \times [0.50 + \ldots] = 4.0$

204 000 000 000 = $2^4 \times [0.50 + \ldots] = 8.0$

203 100 000 000 = $2^3 \times [0.50 + 0.25] = 6.0$

203 140 000 000 = $2^3 \times [0.50 + 0.25 + 0.125] = 7.0$

000 XXX XXX XXX = 0.0 X = don't care

201 200 000 000 = -1.0

175 114 314 314 = +0.1

207 310 063 063 = -100.1

177 000 000 000 = +0.250

200 200 000 000 = -0.5

201 300 000 000 = 1.5

202 200 000 000 = -2.0

202 300 000 000 = -3.0

204 040 000 000 = +10.0

150 126 277 255 = +0.00000005

201 174 017 334 = $\pi/2$ = 1.570796

200 061 245 030 = ln 2 = 0.693147

These last several values have been taken from Juelich's program. Our conclusion is that it is easy to represent any number in floating point binary representation. To make the conversion from decimal to floating point binary, use the floating point package rather than trying to perform the calculation by hand. Use hand calculations only for rough values.

FIXED POINT BINARY NUMBER REPRESENTATION

The second format written by C. E. Ohme, and used by O. C. Juelich, is the fixed point binary number representation. The fixed point data format consists of four-byte (32-bit) binary numbers plus an additional byte, called the binary scaling factor, that locates the binary point, "." , relative to the bits representing the value. Two's complement notation is used to represent negative binary values.

The binary scaling factor, the fifth byte, is not normally recorded in the microcomputer. When a format conversion subroutine is called, the binary scaling factor must be specified in the E register. A binary scaling factor of zero indicates that the binary point is immediately to the left of the most significant bit of the 32-bit word,

.00000000 00000000 00000000 00000000

plus the scaling factor of 000_8. A binary scaling factor of 32_{10} = 040_8 indicates that the binary point is immediately to the right of the least significant bit in the 32-bit word,

00000000 00000000 00000000 00000000.

The permissible range of the binary scaling factor is -128_{10} = 200_8 to $+127_{10}$ = 177_8. Note that bit 7 in the scaling factor byte is the sign, a logic 0 representing a positive scaling factor and a logic 1 representing a negative scaling factor.

In general, we will not use this notation very often. Its main value is in the conversion of multi-byte binary numbers to floating point binary numbers. An example of a binary number that would need conversion is a two-byte COUNT register that is monitoring the number of counts from a device. The fixed point binary number representation would be,

```
     00000000 00000000 01001011 01110100.

       These bytes        COUNT
         set to          register
          zero
```

Since the number of counts is an integer that is greater than 0, the binary scaling factor should be 040_8 and the binary point should be located to the right of the least significant bit in COUNT.

EXAMPLES OF FIXED POINT BINARY NUMBERS

In general, you will be working with binary integers. Given below are the fixed point representations of several low-value integers.

```
000 000 000 001 040 = 1.0

000 000 000 002 040 = 2.0

000 000 000 003 040 = 3.0

000 000 000 007 040 = 7.0

000 000 000 002 037 = 1.0

000 000 000 004 036 = 1.0

000 000 000 010 035 = 1.0

000 000 000 020 034 = 1.0

100 000 000 000 002 = 1.0

200 000 000 000 002 = 2.0

300 000 000 000 002 = -1.0

300 000 000 000 001 = -0.5

100 000 000 000 001 = 0.5

100 000 000 000 001 = 0.25
```

Note that the fifth byte in the above examples is the scaling factor. In most cases, the value of this fifth byte will be 040_8.

CHARACTER STRING FORMAT

The character string format for data processed by the floating point package consists of binary representations of decimal characters occupying consecutive bytes of memory. A character string, according to Ohme, may not cross a memory bank boundary (you may wish to test this limitation as well). The characters that may be included in the character string, along with their corresponding octal representations, are listed below:

Character	Octal representation	ASCII representation
0	000	060
1	001	061
2	002	062
3	003	063
4	004	064
5	005	065
6	006	066
7	007	067
8	010	070
9	011	071
+ (plus)	373	053
- (minus)	375	055
. (decimal point)	376	056
(space)	360	040
E (exponential sign)	025	105

Observe that the octal representation can be converted to the corresponding ASCII representation by adding 060_8.

The output format in the floating point package generates character strings in two formats, each consisting of 13 consecutive characters in memory. The format used in a specific case is dependent upon the magnitude of the value represented. For example, zero and magnitudes between 0.1000000 and 9999999. are represented by a space or minus sign, seven decimal characters, and an appropriate positioned decimal point, all followed by four spaces (octal 360). Magnitudes outside this range are represented by a space or minus sign, a value between 1.000000 and 9.999999, an exponential sign (octal 025), and a signed two-digit power of ten.

The input subroutine in the floating point package converts character settings in either of the above formats, or a modified version of them.

The leading sign character may be included or omitted. Up to 37 digits may be used to indicate the value, with or without an included decimal point. If a power-of-ten multiplier is indicated, it may be signed or unsigned and may contain one or two digits. An input character string is terminated by the first character which departs from the specified format above. We use the octal byte, 377_8, to terminate an input character string.

EXAMPLES OF CHARACTER STRINGS

We first give the following examples of input strings and the corresponding output character strings:

Input character string	Output character string
3.141593	3.141593
-.0000000000001	-1.000000E-13
+1.6E5	160000.0
123456789	1.234568E+08
54321E-10	5.432100E-06
-2718281828	-2.718282E+09

It would be advisable to stick to a certain format, such as that illustrated by the number, 5.432100E-06, for all input character strings. We shall do so wherever possible and appropriate.

Subroutines INP and OUT can be used to interconvert string representations with floating point binary representations. Given below are the resulting string representations for input floating point binary quantities.

Floating point binary	Corresponding string representation
160 000 000 000	360 007 376 006 002 011 003 011 005 025 375 000 006
170 000 000 000	360 001 376 011 005 003 001 002 005 025 375 000 003
171 000 000 000	360 003 376 011 000 006 002 005 000 025 375 000 003
174 000 000 000	360 003 376 001 002 005 000 000 000 025 375 000 002
175 000 000 000	360 006 376 002 005 000 000 000 000 025 375 000 002
176 000 000 000	360 376 001 002 005 000 000 000 000 360 360 360 360
177 000 000 000	360 376 002 005 000 000 000 000 000 360 360 360 360
200 000 000 000	360 376 005 000 000 000 000 000 000 360 360 360 360

```
201 000 000 000    360 001 376 000 000 000 000 000 000 360 360 360 360

204 000 000 000    360 010 376 000 000 000 000 000 000 360 360 360 360

210 000 000 000    360 001 002 010 376 000 000 000 000 360 360 360 360

220 000 000 000    360 003 002 007 006 010 376 000 000 360 360 360 360

221 000 000 000    360 006 005 005 003 005 376 011 011 360 360 360 360

222 000 000 000    360 001 003 001 000 007 002 376 000 360 360 360 360

223 000 000 000    360 002 006 002 001 004 004 376 000 360 360 360 360

224 000 000 000    360 005 002 004 002 010 007 376 011 360 360 360 360

225 000 000 001    360 001 000 004 010 005 007 006 376 360 360 360 360
```

Many of the rules governing string character representations are evident in the listing on the preceding page. Note the following:

- o The four spaces, 360 360 360 360, at the end of many string representations.
- o The exponential representation, 025 375, for small numbers.
- o The space, 360, at the beginning of each character string. Thus all of the numbers in the listing are positive.
- o The number of significant digits in the character strings is seven. Digits beyond seven are rounded off.
- o Each character string format contains thirteen bytes.

FLOATING POINT ACCUMULATOR

The floating point accumulator consists of four successive bytes in read/write memory that store an operand and the result of an arithmetic operation. This accumulator must not be confused with the accumulator (A) register in the 8080 chip that contains only eight bits. The floating point accumulator is a 32-bit word in which the four bytes are arranged in the floating point number representation discussed previously:

Byte 1:	Floating point exponent
Byte 2:	Sign bit and eight most significant bits in fraction
Byte 3:	Next eight most significant bits in fraction
Byte 4:	Least significant eight bits in fraction

Each numeric quantity represented by a 32-bit floating point representation has a precision of approximately one part in 16,000,000. Byte 2 may initially appear to contain nine bits; in practice, the first bit in the binary fraction is always assumed to be at logic 1. Thus, bit 7 in Byte 2 can be used as the sign bit.

You can have many different floating point accumulators in your program simply by defining them at the appropriate points using register pair H.

SUMMARY OF ARITHMETIC SOFTWARE IN FLOATING POINT PACKAGE

The arithmetic and data handling software in the floating point package consists of the following (they are listed by their subroutine names):

INIT	Floating point initialize subroutine. Moves a section of code from ROM to scratchpad read/write memory in preparation for the execution of the floating point multiply and divide subroutines. The overflow flag is also set to zero.
STR	Floating point store registers subroutine. Stores the contents of registers A, B, C, and D into four consecutive memory locations (in the same bank of memory) in read/write memory. The address where the first word will be stored is indicated by the contents of the H register pair.
LOD	Floating point load subroutine. Places the specified floating point operand in the floating point accumulator.
AD	Floating point add subroutine. Adds the specified floating point operand to the value in the floating point accumulator and places the sum in the floating point accumulator. The address of the operand is given by the contents of register pair H.
SB	Floating point subtract subroutine. Subtracts the specified floating point operand from the value in the floating point accumulator and places the difference in the floating point accumulator. The address of the operand is given by the contents of register pair H.
MUL	Floating point multiply subroutine. Multiplies the specified floating point operand by the value in

	the floating point accumulator and places the product in the floating point accumulator. The address of the operand is given by the contents of register pair H.
DIV	Floating point divide subroutine. Divides the specified floating point operand into the value in the floating point accumulator and places the quotient in the floating point accumulator. The address of the operand is given by the contents of register pair H.
ABS	Floating point absolute subroutine. Sets the sign of the value in the floating point accumulator to positive.
ZRO	Floating point zero subroutine. Places the value zero in the floating point accumulator.
TST	Floating point test subroutine. Loads the value in the floating point accumulator into registers A through D and sets the zero and sign flag bits to indicate the corresponding attributes of the floating point number.
CHS	Floating point complement subroutine. Changes the arithmetic sign of the value in the floating point accumulator.
INP	Character string input subroutine. Converts the value represented by a character string stored in memory to floating point format and stores the result in the floating point accumulator. The address of the first character in the character string is contained in register pair H.
OU	Character string output subroutine. Converts the value in the floating point accumulator to a character string format consisting of 13 characters, and stores the string in read/write memory. The address of the first character in the character string is contained in register pair H.
FLT	Float subroutine. Converts the fixed point binary data format in the A, B, C, and D registers to floating point format and stores the result in the floating point accumulator. The binary scaling factor (a single byte) needed for the fixed point binary word is contained in register E upon entry to this subroutine.

FIX	Fixed subroutine. Converts the value in the floating point accumulator to fixed point format and returns the result in the A, B, C, and D registers. The binary scaling factor needed for the fixed point binary word is contained in register E upon entry to this subroutine.

The memory addresses, given in both octal and hexadecimal, of the various subroutines described above are as follows:

Subroutine	Hexadecimal address	Octal address
INIT	02 2F	002 057
STR	02 3E	002 076
LOD	02 6E	002 156
AD	02 D7	002 327
SB	02 D4	002 324
MUL	02 8C	002 214
DIV	02 B4	002 264
ABS	02 50	002 120
ZRO	02 46	002 106
TST	02 59	002 131
CHS	02 4D	002 115
INP	05 4A	005 112
OU	06 0C	006 014
FLT	04 FF	004 377
FIX	05 16	005 026

You would call the subroutines at these addresses when needed, provided only that the proper information is available in the internal 8080 registers and in memory.

DRIVER ROUTINES FOR THE ARITHMETIC FLOATING POINT PACKAGE

In the preceding section, we described the various routines available and their starting addresses. Each such routine is called, and executes a return operation at the end. Let us now proceed to the question of how the routines are used in practice. What we must do is take into account the information needed in registers or memory prior to executing a subroutine.

A. To translate a BCD string (located at address STRNG to STRNG+14) into a floating point accumulator word.

```
LXI H,STRNG
CALL INP
LXI H,FLOAT
CALL STR
```

Memory location FLOAT consists of four bytes, which are the floating point accumulator. Register pair H points at the memory location of the first byte.

B. To translate a floating point word into a BCD string (located at address STRNG to STRNG+14).

```
LXI H, FLOAT
CALL LOD
LXI H, STRNG
CALL OU
```

C. To translate a fixed point binary word (located in FIXED to FIXED+3) with binary scale byte at FIXED+4 to floating point representation.

```
LXI H, FIXED
MOV A,M
INR L
MOV B,M
INR L
MOV C,M
INR L
MOV D,M
INR L
MOV E,M
CALL FLT
LXI H, FLOAT
CALL STR
```

Observe that we are moving five successive bytes in memory into registers A through E, then calling subroutine FLT.

D. To translate floating point representation into a fixed point binary word (located at FIXED to FIXED+3) with binary scale byte at FIXED+4.

```
LXI H, FLOAT
CALL LOD
LXI H,FIXED+4
MOV E,M
CALL FIX
LXI H,FIXED
CALL STR
```

E. To copy a floating point accumulator word in location FLOAT to another location FLOATQ

```
LXI H,FLOAT
CALL LOD
LXI H,FLOATQ
CALL STR
```

In other words, we can move the floating point accumulator around in memory.

F. To multiply the quantity in the floating point accumulator by the operand at memory location FLOATQ and store the result in the floating point accumulator.

```
LXI H,FLOAT
CALL LOD
LXI H,FLOATQ
CALL MUL
LXI H,FLOAT
CALL STR
```

Note how simple it is to multiply two numbers. Addition, subtraction, and division are equally simple.

G. To divide the quantity in the floating point accumulator by the operand at memory location FLOATQ and store the quotient in the floating point accumulator.

```
LXI H,FLOAT
CALL LOD
LXI H,FLOATQ
CALL DIV
LXI H,FLOAT
CALL STR
```

Observe that it is not necessary to lose the contents of the floating point accumulator in a multiplication or division (or for that matter, an addition or subtraction) operation. The second instruction from the bottom, LXI H,FLOAT, could easily point to some other memory location.

H. To add the quantity in the floating point accumulator to the operand at memory location FLOATQ and store the sum in the floating point accumulator.

```
LXI H,FLOAT
CALL LOD
LXI H,FLOATQ
CALL AD
LXI H,FLOAT
CALL STR
```

I. To subtract the operand in memory location FLOATQ from the quantity in the floating point accumulator and store the difference in the floating point accumulator.

```
LXI H,FLOAT
CALL LOD
LXI H,FLOATQ
CALL SB
LXI H,FLOAT
CALL STR
```

J. To set the quantity in the floating point accumulator to zero.

```
CALL ZRO
LXI H,FLOAT
CALL STR
```

K. To change the sign of the quantity in the floating point accumulator.

```
LXI H,FLOAT
CALL LOD
CALL CHS
LXI H,FLOAT
CALL STR
```

L. To replace the quantity in the floating point accumulator by its absolute value.

```
LXI H,FLOAT
CALL LOD
CALL ABS
LXI H,FLOAT
CALL STR
```

M. To set the flags for the value of the quantity in the floating point accumulator.

```
                    LXI H,FLOAT
                    CALL LOD
                    CALL TST
```

This ends the list of driver routines available in the basic mathematics package for the 8080 microprocessor. Juelich has written additional software that permits you to perform operations such as sine, cosine, arc sine, arc cosine, exponential, natural logarithm, arc tangent, hyperbolic sine, hyperbolic cosine, and inverse. The form of the driver routines is very much like that given above.

In the above driver routines, observe the interplay between the various subroutines. If we have a 32-bit floating point number in memory, we can load it into the 8080 registers using subroutine LOD. To perform an addition, subtraction, multiplication, or division, we must first identify the memory address, FLOATQ, of the operand. We then call AD, SB, MUL, or DIV to perform the desired mathematical operation. Finally, we identify the memory address, typically FLOAT (but not always so), where we wish to store the result, and then call subroutine STR.

In practice, we have found that the arithmetic floating point package is quite easy to use with the aid of driver routines such as the above. It takes eighteen instruction bytes to perform a single simple arithmetic operation; this total includes both acquiring the information and storing the result.

One point might be of interest: Why were the labels INP, OU, AD, and SB used instead of IN, OUT, ADD, and SUB? The reason is that the latter group of four labels are identical to 8080 instruction mnemonics.

EXAMPLES OF THE USE OF THE SUBROUTINES IN THE FLOATING POINT PACKAGE

What we wish to give here are some numerical examples that demonstrate the operation of some of the above routines.

FLOAT		FLOATQ		FLOAT
202 000 000 000	x	202 000 000 000	=	203 000 000 000
202 100 000 000	x	202 100 000 000	=	204 040 000 000
203 000 000 000	x	202 000 000 000	=	204 000 000 000
204 000 000 000	x	202 000 000 000	=	205 000 000 000
201 000 000 000	+	201 000 000 000	=	202 000 000 000

```
202 000 000 000  +  201 000 000 000  =  202 100 000 000

202 100 000 000  +  201 000 000 000  =  203 000 000 000

203 040 000 000  +  201 000 000 000  =  203 100 000 000

203 100 000 000  -  201 000 000 000  =  203 040 000 000

203 040 000 000  -  201 000 000 000  =  203 000 000 000

203 000 000 000  -  201 000 000 000  =  202 100 000 000

202 100 000 000  -  201 000 000 000  =  202 000 000 000

202 000 000 000  -  201 000 000 000  =  201 000 000 000

201 000 000 000  -  201 000 000 000  =  000 000 000 000

000 000 000 000  -  201 000 000 000  =  201 200 000 000

201 200 000 000  -  201 000 000 000  =  202 200 000 000
```

We shall leave it to you as an exercise to convert the above floating point binary numbers into decimal numbers and to verify the arithmetic operations shown.

We have also tested the use of the FIX and FLT subroutines. A table of results is provided on the following page. Observe that there are many different ways to represent simple integers in the fixed binary format. We prefer the representation in which the binary point is to the right of the least significant bit and the scaling factor is 040_8.

Decimal number	FIXED	FLOAT
1.0	000 000 000 001 040	201 000 000 000
2.0	000 000 000 002 040	202 000 000 000
3.0	000 000 000 003 040	202 100 000 000
7.0	000 000 000 007 040	203 140 000 000
1.0	000 000 000 002 037	201 000 000 000
1.0	000 000 000 010 035	201 000 000 000
1.0	000 000 000 020 034	201 000 000 000

1.0	000 000 000 004 036	201 000 000 000
1.0	100 000 000 000 002	201 000 000 000
-1.0	300 000 000 000 002	201 200 000 000
0.5	100 000 000 000 001	200 000 000 000
-0.5	300 000 000 000 001	200 200 000 000
2.0	177 377 377 377 002	202 000 000 000
0.25	100 000 000 000 000	177 000 000 000

MEMORY MAP FOR FLOATING POINT PACKAGE

The full floating point package, including the elementary functions, requires several kilobytes or ROM and perhaps no more than 24 bytes of read/write scratch pad memory. The following routines have been placed into 1702A EPROM chips:

02 00 to 04 FE	8080 Binary Floating Point System Arithmetic and Utility Package. Programmer, Cal Ohme. December 26, 1973.
04 FF to 06 F4	8080 Binary Floating Point System Format Conversion Package. Programmer, Cal Ohme. December 26, 1973.
06 F5 to 07 06	IDV. Inverse divide routine.
07 07 to 07 40	FSQRT. Square root routine.
07 41 to 07 BB	FMACL. Maclaurin series routine.
07 BC to 08 52	FCOS and FSIN. Sine and cosine routine.
08 53 to 08 D2	ARCTAN. Arc-tangent routine.
08 D3 to 09 28	FCOSH. Hyperbolic cosine routine.
09 29 to 09 E1	FSINH and FEXP. Hyperbolic sine and exponential routine.
09 E2 to 0A 53	FLOG. Natural logarithm routine using Maclaurin series.

SCRATCH PAD MEMORY

A scratch pad memory is a region of read/write memory that is used to store temporary results. In the floating point package, even a short segment of ROM is copied into the scratch pad memory for use in multiplication and division operations.

For the entire arithemtic floating point package, including the elementary function package written by Jeulich, eighty-seven, or hexadecimal 57, scratch pad bytes are required. They can be located at LO memory addresses 00 through 57 (hexadecimal notation) in any HI memory address byte desired. This HI address byte must be entered into the following memory address locations in the Juelich program:

Memory locations that refer to scratch pad memory

hexadecimal	octal
02 35	002 065
02 47	002 107
02 52	002 122
02 5A	002 132
02 80	002 200
02 C6	002 306
02 CB	002 313
02 E1	002 341
03 9D	003 235
04 8F	004 217
04 DE	004 336
04 F5	004 365
05 08	005 010
05 17	005 027
05 6C	005 154
05 E6	005 346
06 3B	006 073
06 A1	006 241
06 C6	006 306
06 DA	006 332
06 E2	006 342
07 44	007 104
07 51	007 121
07 57	007 127
07 5D	007 135
07 79	007 171
07 7D	007 175
07 89	007 211

In the floating point package that we currently have in 1702A EPROM, the scratch pad memory HI byte is 020, in octal code (10 in hexadecimal). The EPROMs cover the address range of from 02 00 to 0A 53. Depending upon the type of application, it may be necessary to change the HI byte to a different value. It is useful to think of memory as being subdivided into 1K memory blocks. The first such blocks are as follows. We define a memory bank as 256 contiguous memory locations that have the same HI address byte.

	Memory Banks	
Memory block	Hexadecimal	Octal
00	00, 01, 02, 03	000, 001, 002, 003
01	04, 05, 06, 07	004, 005, 006, 007
02	08, 09, 0A, 0B	010, 011, 012, 013
03	0C, 0D, 0E, 0F	014, 015, 016, 017
04	10, 11, 12, 13	020, 021, 022, 023
05	14, 15, 16, 17	024, 025, 026, 027
06	18, 19, 1A, 1B	030, 031, 032, 033
07	1C, 1D, 1E, 1F	034, 035, 036, 037
08	20, 21, 22, 23	040, 041, 042, 043
09	24, 25, 26, 27	044, 045, 046, 047
0A	28, 29, 2A, 2B	050, 051, 052, 053
0B	2C, 2D, 2E, 2F	054, 055, 056, 057

This listing should be sufficient for our purposes. It is quite likely that 8708 (2708) EPROMs, which have one kilobyte of memory, will be used to store programs. Thus, it will not be possible to subdivide a memory block into both read-only memory and read/write memory.

We have made an attempt to identify the types of information that are stored in the scratch pad read/write memory in the floating point package. Of particular interest is the fact that from LO = 00 to LO = 2F (hexadecimal code) is stored an actual segment of ROM program that has been copied into scratch pad memory. This is performed using subroutine INIT.

In the listing below, the HI byte is the one which you have chosen for your scratch pad memory. We simply list the LO bytes, the labels, and the significance of the specific memory locations. Memory locations are given in hexadecimal code.

LO memory address	Name	Description
00	MULX4	
01	MULP3	Operand third fraction
05	MULP2	Operand second fraction
09	MULP1	Operand first fraction
0D	DIVX5	
0E	OP4S	Divisor fourth fraction
11	OP3S	Divisor third fraction
15	OP2S	Divisor second fraction
19	OP1S	Divisor first fraction
1C	OP4A	Remainder fourth fraction
1E	DIVX6	
1F	OP3A	Remainder third fraction
23	OP2A	Remainder second fraction
27	OP1A	Remainder first fraction
2A	OP4X	Remainder fourth fraction
2E	OVER	
2F	PREX	Previous exponent
30	ACCE	Accumulator exponent
31	ACCS	Accumulator sign
32	ACC1	Accumulator first fraction
33	ACC2	Accumulator second fraction
34	ACC3	Accumulator third fraction
35	SF	Subtraction flag
36	ADPL	Character string word
37	ADRH	Character string bank
38	TMP1	Temporary storage
39	TMP2	Temporary storage
3A	TMP3	Temporary storage
3B	VALE	Value exponent
3C	VAL1	Value first fraction
3D	VAL2	Value second fraction
3E	VAL3	Value third fraction
3F	TMP4	Temporary storage
40	FSQRN	
40	FMACX	
44	FSQRX	
44	FMACS	
48	FMACT	
4C	FMACG	
4E	FCSHD	
4E	FLOGE	
4E	FSNHD	

4F	FEXOV
50	FATNT
50	FSINX
50	FSNHX
50	FLOGX
54	FATNU

Stored in the above LO address locations are the indicated quantities, which vary during the arithmetic calculations. This is why these quantities need to be located in scratch pad memory.

APPENDIX C

Instrument Documentation

Westinghouse | 6377

May 1, 1960

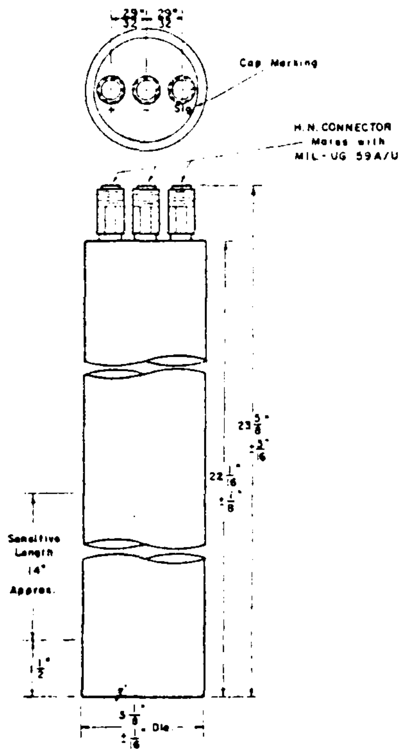
COMPENSATED IONIZATION CHAMBER TYPE 6377

The 6377 compensated ionization chamber is designed to detect thermal neutrons in the range from 2.5×10^2 to 2.5×10^{10} neutrons cm^2 second, in the presence of very high gamma radiation fields. The detector is extremely rugged in construction, meeting MIL-S-901 for shock and MIL-Std-167 (type 1) for vibration, and may be operated in any position at temperatures up to 175°F. The 6377, including the connectors, is constructed of magnesium alloy, with high stability, crosslinked polystyrene insulation. The use of this latter material assures completely noise free performance of the detector, even in the lowest decade of operation.

The 6377 incorporates two outstanding features. The first is the use of guard ring construction throughout to minimize the reduction in signal currents due to electrical leakage of the insulators. The second is the provision for continuously variable, electrical compensation. This feature provides any desired degree of reduction of the signal caused by gamma radiation, including complete cancellation.

The thermal neutron sensitivity of the 6377 is approximately 4×10^{-14} amperes neutron cm^2 second. The gamma sensitivity, when operated uncompensated, is approximately 3×10^{-11} amperes R hour.

The 6377 is similar to the 7353, differing only in outline dimensions.


Cap Marking
H.N. CONNECTOR Mates with MIL-UG 59A/U
Sensitive Length 14" Approx.
23 5/8"
±3/16"
22 1/16"
±1/8"
1 1/2"
5 1/8" Dia.
±1/16

MECHANICAL:

Maximum Diameter	3-3/16	Inches
Maximum Overall Length	23-13/16	Inches
Approximate Sensitive Length	14	Inches
Net Weight	5-3/8	Pounds
Shipping Weight	19	Pounds

MATERIALS:

Outer Case	3% Al, 97% Mg Alloy
Electrodes	3% Al, 97% Mg Alloy
Insulation	Stabilized Polystyrene
Neutron Sensitive Material:	
Content	Boron enriched in B-10
Thickness	1 mg/cm^2
Gas Filling	Nitrogen

IMPEDANCE:

Resistance: (Note 2)		
Signal Electrode to Case (Minimum)	10^{14}	Ohms
H.V. Electrode to Case (Minimum)	10^{12}	Ohms
Compensating Electrode to Case (Minimum)	10^{12}	Ohms
Capacitance: (Note 1)		
Signal Electrode to Case (Approx.)	275	µµf
H.V. Electrode to Case (Approx.)	315	µµf
Compensating Electrode to Case (Approx.)	125	µµf

MAXIMUM RATINGS:

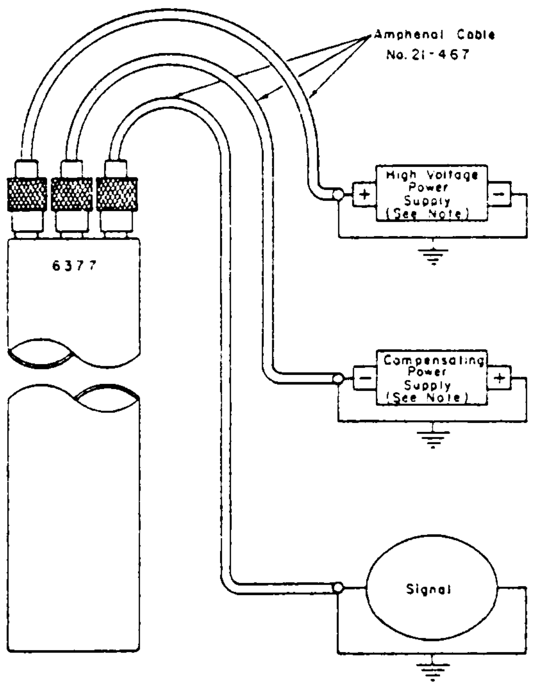
Voltage Between Electrodes (dc)	1500 max.	Volts
Temperature	175 max.	Degrees F
External Pressure (Note 3)	180 max.	Pounds/Inch^2
Thermal Neutron Flux	5×10^{11} max.	nv

TYPICAL OPERATION:

Typical Connection	See Figure 1	
Operating Voltage	300 to 800	Volts
Compensating Voltage (See Figure 3)	−10 to −80	Volts
Saturation Characteristics	See Figure 2	
Thermal Neutron Flux Range	2.5×10^{2} to 2.5×10^{10}	nv
Thermal Neutron Sensitivity	4×10^{-14}	Amperes/nv
Gamma Sensitivity:		
Total Compensation	zero	
Uncompensated	3×10^{-11}	Amperes/R/hour

1. Capacitance is measured between an electrode and case, with all other electrodes grounded.
2. The detector may not be immersed directly in water, and high humidity environments should be avoided as they will impair performance.
3. The pressurizing atmosphere must be dry and non-corrosive.

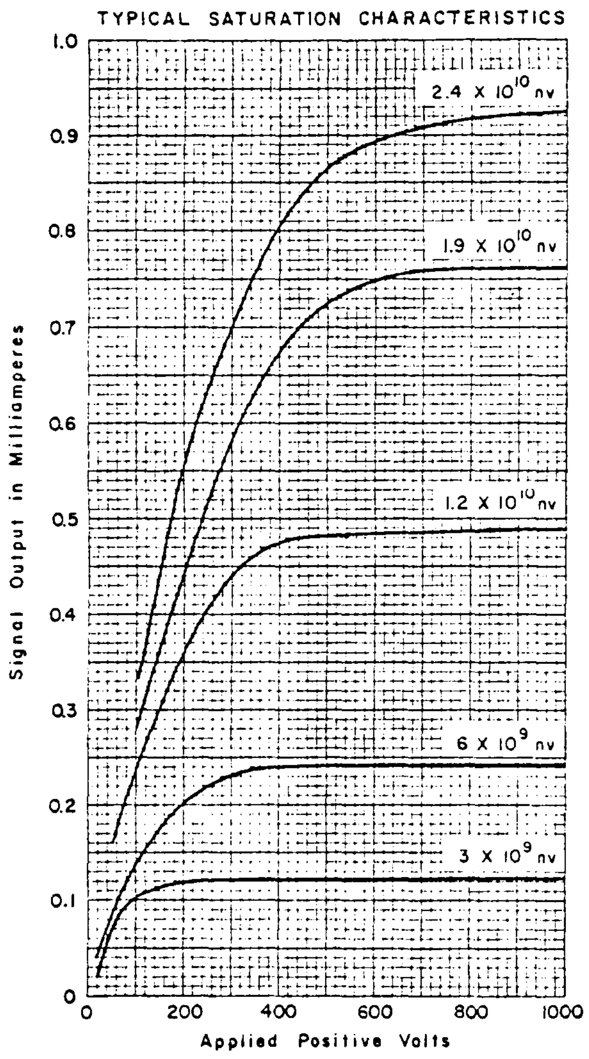
TYPICAL CONNECTION DIAGRAM


Amphenol Cable No. 21-467
6377
High Voltage Power Supply (See Note)
+
−
Compensating Power Supply (See Note)
−
+
Signal

Note: Permissible power supply regulation and ripple will depend upon the particular application. See Section entitled "Ionization Chamber Operation."

CE-A1324 R1

FIGURE 1



CE-A1284 R2

FIGURE 2

TYPICAL COMPENSATION CHARACTERISTICS

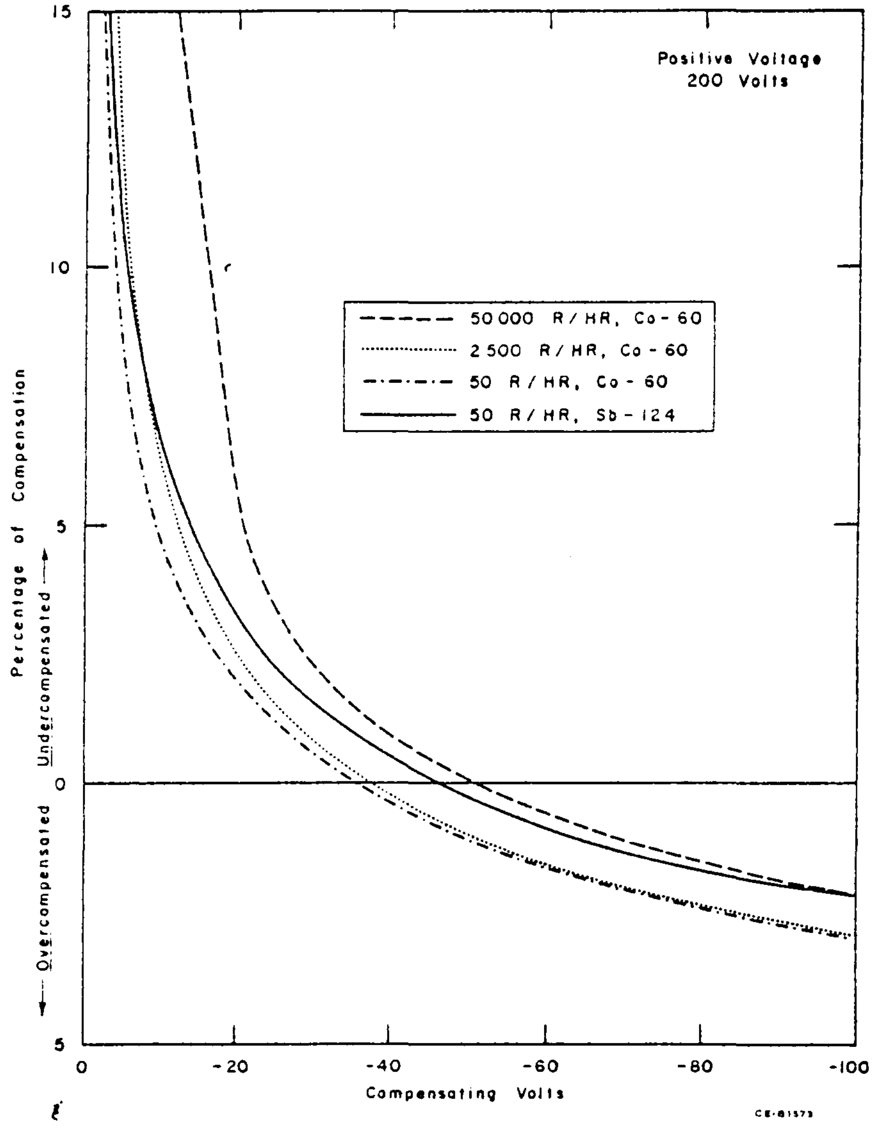


FIGURE 3

CONTENTS

KEITHLEY INSTRUMENTS CLEVELAND, OHIO

SECTION I INTRODUCTION

Model 411

The Keithley Model 411 Micro-microammeter is a line operated vacuum tube electrometer designed and constructed especially for measuring small currents. Full scale ranges are from 10^{-3} to 10^{-11} ampere.

The features include full-scale voltage drop at the input of less than five millivolts, zero drift of less than 2% of full scale per week, good accuracy and calibration stability, and simplicity of operation. It also has an output which will drive a 0-1 or 0-5 milliampere recorder as well as the numerous potentiometer - rebalance recorders; one output terminal is at ground, making it convenient to connect cathode ray oscilloscopes or pen-driving amplifiers, similar to the Brush and Sanborn equipment.

The major panel controls are the range switch (amperes full-scale) and the zero. Minor controls are the Zero Check, used to short circuit the input and in setting the zero, Meter Polarity for providing up-scale readings for currents flowing in either direction, and an ON-OFF power switch. The meter dial is illuminated, and these bulbs serve as the pilot light.

Model 411C

The Keithley Model 411C is identical to the Model 411, except that the panel meter is provided with contacts which can be set to close at any predetermined meter pointer deflection. The delicate contacts of the meter operate a relay in the 411C, and the relay contacts (SPDT) are available for external switching functions through an AN connector on the rear of the chassis.

Response Speed, both models

The 411 and 411C are shipped with capacitors shunting the range resistors on the 10^{-8} through 10^{-11} ampere ranges. The capacitors damp the response, limiting the amplification of spurious disturbances, and preventing overshoot and ringing when a square pulse of current is applied and input cable capacitance is as much as 5000 micro-microfarads. Such damping is usually preferred when long input cables are used, as with remote ion chambers. When maximum speed is desired, as in some production tests, and very short input cables are being used, the capacitors may simply be removed from the range switch. See details on page II-2.

SECTION II DESCRIPTION

Seventeen overlapping current ranges, from 10×10^{-4} ampere to 10×10^{-12} ampere are selected by the Amperes Full Scale switch, located left of the Meter. The accuracy of the ranges from 10×10^{-4} through 3×10^{-7} is within 2%, 10×10^{-9} through 10×10^{-12} is within 4%.

Input Impedance is controlled by negative feedback from the output so that the voltage drop across the input terminals is less than 5 millivolts for full-scale meter deflection.

The Input Connector is located on the back face of the chassis. It is a UHF connector with teflon insulation, and accepts a standard teflon insulated mating plug. The plug and lead wires or cable should be extremely well insulated to prevent the leakage of the small currents. A cap is provided for keeping dirt out when the instrument is stored.

Input Switch Labelled ZERO CHECK is located to the left of the range switch. When depressed, it effectively shorts the input to remove spurious charges, and provides the zero input current reference for zeroing the meter with the Zero Control.

Grid Current is less than 5×10^{-14} ampere, and represents the limit of measurement of a vacuum tube electrometer. This is about 0.5% of full-scale on the most sensitive range.

Zero Drift is less than 2% of full scale per week on all ranges. This includes warmup from a cold start, provided the source voltage is 10 volts or more.

Zero Control - The Zero knob is located to the right of the meter and is used for zeroing the meter with zero input current. Effectively zero input current can be obtained by depressing the Zero Check button. The input must not be short-circuited. This upsets the negative feedback path and makes it impossible to zero the meter.

It is recommended that the meter pointer not be set anywhere but zero on the meter scale with zero input current, because with the feedback used, a dc potential is developed across the input whenever the output and the panel meter are not zero for zero input current. Recorders, of course, can be biased to any part of their scale for zero volts at the Model 411 output.

Output is provided for driving recorders. The amplifier will develop 10 volts for full-scale meter deflection, and 5 milliamperes can be drawn without upsetting the circuits. The OUTPUT connector is at the rear of the chassis. The connection details and suitable output attenuators are discussed in OPERATION, Section III.

Response Speed of the 411 depends upon the current range being used and also upon the capacitance of the external circuitry. On the less sensitive ranges the speed is limited by the amplifier response, which is from dc to approximately 1,000 cps. On the ranges from 3 x 10^{-7} to 10 x 10^{-11} amperes the speed has been reduced to about 1.0 second by the addition of capacitors across the range resistors. On the three most sensitive ranges, shunt capacitance across the input limits the response speed. Because of the method of application of the negative feedback, the slowing effects of capacitance from the high input terminal to ground have been greatly reduced, but are still significant. Table I below gives typical response speeds; viz; the time constant of the response to a step function.

TABLE I

TYPICAL RESPONSE SPEEDS

(to reach 67% of final value)

Ranges	with no significant external capacitance	with 5000 mmf across the input
1 x 10^{-11}	2.0 Seconds	4.0 Seconds
3 x 10^{-11}	1.0	2.0
1 x 10^{-10}	0.5	1.0
3 x 10^{-10}	0.5	1.0

If the maximum speed of response is desired, the capacitors shunting the range resistors may be removed; however the increased response to spurious ac signals may interfere with recording, and the transient response may suffer.

Amplifier Noise is principally power frequency, and is 50 millivolts rms max at the output terminals, irrespective of the current range. From the most general point of view, grid current and amplifier zero drift are also background noise; these have already been discussed.

Circuit Description

The circuit diagram DR 11175-C is enclosed at the back. The amplifier consists of two 5886 electrometer tubes operated as a balanced stage, with a substantial amount of in-phase rejection. Further in-phase rejection is obtained by supplying V1 and V2 screens from V3 and V4. A triode connected 6CM6 is used as the cathode follower output stage.

Negative feedback from the output is accomplished through the shunt resistor to the grid of the input electrometer tube. It is this feedback which keeps the input voltage drop low.

The open loop voltage gain of the amplifier, measured from the first stage grid to the feedback connection which would normally be connected to the low impedance end of the shunt resistor, is about 2500. This assures a low input drop.

To insure low drift, the feedback-voltage (the voltage drop across the high resistance range resistors) is made ten volts on all ranges.

The power supply is regulated by a Sola transformer. Half-wave selenium rectifiers supply the B+ and B- potentials. The filtering is conventional.

Calibration is determined by the value of the high resistance range resistors. From 10^{-3} to 10^{-7} amperes, the overall accuracy is better than 2%. From 3×10^{-8} to 10^{-11} amperes, the accuracy is better than 4%.

The meter is connected between the output terminal and ground. When the range resistor is shorted in zeroing the instrument, the meter measures the voltage existing between the input terminal and the output terminal (which are connected together when the shorting button is pressed) and ground.

The balancing of the amplifier, with the Zero control, is done in the filament circuit of V2. This is a convenient low-impedance point and does not disturb the electrode potentials of the low grid current electrometer tube.

SECTION III OPERATION

Simplicity of operation is an outstanding characteristic of the Model 411. First connect the input to a current source, and the output to a recorder or external indicator, if desired.

Then: a) Plug the power cord into a 110 volt 60 cps outlet. Note that because a Sola resonant regulating transformer is used, the power frequency, as well as voltage, must be the proper value.

b) Turn the amperes Full Scale to the 10 x 10^{-4} position.

c) Turn the power switch to ON.

d) After a few minutes warmup, set the panel meter to zero with the ZERO control.

e) Advance the instrument's sensitivity with the range switch, until a usable deflection is obtained on the panel meter. The current is read directly. Attention should be paid to the METER polarity switch, so that an up-scale deflection is obtained.

f) Periodically check the zero setting by operating the ZERO CHECK switch and rezeroing the meter if necessary.

Input, using cabling

The current source should be connected to the input connector with the high impedance side of the current source associated with central conductor of the connector. The lead-in cable should be polyethylene, polystyrene, or teflon insulated coaxial cable, and the connector should have teflon insulation. Amphenol type 83-756 or equivalent is recommended. During preparation of cable and connectors, it is essential that all high impedance surfaces be kept scrupulously clean to avoid leakage. With graphite coated cable, it is necessary to avoid tracking graphite onto the high impedance surfaces of the cut end of the insulation and the teflon surface of the connector. Movement of the cable during measurement should be avoided since this will cause spurious needle movements, because of capacitance changes and generation of static charges.

RECORDING: The Model 411 is provided with a connector on the rear of the chassis for recording. The output for full-scale meter deflection is $\pm$10 volts. The maximum current that may be drawn from the output terminals is 5 milliamperes. This output is suitable for driving one and five milliampere recorders as well as recorders employing an amplifier. Cinch-Jones S-202-B is the chassis connector, P-202-CCT is the mating plug. Terminal #1 is ground.

Table III gives resistance to be used in series with one and five milliampere recording milliammeters, to make the recorder full-scale deflection equal the panel meter full-scale deflection.

TABLE III

Recorder	Series Resistance
1 m.a.	8.3 to 8.7K
5 m.a.	1920 to 1940

The exact series resistance varies from recorder to recorder, and a portion of the series resistance should be adjustable so that the recorder may be calibrated exactly against the panel meter.

A suitable voltage divider for more sensitive recorders can easily be made, keeping in mind that 10 volts appear at the output terminals for full-scale deflection of the panel meter, and that a 2000 ohm divider will not draw too much output current and will be sufficiently low impedance to connect to amplifier inputs.

<u>The Speed of Response</u>, or the time constant of an input transducer and micro-microammeter, depends upon the speed of response of the circuitry of the instrument and also upon the capacitance of the current source and its connecting cable. Because of the way the negative feedback is applied in the Model 411, the external input capacitance is not nearly as important as in systems using a voltmeter across a shunting resistor, and quite large capacitances can be tolerated without having an impossibly slow response. Thus, a cable run from an ion chamber to the micro-microammeter is permissible.

The internal time constant of the Model 411 depends upon both the frequency response of the amplifier stages and the time constants of the high megohm range resistors and the associated distributed capacitances. These change from range to range on the 411, the speed decreasing as the sensitivity is increased. Table I in Section II, Description, gives quantitative values.

<u>+216 Volts</u>. A connector has been mounted on the back face of the chassis to provide +216 volts for polarizing an ion chamber. The potential is derived from 2 OB2 voltage regulator tubes and is well filtered. The supply can be short circuited without damaging it. The chassis connector is Cinch-Jones S101, and P101 is the mating plug.

SECTION IV MAINTENANCE

The Keithley Model 411 Micro-microammeter has been designed to give long, trouble-free service. High quality components have been used throughout, and the circuits are stabilized by a substantial amount of negative feedback.

DR 11175-C, at the back, is the detailed circuit schematic diagram of the Model 411. The circuit operation was discussed in Section II, Description.

Maintenance Adjustments

One maintenance control is provided. It is accessible from the top of the chassis, and is located behind the meter.

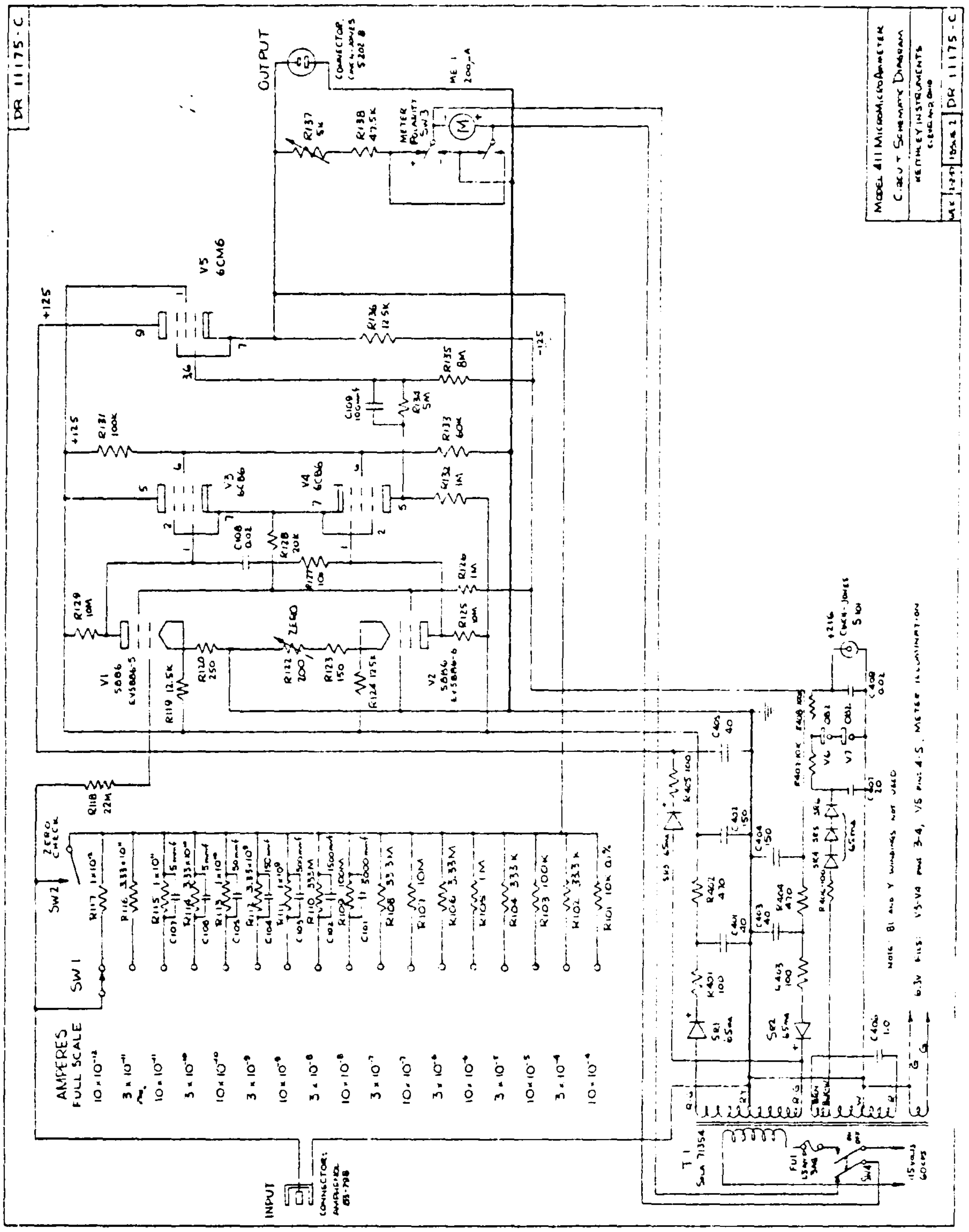
R138, METER CALIBRATION, is in series with the Meter. To recalibrate, use the 10 x 10^{-4} range and, with 7 x 10^{-4} ampere through the input circuit, adjust R138 so the meter reads exactly 7.0. Since the shunt resistor on this range is accurate to 0.1% of its nominal value the overall accuracy can be adjusted to about 1% of full scale. On the 3 x 10^{-4} to 10 x 10^{-8} ampere range the range resistors are accurate to 1% and, providing the calibration was accurately done on the 10 x 10^{-4} range, the overall accuracy will be 2%. From 3 x 10^{-8} to 10 x 10^{-12} amperes the range resistors are accurate to 3% and the overall accuracy will be 4%.

Vacuum Tubes V1 and V2 are the two electrometer tubes, and are located in an aluminum can which plugs onto the top of the chassis near the input terminals. The tubes have been selected, matched and labelled; V1 is Keithley part EV5886-5 and V2 is EV5886-6. The difference between the two is that EV5886-6 does not have to have low grid current. It is recommended that the complete Input Tube assembly, Model 4102 be kept for replacement purposes.

The other tubes are standard receiving tubes and need no special selection to assure satisfactory performance of the Model 411.

INSULATION: All insulation for the high impedance conductors is made of teflon, as are the contact insulators on the range switch. This should give satisfactory service in all humidities. Occasionally, the high impedance insulators should be inspected to insure that they are free from dirt and dust.

CONNECTOR CAP: The cap for the input connector should be kept in place whenever the connector is not being used. In storage and in transport, it keeps the insulation from accumulating dust and dirt. Before screwing the cap back onto the connector, be certain that it is clean, so the insulation will not be contaminated.



DR 11175-C
Model 411 MicroMicroAmmeter
Circuit Schematic Diagram
Keithley Instruments
Cleveland Ohio
OUTPUT
INPUT
AMPERES FULL SCALE
SW1
SW2
ZERO CHECK
ZERO
METER POLARITY SW3
V5 6CM6
V3 6CB6
V4 6CB6
V1 5886
V2 5886
+125
-125
+216
T1
115 volts 60cps
Note B1 and Y windings not used
6.3v fils: V3-V4 pins 3-4, V5 pins 4-5, meter illumination

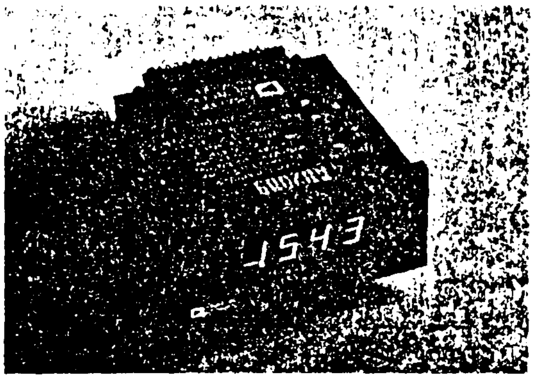
ANALOG DEVICES

3½ Digit AC Line Powered DPM

AD2009

'EATURES
\C Line Powered
3right, Seven Segment Gas Discharge Display
3CD Data Outputs Standard
Hold and Trigger Control Signals
=ull Scale Ranges of ±1.999V or ±199.9mV
)isplay Blanking Control
ndustry Standard Panel Cutout

\PPLICATIONS
3eneral Purpose DPM Applications Requiring AC Power and a High Visibility Display
)ata Logging and Digital Feedback Control Systems



:NERAL DESCRIPTION
e AD2009 is a low cost 3½ digit, AC line powered DPM de-
ned for general purpose DPM applications The AD2009
easures bipolar input voltages over full scale ranges of either
.999V or ±199 9mV, with an accuracy of ±0.1% reading ±1
git and displays the readings on large, bright 0.55" (14mm)
ckman gas discharge displays.

ARGE, BRIGHT DISPLAY
r display only applications, the Beckman display offers ex-
llent appearance and visibility. The AD2009 display is easily
ad up to 50 feet (15m) away and over all ambient lighting
nditions. The non-glare lens allows a choice of either red or
aber display colors, and is easily silk-screened with company
go or measurement units. External control of decimal points
d display blanking is provided.

MPLE DATA INTERFACING
nce the AD2009 is designed around TTL logic circuits, paral-
BCD data, TTL/DTL compatible, is a standard feature,
owing easy interfacing to a variety of data peripherals, such
digital comparators and line printers. Under internal con-
i, the AD2009 converts at a nominal rate of six conversions
second. Using the Hold and Trigger controls, up to 100
nversions per second can be externally triggered.

DUSTRY STANDARD CASE DESIGN
response to industry's urgent need for DPM standardization,
alog Devices has adopted the most popular AC powered DPM
nel cutout size for the AD2009 and all future AC line pow-
ed DPM's. Since this 3.924" x 1.682" (99.67 x 42.72mm)
nel cutout is used by so many AC powered panel meters, the
tential DPM customers can be assured that second-sources will be available and future new products will be usable without mechanical changes to their instruments or systems.

DESIGNED AND BUILT FOR RELIABILITY
Design and manufacturing techniques are chosen to insure reliability in the AD2009. Conservative design techniques and thorough component evaluation are only the beginning. Manufacturing processes are monitored by continuous quality assurance inspections to insure proper workmanship and testing. Like every other Analog Devices' DPM, each AD2009 is fully tested for electrical specifications, calibrated, and given one full week of failure free burn-in before shipment.

THEORY OF OPERATION
The AD2009 uses a dual slope conversion technique with an absolute value voltage to current converter input. The entire conversion cycle takes less than 10 milliseconds, allowing a complete conversion to be done during the negative half cycle of the AC line, and the resulting reading is displayed during the positive half cycle of the AC line. This scheme not only insures a flicker free display, but also allows externally triggered conversions at rates up to 100/second for data interfacing applications. In order to insure a bright display even during operation at low line voltages and to help insure the reliability of the Beckman displays, a separate power supply is provided to continually illuminate two "keep-alives" in the Beckman display.

SPECIFICATIONS (typical @ +25°C and nominal line voltage)

DISPLAY OUTPUT
- Beckman Seven Segment Gas Discharge Display, 0.55" High (14mm) for Three Data Digits, 100% Overrange and Negative Polarity Indication. Overload indicated by blanking the three data digits and displaying the "1" overrange. The polarity remains valid.
- Decimal Points Selectable at Input.
- Display Blanking

ANALOG INPUT
- Configuration: Bipolar, Single Ended
- Full Scale Range: ±1.999V or ±199.9mV (see S option)
- Automatic Polarity
- Input Impedance: 100MΩDC
- Bias Current, Both Ranges. 3nA @ 2V FS, 20nA @ 200mV FS
- Overvoltage Protection, Both Ranges: 200VDC Sustained

ACCURACY
- ±0.1% ±1 Digit[1]
- Resolution: 1mV or 100μV (S option)
- Temperature Range[2]: 0 to +50°C Operating
  -25°C to +85°C Storage
- Temperature Coefficient:

Gain (both ranges)	– ±60ppm/°C
Zero Offset (2V Input)	– ±30μV/°C
(200mV Input)	– ±10μV/°C

- Warm-Up Time to Rated Accuracy 15 minutes
- Settling Time to Rated Accuracy: 0.3 sec

NORMAL MODE REJECTION
- 18dB @ 60Hz

COMMON MODE REJECTION (1kΩ source imbalance @ 50-60Hz, with standard shielded transformer)
- 2V Input – 100dB
- 200mV Input – 80dB

COMMON MODE VOLTAGE
- ±300VDC (600VAC p/p) (floated on power supply transformer when BCD outputs and control signals are not used)

CONVERSION TIME
- 10msec

CONVERSION RATE
- Internal Trigger: 6 conversions per second
- External Trigger: 0-100 conversions per second

DIGITAL CONTROL SIGNALS
- DTL/TTL Compatible

	In	Out
Logic "0"	<0.8V	<0.4V
	>2.0V	>2.4V

CONTROL INPUTS[3]
- Display Blank (1TTL Load). Logic "0" or grounding blanks the entire display, not including the decimal points. Logic "1" or open circuit for normal operation. Display blanking has no effect on output data and the display reading is valid immediately upon removal of a blanking signal.
- Hold (1TTL Load). Logic "0" or grounding disables either the external or internal trigger and the last conversion is held and displayed.
- External Trigger (1TTL Load). Positive pulse (500μsec max width) will initiate conversion.
- Decimal Points (Not TTL Compatible). Grounding will illuminate the desired decimal point. External drive circuitry must be capable of withstanding 100V when the decimal points are turned off.

DATA OUTPUTS[3]
- 3BCD Digits (Drives 6TTL Loads). Positive true, unlatched
- Overrange (Drives 6TTL Loads). Unlatched, Logic "0" indicates overrange (≥1000).
- Overload (Drives 6TTL Loads). Unlatched, Logic "0" indicates overload (≥2000).
- Polarity (Drives 6TTL Loads). Latched, Logic "1" indicates positive polarity.
- Status (Drives 10TTL Loads). All digital outputs are valid when status is at Logic "0". Logic "1" indicates conversion is in progress.
- Internal Trigger Output (Not TTL Compatible). When connected to External Trigger Input will cause the AD2009 to convert at 6 conversions per second. This output can only be used for triggering the AD2009.

POWER INPUT
- AC line, 50–60Hz, 4.2 Watts at 60Hz; 4.7 Watts at 50Hz (at nominal line voltages).

CALIBRATION ADJUSTMENTS
- Gain
- Zero
- Recommended recalibration interval – 6 months

SIZE
- 4.18"W x 1.93"H x 4.15"L (106 x 49 x 112mm)
- 4.77"L (121mm) to rear of card edge connector
- Panel cutout required: 1.682 x 3.924" (42.72 x 99.67mm)

WEIGHT
- 15 ounces (425 grams)

OPTIONS[4] – ORDERING GUIDE
- AC Power Inputs (50–60Hz)

AD2009	117VAC	±10%
AD2009/E	– 220VAC	
AD2009/F	– 100VAC	
AD2009/H	– 240VAC	

-

AD2009	– 1.999VDC Full Scale
AD2009/S	– 199.9mVDC Full Scale

-

Lens 7	– Red with ADI Logo
Lens 8	– Red without ADI Logo
Lens 13	– Amber with ADI Logo
Lens 14	– Amber without ADI Logo

CONNECTOR
- 30 Pin, 0.156" Spacing Card Edge Connector, Amphenol 225-215 24-601 (117) or Equivalent
- Optional: Order AC2611 @ $4.50

PRICING
- $140 (unit quantity)
- Consult Factory for OEM quantity pricing

[1] Guaranteed @ +25°C.
[2] Guaranteed.
[3] Not to be used when the AD2009 is floating on common mode voltages.
[4] Only one input range and AC power input may be specified.
[5] Lens 7 is supplied if no lens option is specified.
Specifications subject to change without notice.

Applying the AD2009

INTERFACING THE AD2009

Input Connections

The AD2009 has a single ended input with common analog and digital grounds. When digital control lines and BCD data outputs are not used, the entire DPM can be floated on the power supply transformer at up to 300VDC common mode voltages. If these signals are used, care should be taken to insure against ground loops within the system causing erratic and/or erroneous readings.

Decimal Points

Grounding the proper pin will illuminate the desired decimal point. If external logic drives are used to control the decimal points, drive circuitry must be able to withstand 100V when the decimal points are turned off

Display Blanking

The entire display (excluding decimal points) may be blanked by applying logic "0" or grounding the proper control input (pin 13). Blanking the display has no effect on the output data or the conversion process. The data remains valid during blanking and the DPM reading is correct immediately upon removal of the blanking signal.

Interfacing Digital Data Outputs

The digital data outputs of the AD2009 are unlatched, positive true, parallel BCD, at DTL/TTL logic levels. As shown in the timing diagram (Figure 1), all data outputs are valid when the STATUS line is low. The STATUS line is high during conversion when erroneous data will be present on the outputs.

TRIGGERING CONVERSIONS

The AD2009 may be triggered internally at six conversions per second, or externally at rates of up to 100 conversions per second. For internal triggering, the Internal Trigger Output (Pin 1) should be connected to the Trigger Input (Pin B). For external triggering, a positive trigger pulse (<500µs width) should be applied to the Trigger Input (Pin B). Whether internal or external triggering is used, the last reading can be held and displayed by grounding or applying logic "0" to the Hold Input. At high conversion rates, the display may flicker unless synchronized to the AC line input, but data outputs will remain valid.

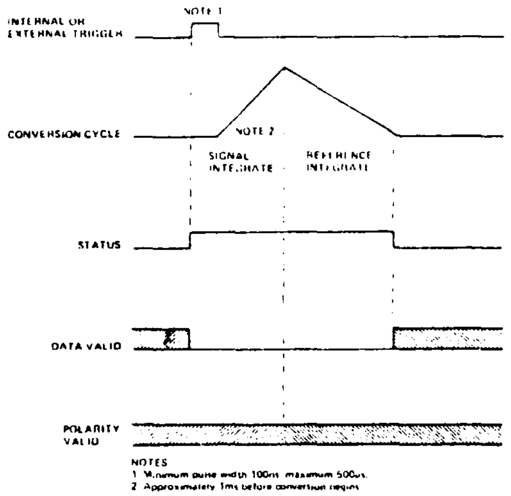
CALIBRATION PROCEDURE

"*WARNING: For the safety of personnel and interconnected equipment, all calibration should be done using a plastic trimming tool only.*"

A precision voltage reference is needed for calibration of the AD2009. The location of calibration potentiometers is shown in Figure 2. Before calibrating the AD2009, allow the unit to warmup to normal operating temperature. Always adjust the zero offset first then the gain.

Zero adjustment: Short the signal input (Pin 2) to the signal ground (Pin 10) and adjust the zero adjustment pot until the meter reads 000.

Gain adjustment. Apply an input of +1.900V (+190.0mV on AD2009/S) and adjust the gain pot until the meter reads 1900 exactly.



NOTE 1
INTERNAL OR EXTERNAL TRIGGER
CONVERSION CYCLE
NOTE 2
SIGNAL INTEGRATE
REFERENCE INTEGRATE
STATUS
DATA VALID
POLARITY VALID
NOTES
1 Minimum pulse width 100ns maximum 500µs
2 Approximately 1ms before conversion begins

Figure 1. AD2009 Timing Diagram

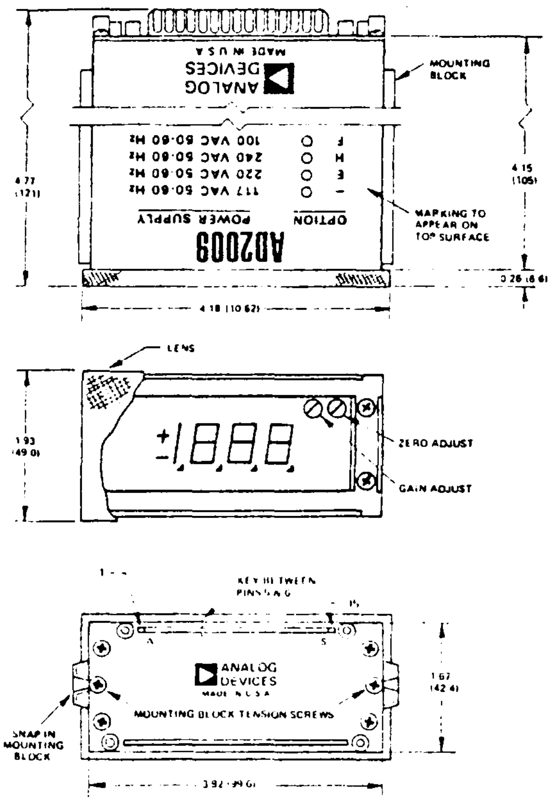
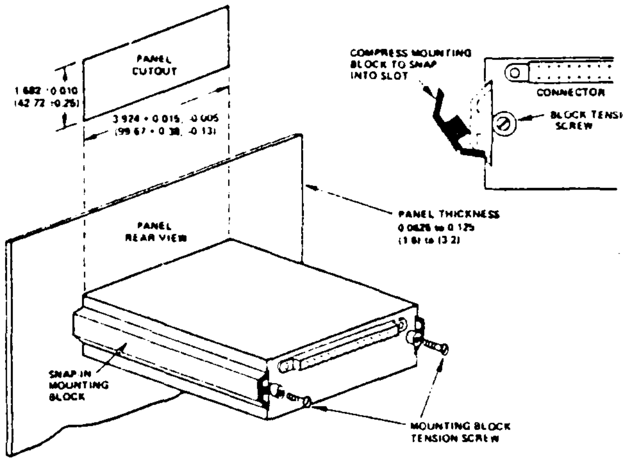


Figure 2. AD2009 Mechanical Outline (Dimensions shown in inches and (mm))



MOUNTING INSTRUCTIONS
1. SLIDE DPM THROUGH PANEL CUTOUT FROM FRONT OF PANEL.
2. SNAP MOUNTING BLOCK INTO SLOT ON DPM SIDES.
3. TIGHTEN MOUNTING BLOCK TENSION SCREWS SNUGLY TO SECURE DPM TO PANEL (DO NOT OVERTIGHTEN!)
4. SNAP LENS ONTO FRONT OF DPM

Figure 3. AD2009 Mounting Instructions (Dimensions shown in inches and (mm))

PIN REF	PIN FUNCTION
1	INTERNAL TRIGGER OUT[1]
2	SIGNAL INPUT
3	STATUS (PRINT)
4	POLARITY
5	BCD 8
6	BCD 2
7	BCD 80
8	BCD 20
9	BCD 800
10	SIGNAL GROUND
11	BCD 400
12	BCD 200
13	DISPLAY BLANK
14	OVERRANGE
15	AC LINE HI

KEY

PIN REF	PIN FUNCTION
A	NO CONNECTION
B	EXTERNAL TRIGGER IN[1]
C	OVERLOAD
D	HOLD
E	BCD 1
F	BCD 4
H	BCD 10
J	BCD 40
K	BCD 100
L	DP3/XX.X
M	DP2/X.XX
N	DIGITAL GROUND
P	DP1/.XXX
R	SHIELD (EARTH GROUND)
S	AC LINE LO

[1] Pin 1 and Pin B must be connected for operation with internal trigger.

Figure 4. AD2009 Signal and Pin Designations

The FTK0040 is a 9 element npn Planar* phototransistor array having exceptionally stable characteristics and high illumination sensitivity. Each transistor is electrically isolated and mounted on 100 mil centers. The case is a plastic compound with transparent resin encapsulation which exhibits stable characteristics under high humidity conditions.

- HIGH ILLUMINATION SENSITIVITY
- EXHIBITS STABLE CHARACTERISTICS UNDER HIGH HUMIDITY CONDITIONS
- ESPECIALLY DESIGNED FOR PUNCHED OR MARKED CARD READING APPLICATIONS
- OTHER APPLICATIONS INCLUDE: OPTICAL ENCODER APPLICATIONS

ABSOLUTE MAXIMUM RATINGS

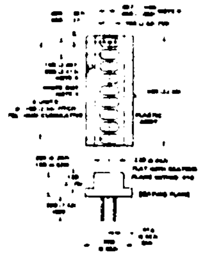
Maximum Temperatures/Humidity	
Storage Temperature	−55°C to +100°C
Operating Junction Temperature	−55°C to +85°C
Relative Humidity at Temperature	98% at 65°C
Maximum Power Dissipation	
Total Dissipation 25 C Case	200 mW
Total Dissipation 25 C Ambient	133 mW
V_{CEO} Collector to Emitter Sustaining Voltage	20 V
Maximum Current	
I_C Collector Current	25 mA

ELECTRICAL CHARACTERISTICS (25°C)

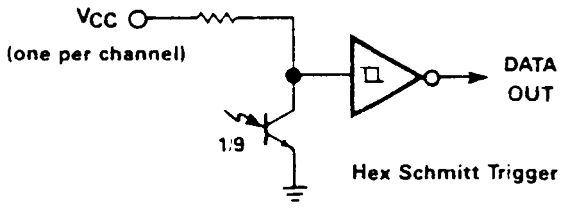
SYMBOL	CHARACTERISTICS	TYP	UNITS	TEST CONDITIONS
I_{CEO}	Collector Dark Current/Cell	4.0	nA	V_{CE} = 5.0 V
$I_{CE(L)}$	Photo Current	200	μA	V_{CE} = 5.0 V, H = 5 mW/cm²
$I_{CE(L)}$	Photo Current	1.75	mA	V_{CE} = 5.0 V, H = 10 mW/cm²
$I_{CE(L)}$	Photo Current	2.25	mA	V_{CE} = 5.0 V, H = 5 mW/cm²
Smin/ Smax	Matching Factor	0.5	—	V_{CE} = 5.0 V, H = 5 mW/cm²
t_r	Light Current Rise Time	4.0	μs	GaAs, I_C = 2.0 mA,
t_f	Light Current Fall Time	—	—	R_L = 100Ω, V_{CC} = 5.0 V
$V_{CE(sat)}$	Collector-Emitter Saturation Voltage	0.16	V	I_C = 500 μA, H = 20 mW/cm²
$V_{CEO(sus)}$	Collector-Emitter Sustaining Voltage	20	V	I_C = 1.0 mA (Pulsed)
BV_{ECO}	Emitter-Collector Breakdown Voltage	7.0	V	I_{EC} = 100 μA

*Planar is a patented Fairchild process

FTK0040



TYPICAL APPLICATION



APPENDIX D

Reactimeter Program

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 1


     1          NEPR1   MACRO CEX1I,CEX2I,CLAMI,Y2TI
     1                  LODE
     1                  LXI H,CEX1I   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;CEX1I
     1                  CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
     1                  LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;HOLD
     1                  STRIN
     1                  LXI H,FLUX2   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;FLUX2
     1                  LODE
     1                  LXI H,CEX2I   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;CEX2I
     1                  CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
     1                  LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;HOLD
     1                  CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
     1                  LXI H,FLUX3   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;FLUX3
     1                  CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
     1                  LXI H,CLAMI   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;CLAMI
     1                  CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
     1                  LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;HOLD
     1                  STRIN
     1                  LXI H,Y2TI    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1                                ;Y2TI
```

```
1                LODE
1                LXI H,CEX1I    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1                               ;CEX1I
1                CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1                LXI H,HOLD     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1                               ;HOLD
1                CALL AD        ;CALL IFPP AD SUBROUTINE TO ADD
1                LXI H,Y2TI     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1                               ;Y2TI
1                STRIN
                 ENDM
1        NEPR2   MACRO AI,YTI
1                LXI H,AI       ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
1                CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1                LXI H,YTI      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1                               ;YTI
1                STRIN
                 ENDM
1        LODE    MACRO
1                CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
1                               ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
1                               ;BY H L REGISTER PAIR
                 ENDM


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 2


1        STRIN   MACRO
1                CALL STR       ;CALL IFPP STR SUBROUTINE THAT STORES THE
1                               ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
```

```
      1                                      ;GIVEN BY THE H L REGISTER PAIR
                           ENDM
      1             NEUCO  MACRO
      1                    CALL AD           ;CALL IFPP AD SUBROUTINE TO AD
      1                    POP H             ;POP THE HIGH AND LOW ADDRESS,STORED BY
      1                                      ;SELCT OFF THE STACK AND LOAD IN THE H L
      1                                      ;REGISTER PAIR
      1                    INX H             ;INCREMENT THE H L REGISTER PAIR
      1                                      ;THE H L REGISTER PAIR CONTAIN THE MEMORY
      1                                      ;LOCATION OF THE LOW ADDRESS BYTE OF CONI
      1                    MOV E,M           ;MOVE THE LOW ADDRESS BYTE OF CONI TO E
      1                    INX H             ;INCREMENT H L REGISTER PAIR
      1                                      ;THE H L REGISTER PAIR CONTAIN THE MEMORY
      1                                      ;LOCATION OF THE HIGH ADDRESS BYTE OF CONI
      1                    MOV H,M           ;MOVE THE HIGH ADDRESS BYTE OF CONI TO H
      1                    MOV L,E           ;MOVE THE LOW ADDRESS BYTE OF CONI TO L
      1                                      ;THE H L REGISTER PAIR CONTAIN THE ADDRESS
      1                                      ;OF CONI
      1                    CALL MUL          ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
      1                    LXI H,FLUX        ;LOAD H L REGISTER PAIR WITH THE ADDRESS
      1                                      ;OF FLUX
      1                    STRIN
                           ENDM
022F                       ORG 022FH
022F    00          INT:   NOP
0230    C9                 RET
023E                       ORG 023EH
023E    00          STR:   NOP
023F    C9                 RET
024D                       ORG 024DH
024D    00          CHS:   NOP
024E    C9                 RET
```

```
026E                    ORG 026EH      ;THE FOLLOWING LABELS ARE FOR THE FLOAT-
026E   00       LOD:    NOP            ;ING POINT ROUTINES THAT ARE IN EPROM AND
026F   C9               RET            ;ARE BY CERTAIN ROUTINES TO PERFORM MATHE
028C                    ORG 028CH      ;MATICAL OPERATIONS.  THESE ROUTINES ARE
028C   00       MUL:    NOP            ;IN ANOTHER PRINT OUT AND ARE GIVEN HERE
028D   C9               RET            ;AS DUMMY PROGRAMS TO BE REFERENCE BY THE
02B4                    ORG 02B4H      ;CROSS ASSEMBLER.
02B4   00       DIV:    NOP
02B5   C9               RET
02D4                    ORG 02D4H
02D4   00       SB:     NOP
02D5   C9               RET
02D7                    ORG 02D7H
02D7   00       AD:     NOP
02D8   C9               RET
054A                    ORG 054AH
```

```
054A   00       INP:    NOP
054B   C9               RET
060C                    ORG 060CH
060C   00       OU:     NOP
060D   C9               RET
0000            START:  ORG 0000H
0000   31FF10           LXI SP,10FFH ;LOAD STACK POINTER TO HI 10 AND LO FF
0003   218010           LXI H,DMANT
0006   36F0             MVI M,360Q
0008   2C               INR L
```

```
0009    2C                  INR L
000A    2C                  INR L
000B    36FE                MVI M,376Q
000D    2C                  INR L
000E    2C                  INR L
000F    36FF                MVI M,377Q
0011    1607                MVI D,7        ;SET D EQUAL TO 7
0013    218610              LXI H, Y2T1    ;LOAD H L REGISTER PAIR WITH Y2T1
0016    3E00                MVI A,0
0018    77          INITZ:  MOV M,A
0019    2C                  INR L
001A    2C                  INR L
001B    2C                  INR L
001C    2C                  INR L
001D    15                  DCR D
001E    C21800              JNZ INITZ
0021    CD2F02              CALL INT       ;CALL IFPP INT SUBROUTINE TO
                                           ;INITIALIZE THE FLOATING-
                                           ;POINT-ROUTINES
0024    C35C00              JMP WAIT       ;JUMP TO WAIT AND BEGIN PROGRAM
0028                INTRO:  ORG 0028H
0028    D30B        RESTR:  OUT 11         ;GENERATE A DEVICE SELECT PULSE
002A    D30A        INPUT:  OUT 10         ;GENERATE A DEVICE SELECT PULSE
                                           ;TO DEVICE TEN, THIS CASE FOUR 7475
                                           ;LATCHES
002C    218410              LXI H,DMANT+4;ADDRESS OF TENTHS DIGIT OF DMANT, WHICH
                                           ;IS THE FIRST DIGIT TO BE READ INTO THE
                                           ;ACCUMULATOR
002F    DB01                IN  1          ;READ TENTHS DIGIT FROM DEVICE 1 INTO
                                           ;ACCUMULATOR
0031    77                  MOV M,A        ;MOVE TENTHS DIGIT TO MEMORY ADDRESS BY
                                           ;H L REGISTER PAIR
```

```
0032    2D              DCR L
0033    2D              DCR L
0034    DB02            IN   2          ;READ ONES DIGIT FROM DEVICE 2 INTO
                                        ;ACCUMULATOR
0036    77              MOV M,A
0037    2D              DCR L
0038    DB03            IN   3          ;READ TENS DIGIT FROM DEVICE 3 INTO
                                        ;ACCUMULATOR
003A    77              MOV M,A         ;MOVE TENS DIGIT TO MEMORY ADDRSS BY H L


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 4


                                        ;REGISTER PAIR
003B    DB00            IN   0          ;READ GRAY
003D    07              RLC             ;ROTATE ACCUMULATOR LEFT
003E    07              RLC
003F    21CE10          LXI H,GRAYC
0042    77              MOV M,A         ;MOVE ACCUMULATOR TO MEMORY LOCATION GIVEN
                                        ;BY H L REGISTER PAIR
0043    218010          LXI H,DMANT     ;LOAD H L REGISTERS WITH ADDRESS OF DMANT
0046    CD4A05          CALL INP        ;CALL IFPP SUBROUTINE TO CHANGE A STRING
                                        ;OF BCD DIGITS INTO A BINARY FLOATING-
                                        ;POINT-NUMBER
0049    21C610          LXI H,MANT      ;LOAD H L RECISTERS WITH ADDRESS OF MANT
      1          +      STRIN
004C  1 CD3E02   +      CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
      1          +                      ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
      1          +                      ;GIVEN BY THE H L REGISTER PAIR
004F    00              NOP
```

```
0050   21CE10   SELCT: LXI H,GRAYC    ;LOAD THE ADDRESS OF GRAYC INTO H L
                                      ;REGISTER PAIR
                                      ;GRAYC CONTAINS THE LOW ADDRESS OF THE
                                      ;FIRST BYTE  OF A FOUR BYTE MEMORY LOCA-
                                      ;TION, CONTAINS THE ADDRESS OF THE ROU-
                                      ;TINE HRFC OR LRFC AND THE CONSTANT CONI
0053   6E              MOV L,M        ;MOVE KEITHLEY CODE INTO L
0054   2601            MVI H,001Q     ;LOAD H WITH 001
0056   5E              MOV E,M        ;MOVE LOW ADDRESS INTO E
0057   2C              INR L          ;INCREMENT L
0058   E5              PUSH H         ;STORE CONTENTS OF H L REGISTER PAIR IN
                                      ;STACK POINTER
0059   66              MOV H,M        ;MOVE HIGH ADDRESS TO H
005A   6B              MOV L,E        ;MOVE LOW ADDRESS FROM E TO L
005B   E9              PCHL           ;PUSH CONTENTS OF H AND L INTO PROGRAM
                                      ;COUNTER AND JUMP TO THAT ADDRESS
005C   FB       WAIT:  EI             ;ENABLE FLAG INTERRUPT
005D   00              NOP            ;NO OPERATION
                                      ;THE PURPOSE OF THIS LOOP IS TO LET THE
                                      ;MICROCOMPUTER IDLE WHILE WAITING FOR THE
                                      ;START OF A NEW TIME INTERVAL SIGNAL FROM
                                      ;555 MONOSTABLE MULTIVIBRATOR CLOCK CIR-
                                      ;CUIT THAT PRODUCES AN INTERRUPT COMMAND
                                      ;EVERY 0.2 SECONDS.
005E   C35C00          JMP WAIT       ;JUMP TO WAIT
0061   21C610   LRFC:  LXI H,MANT     ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                      ;OF MANT
     1          +      LODE
0064 1 CD6E02   +      CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     1          +                     ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     1          +                     ;BY H L REGISTER PAIR
0067   216401          LXI H,CONT2    ;LOAD THE H L REGISTER PAIR WITH THE AD-
```

```
                                                    ;DRESS OF CONT2
  006A   CD8C02               CALL MUL              ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
```

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 5
```

```
  006D   21CC01               LXI H,ONE             ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                                    ;OF ONE
       1           +          NEUCO
  0070 1 CDD702    +          CALL AD               ;CALL IFPP AD SUBROUTINE TO AD
  0073 1 E1        +          POP H                 ;POP THE HIGH AND LOW ADDRESS,STORED BY
       1           +                                ;SELCT OFF THE STACK AND LOAD IN THE H L
       1           +                                ;REGISTER PAIR
  0074 1 23        +          INX H                 ;INCREMENT THE H L REGISTER PAIR
       1           +                                ;THE H L REGISTER PAIR CONTAIN THE MEMORY
       1           +                                ;LOCATION OF THE LOW ADDRESS BYTE OF CONI
  0075 1 5E        +          MOV E,M               ;MOVE THE LOW ADDRESS BYTE OF CONI TO E
  0076 1 23        +          INX H                 ;INCREMENT H L REGISTER PAIR
       1           +                                ;THE H L REGISTER PAIR CONTAIN THE MEMORY
       1           +                                ;LOCATION OF THE HIGH ADDRESS BYTE OF CONI
  0077 1 66        +          MOV H,M               ;MOVE THE HIGH ADDRESS BYTE OF CONI TO H
  0078 1 6B        +          MOV L,E               ;MOVE THE LOW ADDRESS BYTE OF CONI TO L
       1           +                                ;THE H L REGISTER PAIR CONTAIN THE ADDRESS
       1           +                                ;OF CONI
  0079 1 CD8C02    +          CALL MUL              ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  007C 1 21CA10    +          LXI H,FLUX            ;LOAD H L REGISTER PAIR WITH THE ADDRESS
       1           +                                ;OF FLUX
       2           +          STRIN
  007F 2 CD3E02    +          CALL STR              ;CALL IFPP STR SUBROUTINE THAT STORES THE
       2           +                                ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
```

```
     2          +                         ;GIVEN BY THE H L REGISTER PAIR
  0082   C3A600            JMP DIRCT      ;JUMP TO ROUTINE DIRCT
  0085   21C610    HRFC:   LXI H,MANT     ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF MANT
     1          +          LODE
  0088 1 CD6E02 +          CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     1          +                         ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     1          +                         ;BY H L REGISTER PAIR
  008B   216801            LXI H,CONT7    ;LOAD THE H L REGISTER PAIR WITH THE AD-
                                          ;DRESS OF CONT7
  008E   CD8C02            CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  0091   21D001            LXI H,THREE    ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF THREE
     1          +          NEUCO
  0094 1 CDD702 +          CALL AD        ;CALL IFPP AD SUBROUTINE TO AD
  0097 1 E1     +          POP H          ;POP THE HIGH AND LOW ADDRESS,STORED BY
     1          +                         ;SELCT OFF THE STACK AND LOAD IN THE H L
     1          +                         ;REGISTER PAIR
  0098 1 23     +          INX H          ;INCREMENT THE H L REGISTER PAIR
     1          +                         ;THE H L REGISTER PAIR CONTAIN THE MEMORY
     1          +                         ;LOCATION OF THE LOW ADDRESS BYTE OF CONI
  0099 1 5E     +          MOV E,M        ;MOVE THE LOW ADDRESS BYTE OF CONI TO E
  009A 1 23     +          INX H          ;INCREMENT H L REGISTER PAIR
     1          +                         ;THE H L REGISTER PAIR CONTAIN THE MEMORY
     1          +                         ;LOCATION OF THE HIGH ADDRESS BYTE OF CONI
  009B 1 66     +          MOV H,M        ;MOVE THE HIGH ADDRESS BYTE OF CONI TO H
  009C 1 6B     +          MOV L,E        ;MOVE THE LOW ADDRESS BYTE OF CONI TO L

8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 6
     1         +                    ;THE H L REGISTER PAIR CONTAIN THE ADDRESS
     1         +                    ;OF CONI
009D 1 CD8C02  +        CALL MUL    ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
00A0 1 21CA10  +        LXI H,FLUX  ;LOAD H L REGISTER PAIR WITH THE ADDRESS
     1         +                    ;OF FLUX
     2         +        STRIN
00A3 2 CD3E02  +        CALL STR    ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2         +                    ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2         +                    ;GIVEN BY THE H L REGISTER PAIR
00A6   21CF10   DIRCT: LXI H,MEM    ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                    ;WITH MEM
00A9   35               DCR M       ;DECREMENT MEM BY ONE
00AA   CABC00           JZ   LODIR  ;JUMP TO LODIR IF THE RESULT IS ZERO
00AD   21CA10           LXI H,FLUX  ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                    ;OF FLUX
     1         +        LODE
00B0 1 CD6E02  +        CALL LOD    ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     1         +                    ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     1         +                    ;BY H L REGISTER PAIR
00B3   21A210           LXI H,FLUX2 ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                    ;OF FLUX2
     1         +        STRIN
00B6 1 CD3E02  +        CALL STR    ;CALL IFPP STR SUBROUTINE THAT STORES THE
     1         +                    ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     1         +                    ;GIVEN BY THE H L REGISTER PAIR
00B9   C35C00           JMP WAIT    ;JUMP TO WAIT PROGRAM
00BC   21CA10   LODIR: LXI H,FLUX   ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                    ;OF FLUX
     1         +        LODE
00BF 1 CD6E02  +        CALL LOD    ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     1         +                    ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     1              +                       ;BY H L REGISTER PAIR
  00C2    21A610              LXI H,FLUX3   ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                            ;OF FLUX3
     1              +         STRIN
  00C5  1 CD3E02    +         CALL  STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     1              +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     1              +                       ;GIVEN BY THE H L REGISTER PAIR
  00C8    21CF10              LXI H,MEM     ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                            ;OF MEM
  00CB    3602                MVI M,002Q    ;SET MEM EQUAL TO TWO
  00CD    C30014              JMP PREC1     ;JUMP TO PREC1
  00D0    3ADA01     ZERO:    LDA PLUS      ;LOAD ACCUMULATOR WITH PLUS
  00D3    D309                OUT 9         ;OUTPUT ACCUMULATOR TO DEVICE 9
  00D5    C35C00              JMP WAIT      ;JUMP TO WAIT
  00D8    1D         CHECK:   DCR E         ;DECREMENT REGISTER E
  00D9    CCDF00              CZ  WOUT      ;CALL SUBROUTINE WOUT IF RESULT IS ZERO
  00DC    2D                  DCR L         ;DECREMENT L
  00DD    7E                  MOV A,M       ;MOVE THE DATA FROM MEMORY LOCATION
                                            ;ADDRESSED BY THE H L REGISTER PAIR TO
                                            ;THE ACCUMULATOR


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 7


  00DE    C9                  RET           ;RETURN
  00DF    21D901     WOUT:    LXI H,BLANK+1;LOAD H L REGISTER PAIR WITH BLANK + 1
  00E2    1E01                MVI E,1       ;SET REGISTER EQUAL TO 1
  00E4    C9                  RET           ;RETURN
  1400                        ORG 1400H
  1400    219E10     PREC1:   LXI H,FLUX1   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
                                        ;FLUX1
          1           +        NEPR1 CEX11,CEX21,CLAM1,Y2T1
          2           +        LODE
     1403 2 CD6E02    +        CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
          2           +                      ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
          2           +                      ;BY H L REGISTER PAIR
     1406 1 216C01    +        LXI H,CEX11   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;CEX1I
     1409 1 CD8C02    +        CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
     140C 1 21C210    +        LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;HOLD
          2           +        STRIN
     140F 2 CD3E02    +        CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
          2           +                      ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
          2           +                      ;GIVEN BY THE H L REGISTER PAIR
     1412 1 21A210    +        LXI H,FLUX2   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;FLUX2
          2           +        LODE
     1415 2 CD6E02    +        CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
          2           +                      ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
          2           +                      ;BY H L REGISTER PAIR
     1418 1 218401    +        LXI H,CEX21   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;CEX2I
     141B 1 CD8C02    +        CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
     141E 1 21C210    +        LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;HOLD
     1421 1 CDD702    +        CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
     1424 1 21A610    +        LXI H,FLUX3   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;FLUX3
     1427 1 CDD702    +        CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
     142A 1 219C01    +        LXI H,CLAM1   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
          1           +                      ;CLAMI
```

```
 142D 1 CD8C02   +        CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
 1430 1 21C210   +        LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1          +                      ;HOLD
      2          +        STRIN
 1433 2 CD3E02   +        CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
      2          +                      ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
      2          +                      ;GIVEN BY THE H L REGISTER PAIR
 1436 1 218610   +        LXI H,Y2T1    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1          +                      ;Y2TI
      2          +        LODE
 1439 2 CD6E02   +        CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
      2          +                      ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
      2          +                      ;BY H L REGISTER PAIR


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 8


 143C 1 216C01   +        LXI H,CEX11   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1          +                      ;CEXlI
 143F 1 CD8C02   +        CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
 1442 1 21C210   +        LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1          +                      ;HOLD
 1445 1 CD0702   +        CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
 1448 1 218610   +        LXI H,Y2T1    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1          +                      ;Y2TI
      2          +        STRIN
 144B 2 CD3E02   +        CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
      2          +                      ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
      2          +                      ;GIVEN BY THE H L REGISTER PAIR
      1          +        NEPR2 A1,YT1
```

```
144E 1 21B401   +         LXI H,A1      ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
1451 1 CD8C02   +         CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1454 1 21AA10   +         LXI H,YT1     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;YTI
     2          +         STRIN
1457 2 CD3E02   +         CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                       ;GIVEN BY THE H L REGISTER PAIR
145A   21A210   PREC2:    LXI H,FLUX2   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                        ;FLUX1
     1          +         NEPR1 CEX12,CEX22,CLAM2,Y2T2
     2          +         LODE
145D 2 CD6E02   +         CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                       ;BY H L REGISTER PAIR
1460 1 217001   +         LXI H,CEX12   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;CEX1I
1463 1 CD8C02   +         CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1466 1 21C210   +         LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;HOLD
     2          +         STRIN
1469 2 CD3E02   +         CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                       ;GIVEN BY THE H L REGISTER PAIR
146C 1 21A210   +         LXI H,FLUX2   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;FLUX2
     2          +         LODE
146F 2 CD6E02   +         CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                       ;BY H L REGISTER PAIR
1472 1 218801   +         LXI H,CEX22   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;CEX2I
```

```
  1475 1 CD8C02    +         CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  1478 1 21C210    +         LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;HOLD
  147B 1 CDD702    +         CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
  147E 1 21A610    +         LXI H,FLUX3   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;FLUX3


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 9


  1481 1 CDD702    +         CALL AD       ;CALL IFPP AD SUBROUTINE TO ADD
  1484 1 21A001    +         LXI H,CLAM2   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;CLAMI
  1487 1 CD8C02    +         CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  148A 1 21C210    +         LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;HOLD
       2           +         STRIN
  148D 2 CD3E02    +         CALL STR      ;CALL IFPP STR SUBROUTINE THAT STORES THE
       2           +                       ;FLOATING-POINT-ACCUMULATCR IN THE ADDRESS
       2           +                       ;GIVEN BY THE H L REGISTER PAIR
  1490 1 218A10    +         LXI H,Y2T2    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;Y2TI
       2           +         LODE
  1493 2 CD6E02    +         CALL LOD      ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
       2           +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
       2           +                       ;BY H L REGISTER PAIR
  1496 1 217001    +         LXI H,CEX12   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       1           +                       ;CEXlI
  1499 1 CD8C02    +         CALL MUL      ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  149C 1 21C210    +         LXI H,HOLD    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
     1          +                        ;HOLD
149F 1 CDD702   +        CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
14A2 1 218A10   +        LXI H,Y2T2      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                        ;Y2TI
     2          +        STRIN
14A5 2 CD3E02   +        CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                        ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                        ;GIVEN BY THE H L REGISTER PAIR
     1          +        NEPR2 A2,YT2
14A8 1 21B801   +        LXI H,A2        ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
14AB 1 CD8C02   +        CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14AE 1 21AE10   +        LXI H,YT2       ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                        ;YTI
     2          +        STRIN
14B1 2 CD3E02   +        CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                        ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                        ;GIVEN BY THE H L REGISTER PAIR
14B4   219E10   PREC3:   LXI H,FLUX1     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                         ;FLUX1
     1          +        NEPR1 CEX13,CEX23,CLAM3,Y2T3
     2          +        LODE
14B7 2 CD6E02   +        CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                        ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                        ;BY H L REGISTER PAIR
14BA 1 217401   +        LXI H,CEX13     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                        ;CEX1I
14BD 1 CD8C02   +        CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14C0 1 21C210   +        LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                        ;HOLD
     2          +        STRIN
14C3 2 CD3E02   +        CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
```

```
     2          +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                       ;GIVEN BY THE H L REGISTER PAIR
14C6 1 21A210   +       LXI H,FLUX2     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;FLUX2
     2          +       LODE
14C9 2 CD6E02   +       CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                       ;BY H L REGISTER PAIR
14CC 1 218C01   +       LXI H,CEX23     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;CEX2I
14CF 1 CD8C02   +       CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14D2 1 21C210   +       LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;HOLD
14D5 1 CDD702   +       CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
14D8 1 21A610   +       LXI H,FLUX3     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;FLUX3
14DB 1 CDD702   +       CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
14DE 1 21A401   +       LXI H,CLAM3     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;CLAMI
14E1 1 CD8C02   +       CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14E4 1 21C210   +       LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                       ;HOLD
     2          +       STRIN
14E7 2 CD3E02   +       CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                       ;GIVEN BY THE H L REGISTER PAIR
14EA 1 218E10   +       LXI H,Y2T3      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
     1          +                          ;Y2TI
     2          +          LODE
14ED 2 CD6E02   +          CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                          ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                          ;BY H L REGISTER PAIR
14F0 1 217401   +          LXI H,CEX13     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                          ;CEX1I
14F3 1 CD8C02   +          CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14F6 1 21C210   +          LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                          ;HOLD
14F9 1 CDD702   +          CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
14FC 1 218E10   +          LXI H,Y2T3      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                          ;Y2TI
     2          +          STRIN
14FF 2 CD3E02   +          CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                          ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                          ;GIVEN BY THE H L REGISTER PAIR
     1          +          NEPR2 A3,YT3
1502 1 21BC01   +          LXI H,A3        ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
1505 1 CD8C02   +          CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1508 1 21B210   +          LXI H,YT3       ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                          ;YTI
     2          +          STRIN
150B 2 CD3E02   +          CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
```

```
     2          +                          ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                          ;GIVEN BY THE H L REGISTER PAIR
```

```
150E   219E10   PREC4: LXI H,FLUX1  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                    ;FLUX1
     1          +      NEPRI CEX14,CEX24,CLAM4,Y2T4
     2          +      LODE
1511 2 CD6E02   +      CALL LOD     ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                   ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                   ;BY H L REGISTER PAIR
1514 1 217801   +      LXI H,CEX14  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;CEX1I
1517 1 CD8C02   +      CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
151A 1 21C210   +      LXI H,HOLD   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;HOLD
     2          +      STRIN
151D 2 CD3E02   +      CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                   ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                   ;GIVEN BY THE H L REGISTER PAIR
1520 1 21A210   +      LXI H,FLUX2  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;FLUX2
     2          +      LODE
1523 2 CD6E02   +      CALL LOD     ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                   ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                   ;BY H L REGISTER PAIR
1526 1 219001   +      LXI H,CEX24  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;CEX2I
1529 1 CD8C02   +      CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
152C 1 21C210   +      LXI H,HOLD   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;HOLD
152F 1 CDD702   +      CALL AD      ;CALL IFPP AD SUBROUTINE TO ADD
1532 1 21A610   +      LXI H,FLUX3  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                   ;FLUX3
1535 1 CDD702   +      CALL AD      ;CALL IFPP AD SUBROUTINE TO ADD
1538 1 21A801   +      LXI H,CLAM4  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
     1           +                          ;CLAMI
 153B 1 CD8C02   +        CALL MUL          ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
 153E 1 21C210   +        LXI H,HOLD        ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                          ;HOLD
     2           +        STRIN
 1541 2 CD3E02   +        CALL STR          ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                          ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                          ;GIVEN BY THE H L REGISTER PAIR
 1544 1 219210   +        LXI H,Y2T4        ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                          ;Y2TI
     2           +        LODE
 1547 2 CD6E02   +        CALL LOD          ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2           +                          ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2           +                          ;BY H L REGISTER PAIR
 154A 1 217801   +        LXI H,CEX14       ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                          ;CEXLI
 154D 1 CD8C02   +        CALL MUL          ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
```

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 12


 1550 1 21C210   +        LXI H,HOLD        ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                          ;HOLD
 1553 1 CDD702   +        CALL AD           ;CALL IFPP AD SUBROUTINE TO ADD
 1556 1 219210   +        LXI H,Y2T4        ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                          ;Y2TI
     2           +        STRIN
 1559 2 CD3E02   +        CALL STR          ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                          ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                          ;GIVEN BY THE H L REGISTER PAIR
     1           +        NEPR2 A4,YT4
155C 1 21C001    +        LXI H,A4       ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
155F 1 CD8C02    +        CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1562 1 21B610    +        LXI H,YT4      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;YTI
     2           +        STRIN
1565 2 CD3E02    +        CALL STR       ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                       ;GIVEN BY THE H L REGISTER PAIR
1568   219E10     PREC5: LXI H,FLUX1    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                         ;FLUX1
     1           +        NEPR1 CEX15,CEX25,CLAM5,Y2T5
     2           +        LODE
156B 2 CD6E02    +        CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2           +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2           +                       ;BY H L REGISTER PAIR
156E 1 217C01    +        LXI H,CEX15    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;CEX1I
1571 1 CD8C02    +        CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1574 1 21C210    +        LXI H,HOLD     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;HOLD
     2           +        STRIN
1577 2 CD3E02    +        CALL STR       ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                       ;GIVEN BY THE H L REGISTER PAIR
157A 1 21A210    +        LXI H,FLUX2    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;FLUX2
     2           +        LODE
157D 2 CD6E02    +        CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2           +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2           +                       ;BY H L REGISTER PAIR
1580 1 219401    +        LXI H,CEX25    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
     1           +                         ;CEX2I
1583 1 CD8C02    +          CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1586 1 21C210    +          LXI H,HOLD     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;HOLD
1589 1 CDD702    +          CALL AD        ;CALL IFPP AD SUBROUTINE TO ADD
158C 1 21A610    +          LXI H,FLUX3    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;FLUX3
158F 1 CDD702    +          CALL AD        ;CALL IFPP AD SUBROUTINE TO ADD
1592 1 21AC01    +          LXI H,CLAM5    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;CLAMI


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 13


1595 1 CD8C02    +          CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1598 1 21C210    +          LXI H,HOLD     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;HOLD
     2           +          STRIN
159B 2 CD3E02    +          CALL STR       ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                         ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                         ;GIVEN BY THE H L REGISTER PAIR
159E 1 219610    +          LXI H,Y2T5     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;Y2TI
     2           +          LODE
15A1 2 CD6E02    +          CALL LOD       ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2           +                         ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2           +                         ;BY H L REGISTER PAIR
15A4 1 217C01    +          LXI H,CEX15    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                         ;CEX1I
15A7 1 CD8C02    +          CALL MUL       ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
```

```
15AA 1 21C210    +          LXI H,HOLD   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;HOLD
15AD 1 CDD702    +          CALL AD      ;CALL IFPP AD SUBROUTINE TO ADD
15B0 1 219610    +          LXI H,Y2T5   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;Y2TI
     2           +          STRIN
15B3 2 CD3E02    +          CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                       ;GIVEN BY THE H L REGISTER PAIR
     1           +          NEPR2 A5,YT5
15B6 1 21C401    +          LXI H,A5     ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
15B9 1 CD8C02    +          CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
15BC 1 21BA10    +          LXI H,YT5    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;YTI
     2           +          STRIN
15BF 2 CD3E02    +          CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2           +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2           +                       ;GIVEN BY THE H L REGISTER PAIR
15C2   219E10     PREC6: LXI H,FLUX1  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                         ;FLUX1
     1           +          NEPR1 CEX16,CEX26,CLAM6,Y2T6
     2           +          LODE
15C5 2 CD6E02    +          CALL LOD     ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2           +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2           +                       ;BY H L REGISTER PAIR
15C8 1 218001    +          LXI H,CEX16  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;CEX1I
15CB 1 CD8C02    +          CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
15CE 1 21C210    +          LXI H,HOLD   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1           +                       ;HOLD
     2           +          STRIN
15D1 2 CD3E02    +          CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
```

```
      2        +                         ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
      2        +                         ;GIVEN BY THE H L REGISTER PAIR
 15D4 1 21A210 +         LXI H,FLUX2     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
```

```
      1        +                         ;FLUX2
      2        +         LODE
 15D7 2 CD6E02 +         CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
      2        +                         ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
      2        +                         ;BY H L REGISTER PAIR
 15DA 1 219801 +         LXI H,CEX26     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1        +                         ;CEX2I
 15DD 1 CD8C02 +         CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
 15E0 1 21C210 +         LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1        +                         ;HOLD
 15E3 1 CDD702 +         CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
 15E6 1 21A610 +         LXI H,FLUX3     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1        +                         ;FLUX3
 15E9 1 CDD702 +         CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
 15EC 1 21B001 +         LXI H,CLAM6     ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1        +                         ;CLAMI
 15EF 1 CD8C02 +         CALL MUL        ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
 15F2 1 21C210 +         LXI H,HOLD      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
      1        +                         ;HOLD
      2        +         STRIN
 15F5 2 CD3E02 +         CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
      2        +                         ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
      2        +                         ;GIVEN BY THE H L REGISTER PAIR
```

```
15F8 1 219A10   +        LXI H,Y2T6   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                     ;Y2TI
     2          +        LODE
15FB 2 CD6E02   +        CALL LOD     ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
     2          +                     ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
     2          +                     ;BY H L REGISTER PAIR
15FE 1 218001   +        LXI H,CEX16  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                     ;CEX1I
1601 1 CD8C02   +        CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1604 1 21C210   +        LXI H,HOLD   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                     ;HOLD
1607 1 CDD702   +        CALL AD      ;CALL IFPP AD SUBROUTINE TO ADD
160A 1 219A10   +        LXI H,Y2T6   ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                     ;Y2TI
     2          +        STRIN
160D 2 CD3E02   +        CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                     ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                     ;GIVEN BY THE H L REGISTER PAIR
     1          +        NEPR2 A6,YT6
1610 1 21C801   +        LXI H,A6     ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
1613 1 CD8C02   +        CALL MUL     ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1616 1 21BE10   +        LXI H,YT6    ;LOAD H L REGISTER PAIR WITH ADDRESS OF
     1          +                     ;YTI
     2          +        STRIN
1619 2 CD3E02   +        CALL STR     ;CALL IFPP STR SUBROUTINE THAT STORES THE
     2          +                     ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
     2          +                     ;GIVEN BY THE H L REGISTER PAIR
161C   21A610   TRANS: LXI H,FLUX3  ;LOAD H L REGISTER PAIR WITH THE ADDRESS

8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 15
```

```
                                          ;OF FLUX3
       1          +       LODE
161F 1 CD6E02     +       CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
       1          +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
       1          +                       ;BY H L REGISTER PAIR
1622     219E10           LXI H,FLUX1     ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF FLUX1
       1          +       STRIN
1625 1 CD3E02     +       CALL STR        ;CALL IFPP STR SUBROUTINE THAT STORES THE
       1          +                       ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
       1          +                       ;GIVEN BY THE H L REGISTER PAIR
1628     21AA10    REACT: LXI H,YT1       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF YT1
       1          +       LODE
162B 1 CD6E02     +       CALL LOD        ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
       1          +                       ;-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
       1          +                       ;BY H L REGISTER PAIR
162E     21AE10           LXI H,YT2       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF YT2
1631     CDD702           CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
1634     21B210           LXI H,YT3       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF YT3
1637     CDD702           CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
163A     21B610           LXI H,YT4       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF YT4
163D     CDD702           CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
1640     21BA10           LXI H,YT5       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                          ;OF YT5
1643     CDD702           CALL AD         ;CALL IFPP AD SUBROUTINE TO ADD
1646     21BE10           LXI H,YT6       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
```

```
                                              ;OF YT6
  1649   CDD702          CALL AD          ;CALL IFPP AD SUBROUTINE TO ADD
  164C   21A610          LXI H,FLUX3      ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                              ;OF FLUX3
  164F   CDB402          CALL DIV         ;CALL IFPP DIV SUBROUTINE TO DIVIDE
  1652   CD4D02          CALL CHS         ;CALL IFPP CHS SUBROUTINE TO CHANGE THE
                                              ;SIGN OF THE FLOATING-POINT-ACCUMULATOR
  1655   21CC01          LXI H,ONE        ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                              ;OF ONE
  1658   CDD702          CALL AD          ;CALL IFPP AD SUBROUTINE  TO ADD
  165B   21D401          LXI H,TEN5       ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                              ;OF TEN5
  165E   CD8C02          CALL MUL         ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
  1661   214001          LXI H,OFSET      ;LOAD H L REGISTER PAIR WITH ADDRESS OF
                                              ;OFSET
  1664   CDD702          CALL AD          ;CALL IFPP AD SUBROUTINE TO ADD
  1667   21D010          LXI H,RHO        ;LOAD H L REGISTER PAIR WITH THE ADDRESS
                                              ;OF RHO
  166A   CD0C06          CALL OU          ;CALL THE IFPP OU SUBROUTINE THAT STORES
                                              ;THE FLOATING-POINT-ACCUMULATOR IN THE AD-
                                              ;DRESS BY THE H L REGISTER PAIR IN BCD


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 16


                                              ;FORMAT
  166D   3AD910   OUTPT: LDA RHO+9        ;LOAD THE ACCUMULATOR WITH THE VALUE OF
                                              ;RHO+9
                                              ;THIS VALUE IS TESTED TO DETERMINE IF A
                                              ;BLANK OR THE NUMBER 025,WHICH MEANS AN
                                              ;EXPONENT FOLLOWS
1670    06F0                MVI B,360Q        ;LOAD B WITH THE VALUE 360(CODE FOR BLANK)
1672    B8                  CMP B             ;COMPARE B WITH THE ACCUMULATOR, TEST FOR
                                              ;BLANK IN RHO+9
1673    CA9316              JZ  CHK1          ;JUMP TO CHK1 IF THE RESULT IS ZERO
1676    3E00        OTPTO:  MVI A,0H          ;LOAD A WITH 0
1678    1E01                MVI E, 1          ;SET REGISTER EQUAL TO 1
167A    D304        OTNUM:  OUT 4             ;OUTPUT ACCUMULATOR TO DEVICE 4
167C    CDD800              CALL CHECK        ;CALL SUBROUTINE CHECK
167F    D305                OUT 5             ;OUTPUT ACCUMULATOR TO DEVICE 5
1681    CDD800              CALL CHECK        ;CALL SUBROUTINE CHECK
1684    D306                OUT 6             ;OUTPUT ACCUMULATOR TO DEVICE 6
1686    CDD800              CALL CHECK        ;CALL SUBROUTINE CHECK
1689    D307                OUT 7             ;OUTPUT ACCUMULATOR TO DEVICE 7
168B    CDD800              CALL CHECK        ;CALL SUBROUTINE CHECK
168E    D308                OUT 8             ;OUTPUT ACCUMULATOR TO DEVICE 8
1690    C3ED16              JMP SIGN          ;JUMP TO SIGN
1693    3AD110      CHK1:   LDA RHO+1         ;LOAD ACCUMULATOR WITH RHO+1
1696    06FE                MVI B,376Q        ;LOAD REGISTER B WITH 376Q
1698    B8                  CMP B             ;COMPARE B WITH ACCUMULATOR
1699    CA7616              JZ  OTPTO         ;JUMP TO OTPTO
169C    3AD610              LDA RHO+6         ;LOAD ACCUMULATOR WITH RHO+6
169F    06FE                MVI B,376Q        ;LOAD B WITH 376Q(CODE FOR DECIMAL POINT)
16A1    B8                  CMP B             ;COMPARE B WITH ACCUMULATOR
16A2    CAC916              JZ  OTPT5         ;JUMP TO OTPT5
16A5    3AD510              LDA RHO+5         ;LOAD ACCUMULATOR WITH RHO+5
16A8    06FE                MVI B,376Q        ;LOAD B WITH 376Q(CODE FOR DECIMAL POINT)
16AA    B8                  CMP B             ;COMPARE B WITH ACCUMULATOR
16AB    CAD216              JZ  OTPT4         ;JUMP TO OTPT4
16AE    3AD410              LDA RHO+4         ;LOAD ACCUMULATOR WITH RHO+4
16B1    06FE                MVI B,376Q        ;LOAD B WITH 376Q(CODE FOR DECIMAL POINT)
16B3    B8                  CMP B             ;COMPARE B WITH ACCUMULATOR
```

```
16B4    CAD816             JZ   OTPT3      ;JUMP TO OTPT3
16B7    3AD310             LDA  RHO+3      ;LOAD ACCUMULATOR WITH RHO+3
16BA    06FE               MVI  B,376Q     ;LOAD B WITH 376Q (CODE FOR DECIMAL POINT)
16BC    B8                 CMP  B          ;COMPARE B WITH ACCUMULATOR
16BD    CAE416             JZ   OTPT2      ;JUMP TO OTPT2
16C0    21D110             LXI  H,RHO + 1;LOAD H L REGISTER PAIR WITH RHO + 1
16C3    7E                 MOV  A,M        ;MOVE THE DATA FROM THE MEMORY LOCATION
                                           ;ADDRESSED BY THE H L REGISTER PAIR TO
                                           ;THE ACCUMULATOR
16C4    1E01               MVI  E,1        ;SET REGISTER EQUAL TO ONE
16C6    C37A16             JMP  OTNUM      ;JUMP TO OTNUM
16C9    21D510    OTPT5:   LXI  H,RHO+5    ;LOAD H L REGISTER PAIR WITH RHO+5
16CC    7E                 MOV  A,M        ;MOVE THE DATA FROM THE MEMORY LOCATION
                                           ;ADDRESS BY THE H L REGISTER PAIR TO
```

```
                                           ;THE ACCUMULATOR
16CD    1E05               MVI  E,5        ;SET REGISTER EQUAL TO FIVE
16CF    C37A16             JMP  OTNUM      ;JUMP TO OTNUM
16D2    21D410    OTPT4:   LXI  H,RHO+4    ;LOAD H L REGISTER PAIR WITH RHO+4
16D5    7E                 MOV  A,M
16D6    1E04               MVI  E,4        ;SET REGISTER EQUAL TO FOUR
16D8    C37A16             JMP  OTNUM      ;JUMP TO OTNUM
16D8    21D310    OTPT3:   LXI  H,RHO+3    ;LOAD H L REGISTER PAIR WITH RHO+3
16DE    7E                 MOV  A,M
16DF    1E03               MVI  E,3        ;SET REGISTER E EQUAL TO THREE
16E1    C37A16             JMP  OTNUM      ;JUMP TO OTNUM
16E4    21D210    OTPT2:   LXI  H,RHO+2    ;LOAD H L REGISTER PAIR WITH RHO+2
```

```
16E7    7E                   MOV A,M
16E8    1E02                 MVI E,2      ;SET REGISTER EQUAL TO E
16EA    C37A16               JMP OTNUM    ;JUMP TO OTNUM
16ED    3AD010      SIGN:    LDA RHO      ;LOAD ACCUMULATOR WITH RHO
16F0    06F0                 MVI B,360Q   ;LOAD B WITH 360Q (CODE FOR SPACE)
16F2    B8                   CMP B        ;COMPARE B WITH ACCUMULATOR
16F3    CAD000               JZ  ZERO     ;JUMP TO ZERO
16F6    3AD901               LDA MINUS    ;LOAD ACCUMULATOR WITH MINUS
16F9    D309                 OUT 9        ;OUTPUT ACCUMULATOR TO DEVICE 9
16FB    C35C00               JMP WAIT     ;JUMP TO WAIT
1080                RWMEM:   ORG 1080H
1080                DMANT:   DS 6         ;STORAGE FOR INPUT DATA FROM DPM
1086                Y2T1:    DS 4         ;PRECURSOR FLUX FOR GROUP 1
108A                Y2T2:    DS 4         ;PRECURSOR FLUX FOR GROUP 2
108E                Y2T3:    DS 4         ;PRECURSOR FLUX FOR GROUP 3
1092                Y2T4:    DS 4         ;PRECURSOR FLUX FOR GROUP 4
1096                Y2T5:    DS 4         ;PRECURSOR FLUX FOR GROUP 5
109A                Y2T6:    DS 4         ;PRECURSOR FLUX FOR GROUP 6
109E                FLUX1:   DS 4         ;NEUTRON FLUX AT TIME T-2 * DELTA T
10A2                FLUX2:   DS 4         ;NEUTRON FLUX AT TIME T-DELTA T
10A6                FLUX3:   DS 4         ;NEUTRON FLUX AT TIME T
10AA                YT1:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 1
10AE                YT2:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 2
10B2                YT3:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 3
10B6                YT4:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 4
10BA                YT5:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 5
10BE                YT6:     DS 4         ;RELATIVE PRECURSOR FLUX FOR GROUP 6
10C2                HOLD:    DS 4         ;TEMPERARY HOLDING SPACE FOR A MULTIPLICA-
                                          ;TION
10C6                MANT:    DS 4         ;NORMALIZED MANTISSA FOR THE ELECTRICAL
                                          ;CURRENT
10CA                FLUX:    DS 4         ;MOST RECENT NEUTRON FLUX CALCULATION
```

```
 10CE            GRAYC: DS 1         ;KEITHLEY RANGE CODE
 10CF            MEM:   DS 1         ;COUNTER FOR DIRECT
 1000            RHO:   DS 13        ;FINAL ANSWER IN BCD FORMAT,THE REACTIVITY
 0100            DATA:  ORG 0100H
 0100   6100            DW LRFC      ;ADDRESS OF LRFC
 0102   4401            DW CON1      ;ADDRESS OF CON1
 0104   8500            DW HRFC      ;ADDRESS OF HRFC
```

```
8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 18
```

```
 0106   4401            DW CON1      ;ADDRESS OF CON1
 0108   8500            DW HRFC      ;ADDRESS OF HRFC
 010A   4801            DW CON2      ;ADDRESS OF CON2
 010C   6100            DW LRFC      ;ADDRESS OF LRFC
 010E   4801            DW CON2      ;ADDRESS OF CON2
 0110   8500            DW HRFC      ;ADDRESS OF HRFC
 0112   5001            DW CON4      ;ADDRESS OF CON4
 0114   6100            DW LRFC      ;ADDRESS OF LRFC
 0116   5001            DW CON4      ;ADDRESS OF CON4
 0118   6100            DW LRFC      ;ADDRESS OF LRFC
 011A   4C01            DW CON3      ;ADDRESS OF CON3
 011C   8500            DW HRFC      ;ADDRESS OF HRFC
 011E   4C01            DW CON3      ;ADDRESS OF CON3
 0120   8500            DW HRFC      ;ADDRESS OF HRFC
 0122   6001            DW CON8      ;ADDRESS OF CON8
 0124   6100            DW LRFC      ;ADDRESS OF LRFC
 0126   6001            DW CON8      ;ADDRESS OF CON8
 0128   6100            DW LRFC      ;ADDRESS OF LRFC
 012A   5C01            DW CON7      ;ADDRESS OF CON7
```

```
012C   8500           DW HRFC        ;ADDRESS OF HRFC
012E   5C01           DW CON7        ;ADDRESS OF CON7
0130   6100           DW LRFC        ;ADDRESS OF LRFC
0132   5401           DW CON5        ;ADDRESS OF CON5
0134   8500           DW HRFC        ;ADDRESS OF HRFC
0136   5401           DW CON5        ;ADDRESS OF CON5
0138   8500           DW HRFC        ;ADDRESS OF HRFC
013A   5801           DW CON6        ;ADDRESS OF CON6
013C   6100           DW LRFC        ;ADDRESS OF LRFC
013E   5801           DW CON6        ;ADDRESS OF CON6
0140   83      OFSET: DB 203Q        ;NUMBER VALUE IS 5.66
0141   35             DB 065Q
0142   1E             DB 036Q
0143   B8             DB 270Q
0144   88      CON1:  DB 210Q        ;NUMBER VALUE IS 2.5000 * 10**2
0145   7A             DB 172Q
0146   00             DB 000Q
0147   00             DB 000Q
0148   8C      CON2:  DB 214Q        ;NUMBER VALUE IS 2.5000 * 10**3
0149   1C             DB 034Q
014A   40             DB 100Q
014B   00             DB 000Q
014C   8F      CON3:  DB 217Q        ;NUMBER VALUE IS 2.5000 * 10**4
014D   43             DB 103Q
014E   50             DB 120Q
014F   00             DB 000Q
0150   92      CON4:  DB 222Q        ;NUMBER VALUE IS 2.5000 * 10**5
0151   74             DB 164Q
0152   24             DB 044Q
0153   00             DB 000Q
0154   96      CON5:  DB 226Q        ;NUMBER VALUE IS 2.5000 * 10**6
0155   18             DB 030Q
```

```
 0156   96                DB 226Q
 0157   80                DB 200Q
 0158   99        CON6:   DB 231Q        ;NUMBER VALUE IS 2.5000 * 10**7
 0159   3E                DB 076Q
 015A   BC                DB 274Q
 015B   20                DB 040Q
 015C   9C        CON7:   DB 234Q        ;NUMBER VALUE IS 2.5000 * 10**8
 015D   6E                DB 156Q
 015E   6B                DB 153Q
 015F   28                DB 050Q
 0160   A0        CON8:   DB 240Q        ;NUMBER VALUE IS 2.5000 * 10**9
 0161   15                DB 025Q
 0162   02                DB 002Q
 0163   F9                DB 371Q
 0164   7B        CONT2:  DB 173Q        ;NUMBER VALUE IS 0.02000
 0165   23                DB 043Q
 0166   D7                DB 327Q
 0167   0A                DB 012Q
 0168   7D        CONT7:  DB 175Q        ;NUMBER VALUE IS 0.07000
 0169   0F                DB 017Q
 016A   5C                DB 134Q
 016B   29                DB 051Q
 016C   80        CEX11:  DB 200Q        ;NUMBER VALUE IS 0.9949329
 016D   7E                DB 176Q
 016E   B3                DB 263Q
 016F   ED                DB 355Q
 0170    80        CEX12: DB 200Q        ;NUMBER VALUE IS 0.9874001
 0171    7C               DB 174Q
 0172    C6               DB 306Q
 0173    43               DB 103Q
 0174    80        CEX13: DB 200Q        ;NUMBER VALUE IS 0.9550420
 0175    74               DB 164Q
 0176    7D               DB 175Q
 0177    A1               DB 241Q
 0178    80        CEX14: DB 200Q        ;NUMBER VALUE IS 0.8830266
 0179    62               DB 142Q
 017A    0E               DB 016Q
 017B    0A               DB 012Q
 017C    80        CEX15: DB 200Q        ;NUMBER VALUE IS 0.5712091
 017D    12               DB 022Q
 017E    3A               DB 072Q
 017F    C2               DB 302Q
 0180    7E        CEX16: DB 176Q        ;NUMBER VALUE IS 0.2126729
 0181    59               DB 131Q
 0182    C6               DB 306Q
 0183    ED               DB 355Q
 0184    82        CEX21: DB 202Q        ;NUMBER VALUE IS 3.989853
 0185    7F               DB 177Q
 0186    59               DB 131Q
 0187    C0               DB 300Q
 0188    82        CEX22: DB 202Q        ;NUMBER VALUE IS 3.974720
```

```
8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 20
```

```
 0189    7E               DB 176Q
```

```
018A   61             DB 141Q
018B   D0             DB 320Q
018C   82     CEX23:  DB 202Q      ;NUMBER VALUE IS 3.909050
018D   7A             DB 172Q
018E   2D             DB 055Q
018F   E0             DB 340Q
0190   82     CEX24:  DB 202Q      ;NUMBER VALUE IS 3.758780
0191   70             DB 160Q
0192   8F             DB 217Q
0193   DA             DB 332Q
0194   82     CEX25:  DB 202Q      ;NUMBER VALUE IS 3.023135
0195   41             DB 101Q
0196   7B             DB 173Q
0197   0B             DB 013Q
0198   81     CEX26:  DB 201Q      ;NUMBER VALUE IS 1.844659
0199   6C             DB 154Q
019A   1D             DB 035Q
019B   CA             DB 312Q
019C   76     CLAM1:  DB 166Q      ;NUMBER VALUE IS 8.466667 * 10**-4
019D   5D             DB 135Q
019E   F2             DB 362Q
019F   CE             DB 316Q
01A0   78     CLAM2:  DB 170Q      ;NUMBER VALUE IS 2.113333 * 10**-3
01A1   0A             DB 012Q
01A2   7F             DB 177Q
01A3   D8             DB 330Q
01A4   79     CLAM3:  DB 171Q      ;NUMBER VALUE IS 7.666667 * 10**-3
01A5   7B             DB 173Q
01A6   38             DB 070Q
01A7   AA             DB 252Q
01A8   7B     CLAM4:  DB 173Q      ;NUMBER VALUE IS 2.073333 * 10**-2
01A9   29             DB 051Q
```

```
  01AA    D8              DB 330Q
  01AB    F2              DB 362Q
  01AC    7D      CLAM5:  DB 175Q       ;NUMBER VALUE IS 9.333333 * 10**-2
  01AD    3F              DB 077Q
  01AE    25              DB 045Q
  01AF    8B              DB 213Q
  01B0    7F      CLAM6:  DB 177Q       ;NUMBER VALUE IS 0.258000
  01B1    04              DB 004Q
  01B2    18              DB 030Q
  01B3    93              DB 223Q
  01B4    7C      A1:     DB 174Q       ;NUMBER VALUE IS 0.038000
  01B5    1B              DB 033Q
  01B6    A5              DB 245Q
  01B7    E3              DB 343Q
  01B8    7E      A2:     DB 176Q       ;NUMBER VALUE IS 0.2131000
  01B9    5A              DB 132Q
  01BA    36              DB 066Q
  01BB    E3              DB 343Q


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 21


  01BC    7E      A3:     DB 176Q       ;NUMBER VALUE IS 0.1880000
  01BD    40              DB 100Q
  01BE    83              DB 203Q
  01BF    12              DB 022Q
  01C0    7F      A4:     DB 177Q       ;NUMBER VALUE IS 0.4069000
  01C1    50              DB 120Q
  01C2    55              DB 125Q
  01C3    32              DB 062Q
```

```
 01C4   7E        A5:     DB 176Q        ;NUMBER VALUE IS 0.128000
 01C5   03                DB 003Q
 01C6   12                DB 022Q
 01C7   6E                DB 156Q
 01C8   7B        A6:     DB 173Q        ;NUMBER VALUE IS 0.0260000
 01C9   54                DB 124Q
 01CA   FD                DB 375Q
 01CB   F3                DB 363Q
 01CC   81        ONE:    DB 201Q        ;NUMBER VALUE IS 1.0000
 01CD   00                DB 000Q
 01CE   00                DB 000Q
 01CF   00                DB 000Q
 01D0   82        THREE:  DB 202Q        ;NUMBER VALUE IS 3.0000
 01D1   40                DB 100Q
 01D2   00                DB 000Q
 01D3   00                DB 000Q
 01D4   91        TEN5:   DB 221Q        ;NUMBER VALUE IS 1.0000 * 10**5
 01D5   43                DB 103Q
 01D6   50                DB 120Q
 01D7   00                DB 000Q
 01D8   0D        BLANK:  DB 015Q        ;CODE WORD FOR BLANKING DISPLAY
 01D9   80        MINUS:  DB 200Q        ;CODE WORD FOR MINUS SIGN
 01DA   00        PLUS:   DB 000Q        ;CODE WORD FOR PLUS SIGN
                  END
NO PROGRAM ERRORS


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 22


                        SYMBOL TABLE
```

```
* 01

A       0007       A1      01B4       A2      01B8       A3      01BC
A4      01C0       A5      01C4       A6      01C8       AD      02D7
B       0000       BLANK   01D8       C       0001       CEX11   016C
CEX12   0170       CEX13   0174       CEX14   0178       CEX15   017C
CEX16   0180       CEX21   0184       CEX22   0188       CEX23   018C
CEX24   0190       CEX25   0194       CEX26   0198       CHECK   00D8
CHK1    1693       CHS     024D       CLAM1   019C       CLAM2   01A0
CLAM3   01A4       CLAM4   01A8       CLAM5   01AC       CLAM6   01B0
CON1    0144       CON2    0148       CON3    014C       CON4    0150
CON5    0154       CON6    0158       CON7    015C       CON8    0160
CONT2   0164       CONT7   0168       D       0002       DATA    10DD *
DIRCT   00A6       DIV     02B4       DMANT   1080       E       0003
FLUX    10CA       FLUX1   109E       FLUX2   10A2       FLUX3   10A6
GRAYC   10CE       H       0004       HOLD    10C2       HRFC    0085
INITZ   0018       INP     054A       INPUT   002A *     INT     022F
INTRO   0027 *     L       0005       LOD     026E       LODE    026F
LODIR   00BC       LRFC    0061       M       0006       MANT    10C6
MEM     10CF       MINUS   01D9       MUL     028C       NEPR1   02BF
NEPR2   0292       NEUCO   0184       OFSET   0140       ONE     01CC
OTNUM   167A       OTPTO   1676       OTPT2   16E4       OTPT3   16DB
OTPT4   16D2       OTPT5   16C9       OU      060C       OUTPT   166D *
PLUS    01DA       PREC1   1400       PREC2   145A *     PREC3   14B4 *
PREC4   150E *     PREC5   1568 *     PREC6   15C2 *     PSW     0006
REACT   1628 *     RESTR   0028 *     RHO     10D0       RWMEM   16FE *
SB      02D4 *     SELCT   0050 *     SIGN    16ED       SP      0006
START   060E *     STR     023E       STRIN   024B       TEN5    01D4
THREE   01D0       TRANS   161C *     WAIT    005C       WOUT    00DF
Y2T1    1086       Y2T2    108A       Y2T3    108E       Y2T4    1092
Y2T5    1096       Y2T6    109A       YT1     10AA       YT2     10AE
```

```
YT3     10B2      YT4     10B6      YT5     10BA      YT6     10BE
ZERO    00D0
```

APPENDIX E

Intel 8080

Math Floating Point Package

The following is a reproduction of the Intel 8080 Math Floating Point Package that was developed by Otto C. Juelich.

```
8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 1


 0200                      ORG       200H
 0002            ARTHB               EQU       $ SHR 8 AND OFFH
 0200            ARITH               EQU       $
 0100            SCR   EQU           100H
 0001            SCRB                EQU       SCR SHR 8 AND OFFH
                                               ;BLANK NUMBER OF SCRATCHPAD
                 ;     8080 BINARY FLOATING POINT SYSTEM
                 ;     ARITHMETIC AND UTILITY PACKAGE
                 ;     PROGRAMMER CAL OHME
                 ;     DATE 26 DECEMBER 1973
                 ;     ARITH IS THE BEGINNING ADDRESS OF THE
                 ;     ARITHMETIC AND UTILITY PACKAGE OF THE FLOATING
                 ;     POINT SYSTEM.
                 ;     SCR IS THE BEGINNING ADDRESS OF THE
                 ;     RAM USED AS SCRATCHPAD FOR THE SYSTEM.
                 ;     THE RAM MULTIPLY AND DIVIDE SUBROUTINES
                 ;     ARE MOVED FROM ROM TO RAM BY SUBROUTINE
                 ;     INIT AND ARE EXECUTED IN RAM ONLY.
                 ;     RAM MULTIPLY SUBROUTINE.
 0100            MULX4              EQU        $-ARITH+SCR
 0200    C600          ADI          0;         ADDOPERAND 3RD FRACTION
 0001            MULP3              EQU        $-1-ARITH
 0202    5F            MOV          E,A;       4TH PARTIAL PRODUCT
 0203    7A            MOV          A,D;       3RD PARTIAL PRODUCT
 0204    CE00          ACI          0;         ADD OPERAND 2ND FRACTION
 0005            MULP2              EQU        $-1-ARITH
 0206    57            MOV          D,A;       3RD PARTIAL PRODUCT
 0207    79            MOV          A,C;       2ND PARTIAL PRODUCT
```

```
0208    CE00            ACI        0;         ADD OPERAND 1ST FRACTION
0009              MULP1            EQU        $-1-ARITH
020A    C37304          JMP        MULX5;     TO ROM CODE
                  ;     RAM DIVIDE SUBROUTINE,
0100              DIVX5            EQU        $-ARITH+SCR
020D    D600            SUI        0;         SUB DIVISOR 4TH FRACTION
000E              OP4S             EQU        $-1-ARITH
020F    7D              MOV        A,L;       REMAINDER 3RD FRACTION
0210    DE00            SBI        0;         SUB DIVISOR 3RD FRACTION
0011              OP3S             EQU        $-1-ARITH
0212    6F              MOV        L,A;       REMAINDER 3RD FRACTION
0213    7C              MOV        A,H;       REMAINDER 2ND FRACTION
0214    DE00            SBI        0;         SUB DIVISOR 2ND FRACTION
0015              OP2S             EQU        $-1-ARITH
0216    67              MOV        H,A;       REMAINDER 2ND FRACTION
0217    7B              MOV        A,E;       REMAINDER 1ST FRACTION
0218    DE00            SBI        0;         SUB DIVISOR 1ST FRACTION
0019              OP1S             EQU        $-1-ARITH
021A    5F              MOV        E,A;       REMAINDER 1ST FRACTION
021B    3E00            MVI        A,0;       REMAINDER 4TH FRACTION
001C              OP4A             EQU        $-1-ARITH
021D    C9              RET        ;          RETURN TO ROM
011E              DIVX6            EQU        $-ARITH+SCR


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 2


021E    C600            ADI        0;         ADDDIVISOR 3RD FRACTION
001F              OP3A             EQU        $-1-ARITH
0220    6F              MOV        L,A;       REMAINDER 3RD FRACTION
```

```
0221   7C            MOV       A,H;       REMAINDER 2ND FRACTION
0222   CE00          ACI       0;         ADD DIVISOR 2ND FRACTION
0023          OP2A             EQU        $-1-ARITH
0224   67            MOV       H,A;       REMAINDER 2ND FRACTION
0225   7B            MOV       A,E;       REMAINDRER 1ST FRACTION
0226   CE00          ACI       0;         ADD DIVISOR 1ST FRACTION
0027          OP1A             EQU        $-1-ARITH
0228   5F            MOV       E,A;       REMAINDER 1ST FRACTION
0229   3E00          MVI       A,0;       REMAINDER 4TH FRACTION
002A          OP4X             EQU        $-1-ARITH
022B   C3DF04        JMP       DIVX2;     TO ROM CODE
              ;      RAM LOCATIONS USED BY THE BINARY
              ;      FLOATING POINT SYSTEM
002E          OVER             EQU        $-ARITH
022E   00            DB        0;         INITIALLY CLEAR
002F          PREX             EQU        OVER+1;  PREVIOUS EXPONENT
0030          ACCE             EQU        PREX+1;  ACCUMULATOR EXPONENT
0031          ACCS             EQU        ACCE+1;  ACCUMULATOR SIGN
0032          ACC1             EQU        ACCS+1;  ACCUMULATOR 1ST FRCTN
0033          ACC2             EQU        ACC1+1;  ACCUMULATOR 2ND FRCTN
0034          ACC3             EQU        ACC2+1;  ACCUMULATOR 3RD FRCTN
0035          SF     EQU       ACC3+1;    SUBTRACTION FLAG
              ;      INT SUBROUTINE ENTRY POINT
022F   2E2F   INIT:            MVI        L,PREX; TO ADDR LAST WD TO MOVE
0231   2602   INIT1:              MVI     H,ARTHB; TO ADDRESS ROM COPY
0233   5E            MOV          E,M;    CURRENT WORD OF ROM COPY
0234   2601          MVI          H,SCRB; TO ADDRESS RAM COPY
0236   73            MOV          M,E;    WRITE CURRENT WD TO RAM
0237   2D            DCR          L;      DECREMENT WORD ADDRESS
0238   F23102        JP           INIT1;  IF MORE TO MOVE
023B   C9            RET          ;       RETURN TO CALLER
              ;      STR SUBROUTINE ENTRY POINT,
 023C    73          STR0:               MOV      M,E; STOR ZEROETH WORD
 023D    2C               INR            L;       TO ADDRESS FIRST WORD
 023E    77          STR: MOV            M,A;     STORE FIRST WORD
 023F    2C          STR1:               INR      L; TO ADDRESS SECOND WORD
 0240    70               MOV            M,B;     STORE SECOND WORD
 0241    2C               INR            L;       TO ADDRESS THIRD WORD
 0242    71               MOV            M,C;     STORE THIRD WORD
 0243    2C               INR            L;       TO ADDRESS OF FOURTH WORD
 0244    72               MOV            M,D;     STORE FOURTH WORD
 0245    C9               RET          ;           RETURN TO CALLER
                     ;FLOATING POINT ZERO SUBROUTINE ENT. PT.
 0246    2601        ZRO:  MVI           H,SCRB; TO ADDRESS SCRATCH BLAND
 0248    2E30        ZRO1:               MVI      L,ACCE; TO ADDR ACCUM EXPONENT
 024A    AF                XRA           A;       ZERO
 024B    77                MOV           M,A;     CLEAR ACCUMULATOR EXPONENT
 024C    C9                RET           ;        RETURN TO CALLER


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 3


                     ;      FLOATING POINT CHS SUBROUTINE ENT.PNT.
 024D    3E80        CHS:  MVI           A,200Q; MASK FOR SIGN BIT
 024F    0E                DB            016Q;    LBI INST TO SKIP NEXT WD
                     ;FLOATING POINT ABS SUBROUTINE ENT. PNT.
 0250    AF          ABS:  XRA           A;       ZERO
 0251    2601              MVI           H,SCRB; TO ADDRESS SCRATCH BLANK
 0253    2E31              MVI           L,ACCS;    TO ADDRESS ACCUM SIGN
 0255    A6                ANA           M        ; COMPLIMENT OF SIGN
 0256    EE80              XRI           200Q;    COMPLIMENT THE SIGN BIT
 0258    77                MOV           M,A;     ACCUMULATOR SIGN
```

```
                     ;      FLOATING POINT TEST ENTRY POINT,
0259    2601         TST:   MVI        H,SCRE; TO ADDRESS SCRATCH BLANK
025B    2E30         TST1:             MVI     L,ACCE; TO ADDR OF ACCUM
025D    7E                  MOV        A,M;    ACCUMULATOR EXPONENT
025E    A7                  ANA        A;      SET CONTROL BITS
025F    CA4602              JZ         ZRO;    IF ACCUMULATOR IS ZERO
0262    5F                  MOV        E,A;    ACCUMULATOR EXPONENT
0263    2C                  INR        L;      TO ADDR ACCUMULATOR SIGN
0264    7E                  MOV        A,M;     ACCUMULATOR SIGN
0265    2C                  INR        L;       TO ADDR ACCUM 1ST FRACTN
0266    AE                  XRA        M;       ACCUM SIGN AND 1ST FRCTN
0267    2C                  INR        L;       TO ADDR ACCUM 2ND FRCTN
0268    4E                  MOV        C,M;     ACCUMULATOR 2ND FRACTION
0269    2C                  INR        L;       TO ADDR ACCUM 3RD FRCTN
026A    56                  MOV        D,M;     ACCUMULATOR 3RD FRCTN
026B    C37A03              JMP        ADD12;   TO SET EXIT CONDITIONS
                     ;      FLOATING POINT LOAD ENTRY POINT.
026E    7E           LOD:   MOV        A,M;        OPERAND EXPONENT
026F    A7                  ANA        A;          SET CONTROL BITS
0270    CA4602              JZ         ZRO;        IF OPERAND IS ZERO
0273    5F                  MOV        F,A;        OPERAND EXPONENT
0274    2C                  INR        L;          TO ADDR OP SIGN AND 1ST
0275    7E                  MOV        A,M;        OPERAND SIGN AND 1ST FRCTN
0276    2C                  INR        L;          TO ADDRESS OPERAND 2ND FRCTN
0277    4E                  MOV        C,M;        OPERAND 2ND FRACTION
0278    2C                  INR        L;          TO ADDRESS OPERAND 3RD FRCTN
0279    56                  MOV        D,M;        OPERAND 3RD FRACTION
                     ;      STORE THE OPERAND IN THE ACCUMULATOR.
027A    6F                  MOV        L,A;        OPERAND SIGN AND 1ST FRCTN
027B    F680         LOD1:             ORI         200Q; ACCUMULATOR 1ST FRACTION
027D    47                  MOV        B,A;        ACCUMULATOR 1ST FRACTION
027E    AD                  XRA        L;           ACCUMULATOR SIGN
```

```
 027F    2601              MVI        H,SCRB;      TO ADDRESS SCRATCH BLANK
 0281    2E30              MVI        L,ACCE;      TO ADDR ACCUM EXPONENT
 0283    CD3C02            CALL       STRO;        SET THE ACCUMULATOR
 0286    A8                XRA        B;           ACCUM SIGN AND 1ST FRACTION
                   ;       SET CONTROL BITS AND EXIT
 0287    47                MOV        B,A;         ACCUM SIGN AND 1ST FRACTION
 0288    F601              ORI        1;           SET SIGN BIT FOR EXIT
 028A    7B                MOV        A,E;         ACCUMULATOR EXPONENT
 028B    C9                RET        ;            RETURN TO CALLER


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 4


                   ;       FLOATING POINT MUL SUBROUTINE ENT. PNT.
 028C    7E        MUL:    MOV        A,M;         OPERAND EXPONENT
 028D    A7                ANA        A           ; SET CONTROL BITS
 028E    C49503            CNZ        MDEX;        READ OPERAND IF NOT ZERO
 0291    CA4602            JZ         ZRO;         IF ZERO OR UNDERFLOW
 0294    DACA02            JC         OVERF;       IF OVERFLOW
 0297    CD4D04            CALL       MULX;        CALL FIXED MULT SUBRTN
                   ;       NORMALIZE IF NECESSARY.
 029A    78                MOV        A,B;         1ST PRODUCT
 029B    A7                ANA        A             ; SET CONTROL
 029C    FAA902            JM         ENDA;        IF NO NORMALIZATION REQUIRED
 029F    2E30              MVI        L,ACCE;      TO ADDR ACCUM EXPONENT
 02A1    7E                MOV        A,M;         ACCMUMLATOR EXPONENT
 02A2    DE01              SBI        1;           DECREMENT ACCUMULATOR EXPONENT
 02A4    77                MOV        M,A;         ACCUMULATOR EXPONENT
 02A5    C8                RZ         ;            RETURN TO CALLER IS UNDERFLOW
 02A6    CDBC03            CALL       LSH;         CALL LEFT SHIFT SUBROUTINE
```

```
                   ;       ROUND IF NECESSARY.
02A9    CD3004     RNDA:             CALL          ROND; CALL ROUNDING SUBROUTINE
02AC    DACA02             JC        OVERF;        IF OVERFLOW
02AF    47                 MOV       B,A;          ACCUM SIGN AND 1ST FRACTION
02B0    F601               ORI       1;            SET SIGN BIT
02B2    7B                 MOV       A,E;          ACCUMULATOR EXPONENT
02B3    C9                 RET       ;             RETURN TO CALLER
                   ;       FLOATING POINT DIV SUBROUTINE ENT. PNT.
02B4    AF         DIV:    XRA       A;            ZERO
02B5    96                 SUB       M;            COMPLEMENT OF DIVISOR EXPONENT
02B6    FE01               CPI       1;            SET CARRY IF DIVISION BY ZERO
02B8    D49503             CNC       MDEX;         READ OPERAND IF NOT ZERO
02BB    DACA02             JC        OVERF;        IF OVERFLOW OR DIVISION BY ZERO
02BE    CA4802             JZ        ZRO1;         IF UNDERFLOW OR ZERO
02C1    4F                 MOV       C,A;          DIVISOR 1ST FRACTION
02C2    CD9004             CALL      DIVX;         CALL FIXED DIV SUBRTN
02C5    2601               MVI       H,SCRB;       TO ADDRESS SCRATCH BANK
02C7    DAA902             JC        RNDA;         IF NO OVERFLOW
                   ;       SET OVERFLOW FLAG.
02CA    2601       OVERF:            MVI           H,SCRB; TO ADDRESS SCRATCH
02CC    2E2E               MVI       L,OVER;       TO ADDR OVERFLOW FLAG
02CE    3EFF               MVI       A,377Q;       OVERFLOW FLAG
02D0    77                 MOV       M,A;          OVERFLOW FLAG
02D1    07                 RLC       ;             SET CARRY BIT FOR EXIT
02D2    C9                 RET       ;             RETURN TO CALLER
02D3    00                 DB        0;            CHECK SUM WORD
                   ;       FLOATING POINT SUB SUBROUTINE ENT. PNT.
02D4    3E80       SB:     MVI       A,200Q;       MASK TO CHANGE OP SIGN
02D6    0E                 DB        016Q;         LBI INST TO SKIP NEXT WD
                   ;       FLOATING POINT ADD SUBROUTINE ENT. PNT.
02D7    AF         AD:     XRA       A;            ZERO
                   ;       LOAD THE OPERAND.
```

```
  02D8    5E                  MOV       E,M;         OPERAND EXPONENT
  02D9    2C                  INR       L;           TO ADDR OP SIGN, 1ST FRCTN


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 5


  02DA    AE                  XRA       M;           OPERNAD SIGN AND 1ST FRCTN
  02DB    47                  MOV       B,A;         OPERAND SIGN AND 1ST FRCTN
  02DC    2C                  INR       L;           TO ADDR OPERAND 2ND
  02DD    4E                  MOV       C,M;         OPERAND 2ND FRACTION
  02DE    2C                  INR       L;           TO ADDR OPERAND 3RD FRCTN
  02DF    56                  MOV       D,M;         OPERNAD 3RD FRACTION
                      ;       SAVE INITIAL EXPONENT.
  02E0    2601                MVI       H,SCRB;      TO ADDRESS SCRATCH BANK
  02E2    2E30                MVI       L,ACCE;      TO ADDR ACCUM EXPONTNT
  02E4    7E                  MOV       A,M;         ACCUMULATOR EXPONENT
  02E5    2D                  DCR       L;           TO ADDR INITIAL EXPONENT
  02E6    77                  MOV       M,A;         INITIAL EXPONENT
                      ;       CHECK FOR ZERO OPERAND.
  02E7    7B                  MOV       A,E;         OPERAND EXPONENT
  02E8    A7                  ANA       A            ; SET CONTROL BITS
  02E9    CA5B02              JZ        TST1;        IF OPERAND IS ZERO
                      ;       GENERATE SUBTRACTION FLAG, RESTORE
                      ;       SUPPRESSED FRACTION BIT.
  02EC    68                  MOV       L,B;         OPERAND SIGN AND 1ST FRCTN
  02ED    78                  MOV       A,B;         OPERAND SIGN AND 1ST FRACTION
  02EE    F680                ORI       200Q;        OPERAND 1ST FRACTION
  02F0    47                  MOV       B,A;         OPERAND 1ST FRACTION
  02F1    AD                  XRA       L;           OPERAND SIGN
  02F2    2E31                MVI       L,ACCS;      TO ADDRESS ACCUMULATOR SIGN
```

```
 02F4    AE                 XRA       M;              SUBTRACTION FLAG
 02F5    2E35               MVI       L,SF;           TO ADDRESS SUBTRACTION FLAG
 02F7    77                 MOV       M,A;            SUBTRACTION FLAG
                   ;        DETERMINE RELATIVE MAGNITUDES OF
                   ;        OPERAND AND ACCUMULATOR.
 02F8    2E30               MVI       L,ACCE;          TO ADDRESS ACCUMULATOR
                                                       ;EXPONENT
 02FA    7E                 MOV       A,M;             ACCUMULATOR EXPONENT
 02FB    A7                 ANA       A                ;SET CONTROL BITS
 02FC    CA8603             JZ        ADD17;           IF ACCUMULATOR IS ZERO
 02FF    93                 SUB       E;               DIFFERENCE IN EXPONENTS
 0300    DA0E03             JC        ADD2;            IF ACCUM SMALLER THAN OP
                   ;        CHECK FOR INSIGNIFICANT OPERAND
 0303    FA5B02             JM        TST1;            IF THE OPERAND IS INSIGNIFICANT
 0306    FE19               CPI       031Q;            COMPARE SHIFT COUNT TO 25
 0308    DA2D03             JC        ADD3;            JOIN EXCH PATH IF OP SIGNIF
 030B    C35B02             JMP       TST1;            OPERAND IS INSIGNIFICANT
                   ;        CHECHK FOR INSIGNIFICANT ACCUMULATOR
 030E    F28603    ADD2:              JP               ADD17; IF ACCUM IS INSIGNIFICANT
 0311    FEE7               CPI       347Q;            COMPARE SHIFT COUNT TO MINUS 25
 0313    DA8603             JC        ADD17;           IF ACCUM IS INSIGNIFICANT
 0316    73                MOV       M,E;              OPERAND EXPONENT
 0317    5F                 MOV       E,A;             SHIFT COUNT
 0318    2E35               MVI       L,SF;            TO ADDRESS THE SUBTRACTION
                                                             ;FLAG
 031A    7E                 MOV       A,M;             SUBTRACTION FLAG
 031B    2E31               MVI       L,ACCS;          TO ADDRESS THE ACCUMULATOR

8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 6
```

```
                                                        ;SIGN
031D    AE              XRA       M;                CPERAND SIGN
031E    77              MOV       M,A;              ACCUMULATOR SIGN
031F    AF              XRA       A;                ZERO
0320    93              SUB       E;                COMPLIMENT SHIFT COUNT
                ;       EXCHANGE ACCUMULATOR AND OPERAND
0321    2C              INR       L;                TO ADDR ACCUM 1ST FRACTION
0322    5E              MOV       E,M;              ACCUMULATOR 1ST FRACTION
0323    70              MOV       M,B;              OPERAND 1ST FRACTION
0324    43              MOV       B,E;              ACCUMULATOR 1ST FRACTION
0325    2C              INR       L;                TO ADD ACCUM 2ND FRACTION
0326    5E              MOV       E,M;              ACCUMULATOR 2ND FRACTION
0327    71              MOV       M,C;              OPERAND 2ND FRACTION
0328    4B              MOV       C,E;              ACCUMULATOR 2ND FRACTION
0329    2C              INR       L;                TO ADDR ACCUM 3RD FRACTION
032A    5E              MOV       E,M;              ACCUMULATOR 3RD FRACTION
032B    72              MOV       M,D;              OPERAND 3RD FRACTION
032C    53              MOV       D,E;              ACCUMULATOR 3RD FRACTION
                ;       POSITION THE OPERAND.
032D    CDC903  ADD3:             CALL              RSH; POSITION THE OPERAND
0330    2E35            MVI       L,SF;             TO ADDRESS SUBTRACTION FLAG
0332    7E              MOV       A,M;              SUBTACTION FLAG
0333    A7              ANA       A                 ; SET CONTROL BITS
0334    2E34            MVI       L,ACC3;           TO ADDR ACCUM 3RD FRCTN
0336    FA5003          JM        ADD9;              IF SUBRACTION REQUIRED
                ;       ADD ADDEND TO AUGEND.
0339    7E              MOV       A,M;              AUGEND 3RD FRACTION
033A    82              ADD       D;                ADDEND 3RD FRACTION
033B    57              MOV       D,A;              SUB 3RD FRACTION
033C    2D              DCR       L;                TO ADDRESS AUGEND 2ND FRACTION
033D    7E              MOV       A,M;              AUGNED 2ND FRACTION
```

```
 033E   89              ADC     C;              ADDEND 2ND FRACTION
 033F   4F              MOV     C,A;            SUB 2ND FRACTION
 0340   2D              DCR     L;              TO ADDRESS AUGEND 1ST FRACTION
 0341   7E              MOV     A,M;            AUGEND 1ST FRACTION
 0342   88              ADC     B;              ADDEND 1ST FRACTION
 0343   47              MOV     B,A;            SUB 1ST FRACTION
 0344   D27403          JNC     ADD11;      IF NO CARRY FROM 1ST FRACTION
                  ;     RIGHT SHIFT SUM TO NORMALIZED POSITION.
 0347   1F              RAR     ;               RIGHT SHIFT SUM 1ST FRACTION
 0348   47              MOV     B,A;            SUM 1ST FRACTION
 0349   79              MOV     A,C;            SUM 2ND FRACTION
 034A   1F              RAR     ;               RIGHT SHIFT SUM 2ND FRACTION
 034B   4F              MOV     C,A;            SUM 2ND FRACTION
 034C   7A              MOV     A,D;            SUM 3RD FRACTION
 034D   1F              RAR     ;               RIGHT SHIFT SUM 3RD FRACTION
 034E   57              MOV     D,A;            SUM 3RD FRACTION
 034F   1F              RAR     ;               4TH FRCTN = LOW BIT OF 3RD
 0350   5F              MOV     E,A;            SUM 4TH FRACTION
 0351   2E30            MVI     L,ACCE;         TO ADDRESS ACCUMULATOR
                                                 ; EXPONENT


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 7


 0353   7E              MOV     A,M;            ACCUMULATOR EXPONENT
 0354   C601           ADI     1;              INCREMENT ACCUMULATOR EXPONENT
 0356   DACA02          JC      OVERF;          IF OVERFLOW
 0359   77              MOV     M,A;            ACCUMULATOR EXPONENT
 035A   C37403          JMP     ADD11;          TO ROUND FRACTION
                  ;     SUBRACT SUBTRAHEND FROM MINUEND
```

```
035D   AF       ADD9:           XRA        A; MINUEND 4TH FRCTN IS ZERO
035E   93             SUB       E;         SUBTRAHEND 4TH FRACTION
035F   5F             MOV       E,A;       DIFFERENCE 4TH FRACTION
0360   7E             MOV       A,M;       MINUEND 3RD FRACTION
0361   9A             SBB       D;         SUBTRAHEND 3RD FRACTION
0362   57             MOV       D,A;       DIFFERENCE 3RD FRACTION
0363   2D             DCR       L;         TO ADDRESS MINUEND 2ND
0364   7E             MOV       A,M;       MINUEND 2ND FRACTION
0365   99             SBB       C;         SUBTRAHEND 2ND FRACTION
0366   4F             MOV       C,A;       DIFFERENCE 2ND FRACTION
0367   2D             DCR       L;         TO ADDRESS MINUEND 1ST FRCTN
0368   7E             MOV       A,M;       MINUEND 1ST FRACTION
0369   98             SBB       B;         SUBTRAHEND 1ST FRACTION
036A   47             MOV       B,A;       DIFFERENCE 1ST FRACTION
036B   DCEF03   ADD10:          CC         COMP; COMPLEMENT IF NEGATIVE
036E   F40204         CP        NORM;      NORMALIZED IF NECESSARY
0371   F24802         JP        ZRO1;      IF UNDERFLOW OR ZERO
0374   CD3004   ADD11:          CALL       ROND; CALL ROUNDING SUBROUTINE
0377   DACA02         JC        OVERF;     IF OVERFLOW
037A   47       ADD12:          MOV        B,A; ACCUM SIGN AND 1ST FRCTN
037B   2E2F          MVI        L,PREX;    TO ADDRESS PREV EXPONENT
037D   7B            MOV        A,E;       ACCUMULATOR EXPONENT
037E   96            SUB        M;         DIFFERENCE IN EXPONENTS
037F   6F            MOV        L,A;       DIFFERENCE IN EXPONENTS
0380   78            MOV        A,B;       ACCUM SIGN AND 1ST FRCTN
0381   F601          ORI        1;         SET SIGN BIT FOR EXIT
0383   7B            MOV        A,E;       ACCUMULATOR EXPONENT
0384   5D            MOV        E,L;       SIGNIFICANT INDEX
0385   C9            RET        ;          RETURN TO CALLER
                ;     LOAD ACCUMULATOR WITH THE OPERAND.
0386   2E35     ADD17:        MVI        L,SF; TO ADDR SUBTRACTION
0388   7E            MOV      A,M;       SUBTRACTION FLAG
```

```
0389    2E31            MVI         L,ACCS;         TO ADDR ACCUMULATOR SIGN
038B    AE              XRA         M;              OPERAND SIGN
038C    2D              DCR         L;              TO ADDR ACCUM EXPONENT
038D    CD3C02          CALL        STRO;           SET THE ACCUMULATOR
0390    A8              XRA         B;              ACCUM SIGN AND 1ST FRCTN
0391    C37A03          JMP         ADD12;          JOIN EXIT CODE
0394    00              DB          0;              CHECK SUM WORD
                  ;     SUBROUTINE TO READ THE OPERAND AND
                  ;     CHECK THE ACCUMULATOR EXPONENT
0395    47        MDEX:             MOV             B,A; EXPONENT MODIFIER
0396    2C              INR         L;              TO ADDR OP SIGN, 1ST FRCTN
0397    4E              MOV         C,M;            OPERAND SIGN AND 1ST FRACTION
0398    2C              INR         L;              TO ADDRESS OPERAND 2ND FRCTN
```

```
0399    56              MOV         D,M;            OPERAND 2ND FRACTION
039A    2C              INR         L;              TO ADDRESS OPERAND 3RD FRCTN
039B    5E              MOV         E,M;            OPERAND 3RD FRACTION
039C    2601            MVI         H,SCRB;         TO ADDRESS SCRATCH BANK
039E    2E30            MVI         L,ACCE;         TO ADDRESS ACCUMULATOR
                                                     ; EXPONENT
03A0    7E              MOV         A,M;            ACCUMULATOR EXPONENT
03A1    A7              ANA         A               ; SET CONTROL BITS
03A2    C8              RZ          ;               RETURN IF ACCUM IS ZERO
03A3    80              ADD         B;              RESULT EXPONENT PLUS BIAS
03A4    47              MOV         B,A;            RESULT EXPONENT PLUS BIAS
03A5    1F              RAR         ;               CARRY TO SIGN
03A6    A8              XRA         B;              CARRY AND SIGN MUST DIFFER
```

```
03A7    78               MOV       A,B;          RESULT EXPONENT PLUS BIAS
03A8    0680             MVI       B,200Q;       EXP BIAS, SIGN MASK, MS BIT
03AA    F2B803           JP        OVUN;         IF OVERFLOW OR UNDERFLOW
03AD    90               SUB       B;            REMOVE EXCESSS EXP BIAS
03AE    C8               RZ        ;             RETURN IF UNDERFLOW
03AF    77               MOV       M,A;          RESULT EXPONENT
03B0    2C               INR       L;            TO ADDRESS ACCUMULATOR SIGN
03B1    7E               MOV       A,M;          ACCUMULATOR SIGN
03B2    A9               XRA       C;            RESULT SIGN IN SIGN BIT
03B3    A0               ANA       B;            RESULT SIGN
03B4    77               MOV       M,A;          RESULT SIGN
03B5    79               MOV       A,C;          OPERAND SIGN AND 1ST FRCTN
03B6    B0               ORA       B;            OPERAND 1ST FRACTION
03B7    C9               RET       ;             RETURN TO CALLER
03B8    07          OVUN:RLC;                    SET CARRY BIT IF OVERFLOW
03B9    D8               RC        ;             RETURN IF OVERFLOW
03BA    AF               XRA       A;            ZERO
03BB    C9               RET       ;             RETURN IF UNDERFLOW
                    ;    SUBROUTINE TO LEFT SHIFT THE B,C,
                    ;    D, AND E REGISTERS ONE BIT,
03BC    7B          LSH: MOV       A,E;          ORIGINAL CONTENTS OF E
03BD    17               RAL       ;             LEFT SHIFT E
03BE    5F               MOV       E,A;          RESTORE CONTENTS OF REGISTER E
03BF    7A          LSH1:          MOV           A,D; ORGINAL CONTENTSS OF D
                                                 ; REGISTER
03C0    17               RAL       ;             LEFT SHIFT D
03C1    57               MOV       D,A;          RESTORE CONTENTS OF D REGISTER
03C2    79               MOV       A,C;          ORIGINAL CONTENTS OF C
                                                 ; REGISTER
03C3    17               RAL       ;             LEFT SHIFT C
03C4    4F               MOV       C,A;          RESTORE CONTENTS OF C REGISTER
03C5    78               MOV       A,B;          ORIGINAL CONTENTS OF B
```

```
                                                  ;REGISTER
 03C6   8F              ADC        A;             LEFT SHIFT B
 03C7   47              MOV        B,A;           RESTORE CONTENTS OF B REGISTER
 03C8   C9              RET        ;              RETURN TO CALLER
                     ;   RIGHT SHIFT THE B, C, D, AND E REGISTERS
                     ;   BY THE SHIFT COUNT IN THE A REGISTER


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 9


                     ;   SHIFT OPERAND TO REGISTER INDICATED BY
                     ;   SHIFT COUNT
 03C9   1E00         RSH: MVI      E,0;           OPERAND 4TH FRCTN IS ZERO
 03CB   2E08         RSH0:         MVI            L,010Q; EACH REG IS 8 BITS OF
                                                     ;SHIFT
 03CD   BD           RSH1:         CMP            L; COMPARE SHIFT COUNT TO 8
 03CE   FADA03          JM         RSH2;          IF REQ SHIFT LESS THAN 8
 03D1   5A              MOV        E,D;           OPERAND 4TH FRACTION
 03D2   51              MOV        D,C;           OPERAND 3RD FRACTION
 03D3   48              MOV        C,B;           OPERAND 2ND FRACTION
 03D4   0600            MVI        B,0;           OPERAND 1ST FRACTION IS ZERO
 03D6   95              SUB        L;;            REDUCE SHIFT COUNT BY 1 REG
 03D7   C2CD03          JNZ        RSH1;          IF MORE SHIFTS REQUIRED
                     ;  SHIFT OPERAND RIGHT BY - SHIFT COUNT-
                     ;  BITS.
 03DA   A7           RSH2:         ANA            A; SET CONTROL BITS
 03DB   C8              RZ         ;              RETURN  IF SHIFT IS COMPLETE
 03DC   6F              MOV        L,A;           SHIFT COUNT
 03DD   A7           RSH3:         ANA            A; CLEAR CARRY BIT
 03DE   78               MOV       A,B;           OPERAND 1ST FRACTION
```

```
03DF    1F              RAR         ;             RIGHT SHIFT OP 1ST FRCTN
03E0    47              MOV         B,A;          OPERAND 1ST FRACTION
03E1    79              MOV         A,C;          OPERAND 2ND FRACTION
03E2    1F              RAR         ;             RIGHT SHIFT OP 2ND FRACTION
03E3    4F              MOV         C,A;          OPERAND 2ND FRACTION
03E4    7A              MOV         A,D;          OPERAND 3RD FRACTION
03E5    1F              RAR         ;             RIGHT SHIFT OP 3RD FRACTION
03E6    57              MOV         D,A;          OPERAND 3RD FRACTION
03E7    7B              MOV         A,E;          OPERAND 4TH FRACTION
03E8    1F              RAR         ;             RIGHT SHIFT OP 4TH FRACTION
03E9    5F              MOV         E,A;          OPERAND 4TH FRACTION
03EA    2D              DCR         L;            DECREMENT SHIFT COUNT
03EB    C2DD03          JNZ         RSH3;         IF MORE SHIFTS REQUIRED
03EE    C9              RET
                ;       COMPLEMENT THE B,C,D, AND E REGISTERS,
03EF    2D      COMP:               DCR           L; TO ADDR ACCUM SIGN
03F0    7E              MOV         A,M;          ACCUMUALATOR SIGN
03F1    EE80            XRI         200Q;         CHANGE SIGN
03F3    77              MOV         M,A;          ACCUMULATOR SIGN
03F4    AF      COMP1:              XRA           A; ZERO
03F5    6F              MOV         L,A;          ZERO
03F6    93              SUB         E;            COMPLIMENT 4TH FRACTION
03F7    5F              MOV         E,A;          4TH FRACTION
03F8    7D              MOV         A,L;          ZERO
03F9    9A              SBB         D;            COMPLIMENT 3RD FRACTION
03FA    57              MOV         D,A;          3RD FRACTION
03FB    7D              MOV         A,L;          ZERO
03FC    99              SBB         C;            COMPLIMENT 2ND FRCTN
03FD    4F              MOV         C,A;          2ND FRACTION
03FE    7D              MOV         A,L;          ZERO
03FF    98              SBB         B;            COMPLIMENT 1ST FRCTN
```

```
 0400   47              MOV       B,A;          1ST FRACTION
 0401   C9              RET       ;             RETURN TO CALLER
                  ;     NORMALIZED THE REGISTERS.
 0402   2E20      NORM:MVI        L,040Q;       MAX NORMALIZING SHIFT
 0404   78        NORM1:          MOV           A,B; 1ST FRACTION
 0405   A7              ANA       A             ; SET CONTROL BITS
 0406   C22204          JNZ       NORM3;        IF 1ST FRACTION NONZERO
 0409   41              MOV       B,C;          1ST FRACTION
 040A   4A              MOV       C,D;
 040B   53              MOV       D,E;          3RD FRACTION
 040C   5F              MOV       E,A;          ZERO 4TH FRACTION
 040D   7D              MOV       A,L;          NORMALIZING SHIFT COUNT
 040E   D608            SUI       010Q;         REDUCE SHIFT COUNT
 0410   6F              MOV       L,A;          NORMALIZING SHIFT COUNT
 0411   C20404          JNZ       NORM1;        IF FRACTION NONZERO
 0414   C9              RET       ;             IF FRACTION IS ZERO
 0415   2D        NORM2:          DCR           L; DECREMENT SHIFT COUNT
 0416   7B              MOV       A,E;          ORIGINAL CONTENTS OF E
 0417   17              RAL       ;             LEFT SHIFT E
 0418   5F              MOV       E,A;          RESTORE CONTENTS OF E REGISTER
 0419   7A              MOV       A,D;          ORIGINAL CONTENTS OF D
 041A   17              RAL       ;             LEFT SHIFT D
 041B   57              MOV       D,A;          RESTORE CONTENTS OF D REGISTER
 041C   79              MOV        A,C;         ORIGINAL CONTENTS OF C
 041D   17              RAL        ;            LEFT SHIFT C
 041E   4F              MOV        C,A;         RESTORE CONTENTS OF C REGISTER
 041F   78              MOV        A,B;         ORIGINAL CONTENTS OF B
 0420    8F              ADC          A;          LEFT SHIFT B
 0421    47              MOV          B,A;        RESTORE CONTENTS OF B REGISTER
 0422    F21504    NORM3:             JP          NORM2; IF NOT NORMALIZED
 0425    7D              MOV          A,L;        NORMALIZING SHIFT COUNT
 0426    D620            SUI          040Q;       REMOVE BIAS
 0428    2E30            MVI          L,ACCE;     TO ADDR ACCUM EXPONENT
 042A    86              ADD          M;          ADJUST ACCUM EXPONENT
 042B    77              MOV          M,A;        NEW ACCUM EXPONENT
 042C    C8              RZ           ;           RETURN IF ZERO EXP
 042D    1F              RAR          ;           BORROW BIT TO SIGN
 042E    A7              ANA          A;          SET SIGN TO IND, UNDERFLOW
 042F    C9              RET          ;           RETURN TO CALLER
                   ;     SUBROUTINE TO ROUND THE B,C, D REGISTERS.
 0430    2E30      ROND:              MVI         L,ACCE; TO ADDR ACCUM EXPONENT
 0432    7B              MOV          A,E;        4TH FRACTION
 0433    A7              ANA          A           ; SET CONTROL BITS
 0434    5E              MOV          E,M;        ACCUMULATOR EXPONENT
 0435    FC3F04          CM           RNDR;       CALL 2ND LEVEL ROUNDER
 0438    D8              RC           ;           IF OVERFLOW
 0439    78              MOV          A,B;        1ST FRACTION
 043A    2C              INR          L;          TO ADDR ACCUM SIGN
 043B    AE              XRA          M;          ACCUM SIGN AND 1ST FRCTN
 043C    C33F02          JMP          STR1; RETURNG THRU STORE SUBR.
                   ;     SECOND LEVEL ROUNDING SUBROUTINE.


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 11


 043F    14        RNDR:              INR         D; ROUND 3RD FRACTION
 0440    C0              RNZ          ;           RETURN IF NO CARRY
```

```
0441   0C          INR       C;            CARRY TO 2ND FRACTION
0442   C0          RNZ       ;             RETURN IF NO CARRY
0443   04          INR       B;            CARRY TO 1ST FRACTION
0444   C0          RNZ       ;             RETURN IF NO CARRY
0445   7B          MOV       A,E;          ACCUMULATOR EXPONENT
0446   C601        ADI       1;            INCREMENT ACCUM EXPONENT
0448   5F          MOV       E,A;          NEW ACCUMULATOR EXPONENT
0449   0680        MVI       B,200Q;       NEW 1ST FRACTION
044B   77          MOV       M,A;          NEW ACCUM EXPONENT
044C   C9          RET       ;             RETURN TO ROND SUBROUTINE
                ;  FIXED POINT MULTIPLY SUBROUTINE.
044D   2E09     MULX:        MVI           L,MULP1; TO ADDR 1ST
                                           ; MULTIPLICAND
044F   77          MOV       M,A;          1ST MULTIPLICAND
0450   2E05        MVI       L,MULP2;      TO ADDR 2ND MULTIPLICAND
0452   72          MOV       M,D;          2ND MULTIPLICAND
0453   2E01        MVI       L,MULP3;      TO ADDR 3RD MULTIPLICAND
0455   73          MOV       M,E;          3RD MULTIPLICAND
0456   AF          XRA       A;            CLEAR 6TH PRODUCT
0457   5F          MOV       E,A;          CLEAR 5TH PRODUCT
0458   57          MOV       D,A;          CLEAR 4TH PRODUCT
                ;  MULTIPLY BY EACH ACCUMULATOR
                ;  FRACTION IN TURN.
0459   2E34        MVI       L,ACC3;       TO ADDRESS 3RD FRCTN
045B   CD6804      CALL      MULX2;        MULTIPLY BY ACCUM 3RD FRCTN
045E   2E33        MVI      L,ACC2;        TO ADDRESS 2ND FRCTN
0460   CD6504      CALL      MULX1;      MULTIPLY BY ACCUM 2ND FRCTN
0463   2E32        MVI      L,ACC1;    TO ADDRESS 1ST FRCTN
                ;  MULTILPY BY ONE ACCUMULATOR WORD.
0465   7A       MULX1:      MOV            A,D; 5TH PARTIAL PRODUCT
0466   59          MOV      E,C;            4TH PARTIAL PRODUCT
0467   50          MOV      D,B;            3RD PARTIAL PRODUCT
```

```
0468    46          MULX2:          MOV             B,M; MULTIPLIER
0469    6F                  MOV     L,A;            5TH PARTIAL PRODUCT
046A    AF                  XRA     A;              ZERO
046B    4F                  MOV     C,A;            2ND PARTIAL PRODUCT
046C    90                  SUB     B;              SET CARRY BIT FOR EXIT FLAG
046D    DA7904              JC      MULX3;          IF MULTIPLIER IS NOT ZERO
0470    4A                  MOV     C,D;            2ND PARTIAL PRODUCT
0471    53                  MOV     D,E;            3RD PARTIAL PRODUCT
0472    C9                  RET     ;               MULT BY ZERO COMPLETE
                    ;       COMPLETE ADDITION OF MULTIPLICAND
0473    4F          MULX5:          MOV            C,A; 2ND PARTIAL PRODUCT
0474    D27904              JNC     MULX3;         IF NO CARRY TO 1ST PRODUCT
0477    04                  INR     B;             ADD CARRY TO 1ST PRODUCT
0478    A7                  ANA     A;             CLEAR CARRY BIT
                    ;       LOOP FOR EACH BIT OF MULTIPLIER WORD.
0479    7D          MULX3:          MOV            A,L; 5TH PART PRODUCT, EXIT
                                                       ; FLAG
```

```
8C80 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 12


047A    8F                  ADC     A;             SHIFT EXIT FLAG OUT IF DONE
047B    C8                  RZ      ;              EXIT IF MULTIPLICATION DONE
047C    6F                  MOV     L,A;           5TH PART PRODUCT, EXIT FLAG
047D    7B                  MOV     A,E;            4TH PARTIAL PRODUCT
047E    17                  RAL     ;              SHIFT 4TH PARTILA PRODUCT
047F    5F                  MOV     E,A:           4TH PARTIAL PRODUCT
0480    7A                  MOV     A,D;           3RD PARTIAL PRODUCT
0481    17                  RAL     ;              SHIFT 3RD PARTIAL PRODUCT
0482    57                  MOV     D,A;           3RD PARTIAL PRODUCT
```

```
0483   79              MOV        A,C;          2ND PARTIAL PRODUCT
0484   17              RAL         ;            SHIFT 2ND PARTIAL PRODUCT
0485   4F              MOV        C,A;          2ND PARTIAL PRODUCT
0486   78              MOV        A,B;          1ST PARTIAL PRODUC AND MULTIPLIFR
0487   17              RAL         ;            SHIFT 1ST PROOD AND MULTIPLIER
0488   47              MOV        B,A;          1ST PART PROD AND MULTIPLIER
0489   D27904         JNC       MULX3;          IF NO ADDITION REQUIRED
                ;      ADD THE MULTIPLICAND TO THE PRODUCT
                ;     IF THE MULTIPLIER BIT IS ONE.
048C   7B             MOV         A,E;          4TH PARTIAL PRODUCT
048D   C30001         JMP         MULX4;        TO RAM CODE
                ;     FIXED POINT DIVIDE SUBROUTINE.
                ;     SUBTRACT DIVISOR FROM ACCUMULATOR TO
                ;     OBTAIN 1ST REMAINDER.
0490   2E34     DIVX:             MVI        L,ACC3; TO ADDRESS ACCUM 3RD
0492   7E             MOV         A,M;       ACCUMULATOR 3RD FRACTION
0493   93             SUB         E;         DIVISOR 3RD FRACTION
0494   73             MOV         M,E;       REMAINDER 3RD FRACTION
0495   2D             DCR         L;         TO ADDRESS ACCUM 2ND FRACTION
0496   7E             MOV         A,M;       ACCUMULATOR 2ND FRACTION
0497   9A             SBB         D;         DIVISOR 2ND FRACTION
0498   77             MOV         M,A;       REMAINDER 2ND FRACTION
0499   2D             DCR         L;         TO ADDRESS ACCUM 1ST FRCTN
049A   7E             MOV         A,M;       ACCUMULATOR 1ST FRCTN
049B   99             SBB         C;         DIVISOR 1ST FRACTION
049C   77             MOV         M,A;       REMAINDER 1ST FRACTION
                ;     HALVE THE DIVISOR AND STORE FOR
                ;     ADDITION OR SUBTRACTION
049D   79             MOV         A,C;       DIVISOR 1ST FRACTION
049E   17             RAL          ;         SET CARRY BIT
049F   79             MOV         A,C;       DIVISOR 1ST FRACTION
04A0   1F             RAR         ;          HALF OF DIVISOR 1ST FRACTION
```

```
                     ;             + 2000 TO CORRECT QUOTIENT
   04A1    2E19           MVI      L,OP1S;   TO ADDRESS 1ST SUBTRACT
                                             ; DIVISOR
   04A3    77             MOV      M,A;      1ST SUBTRACT DIVISOR
   04A4    2E27           MVI      L,OP1A;   TO ADDRESS 1ST ADD DIVISOR
   04A6    77             MOV      M,A;      1ST ADD DIVISOR
   04A7    7A             MOV      A,D;      DIVISOR 2ND FRACTION
   04A8    1F             RAR      ;         HALF OF DIVISOR 2ND FRACTION
   04A9    2E15           MVI      L,OP2S;   TO ADDRESS 2ND SUBTRACT
                                              ;DIVISOR
```

```
8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 13
```

```
   04AB    77             MOV      M,A;      2ND SUBTRACT DIVISOR
   04AC    2E23           MVI      L,OP2A;   TO ADDRESS 2ND ADD DIVISOR
   04AE    77             MOV      M,A;      2ND ADD DIVISOR
   04AF    7B             MOV      A,E;      DIVISOR 3RD FRACTION
   04B0    1F             RAR      ;         HALF OF DIVISOR 3RD FRACTION
   04B1    2E11           MVI     L,OP3S;    TO ADDRESS 3RD SUBTRACT
                                             ; DIVISOR
   04B3    77             MOV     M,A;        3RD SUBTRACT DIVISOR
   04B4    2E1F           MVI     L,OP3A;     TO ADDRESS 3RD ADD DIVISOR
   04B6    77             MOV     M,A;        3RD ADD DIVISOR
   04B7    0600           MVI     B,0;        INIT QUOTIENT 1ST FRCTN
   04B9    78             MOV     A,B;        DIVISOR FOURTH FRACTION IS
                                               ;ZERO
   04BA    1F             RAR       ;         LOW BIT OF DIVISOR 3RD
                                               ;FRACTION
   04BB    2E0E           MVI     L,OP4S;     TO ADDRESS 4TH SUBTRACT
```

```
                                                     ; DIVISOR
04BD    77          MOV        M,A;          4TH SUBTRACT DIVISOR
04BE    2E1C        MVI        L,OP4A;       TO ADDRESS 4TH ADD DIVISOR
04C0    77          MOV        M,A;          4TH ADD DIVISOR
04C1    2E2A        MVI        L,OP4X;       TO ADDRESS 4TH ADD DISIOR
04C3    77          MOV        M,A;          4TH ADD DIVISOR
                ;   LOAD 1ST REMAINDER, CHECK SIGN
04C4    2E32        MVI        L,ACC1;         TO ADDR REMAINDER 1ST FRCTN
04C6    7E          MOV        A,M;             REMAINDER 1ST FRACTION
04C7    2C          INR        L;               TO ADDR REMAINDER 2ND FRCTN
04C8    56          MOV        D,M;             REMAINDER 2ND FRACTION
04C9    2C          INR        L;               TO ADDR REMAINDER 3RD FRCTN
04CA    5E          MOV        E,M;             REMAINDER 3RD FRACTION
04CB    A7          ANA        A;               SET CONTROL BITS
04CC    FAF604      JM         DIVX4;           IF REMAINDER IS NEGATIVE
                ;    ADJUST EXPONENT, POSITION REMAINDER
                ;   AND INITIALIZE THE QUOTIENT,
04CF    2E30        MVI        L,ACCE;          TO ADDRESS ACCUMULATOR
                                                ; EXPONENT
04D1    4E          MOV        C,M;             QUOTIENT EXPONENT
04D2    0C          INR        C;               INCREMENT QUOTIENT EXPONENT
04D3    C8          RZ         ;                RETURN IF OVERFLOW
04D4    71          MOV        M,C;             QUOTIENT EXPONENT
04D5    6B          MOV        L,E;             REMAINDER 3RD FRACTION
04D6    62          MOV        H,D;             REMAINDER 2ND FRACTION
04D7    5F          MOV        E,A;             REMAINDER 1ST FRACTION
04D8    1601        MVI        D,1;             INITIALIZE QUOT 3RD FRCTN
04DA    48          MOV        C,B;             INITIALIZE QUOT 2ND FRCTN
                ;   SUBTRACT THE DIVISOR FROM THE REMAINDER
                ;   IF IT IS POSITIVE.
04DB    AF      DIVX1:         XRA              A; REMAINDER 4TH FRCTN IS ZERO
04DC    CD0001      CALL       DIVX5;           CALL RAM SECTION
```

```
 04DF    07        DIVX2:        RLC                ; SHIFT REM 4TH FRCTN TO CY
                   ;      SHIFT THE REMAINDER LEFT ONE BIT
 04E0    78               MOV        A,B;          QUOTIENT 1ST FRACTION
```

```
 04E1    17               RAL        ;             MS BIT OF QUOTIENT TO CY
 04E2    D8               RC         ;             IF DIVISION COMPLETE
 04E3    1F               RAR        ;             REMAINDER 4TH FRCTN TO CY
 04E4    7D               MOV        A,L;          REMAINDER 3RD FRACTION
 04E5    17               RAL        ;             LEFT SHIFT REM 3RD FRACTION
 04E6    6F               MOV        L,A;          REMAINDER 3RD FRACTION
 04E7    7C               MOV        A,H;          REMAINDER 2ND FRACTION
 04E8    17               RAL         ;            LEFT SHIFT REM 2ND FRCTN
 04E9    67               MOV        H,A;          REMAINDER 2ND FRACTION
 04EA    CD8C03           CALL     LSH;            CALL LEFT SHIFT SUBROUTINE
                   ;      BRANCH IF SUBTRACTION IS REQUIRED
 04ED    7A               MOV        A,D;          QUOTIENT 3RD FRACTION
 04EE    0F               RRC        ;             REM SIGN INDIC TO CARRY BIT
 04EF    DADB04           JC         DIVX1;        TO SUB DIVISOR IF REM POS
                   ;      ADD THE DIVISOR IF THE REMAINDER
                   ;      IS NEGATIVE.
 04F2    7D        DIVX3:        MOV                A,L; REMAINDER 3RD FRACTION
 04F3    C31D03           JMP        DIVX6;        TO RAM CODE
                   ;      POSITION THE REMAINDER AND INITIALIZE
                   ;      THE QUOTIENT.
 04F6    6B        DIVX4:        MOV               L,E; REMAINDER 3RD FRACTION
 04F7    62             MOV          H,D;         REMAINDER 2ND FRACTION
 04F8    5F             MOV          E,A;         REMAINDER 1ST FRACTION
```

```
04F9   50          MOV      D,B;          INITIALIZE QUOT 3RD FRCTN
04FA   48          MOV      C,B;          INITIALIZE QUOT 2ND FRCTN
04FB   C3F204      JMP      DIVX3;        ADD DIVISOR IF REM IS NEG
04FE   00          DB       0;            CHECKSUM WORD
                ; SCR EQU      100H
                ; SCRB         EQU           SCR SHR 8 AND 0FFH
                ;                           ;BANK NUMBER OF SCRATCH PAD
                ; ARITH        EQU          200H;  BASE ADDRESS OF
                                             ;ARITHEMATIC PACKAGE
                ;      8080 BINARY FLOATING POINT SYSTEM
                ;      FORMAT CONVERSION PACKAGE
                ;      PROGRAMMER CAL OHME
                ;      DATE 26 DECEMBER 1973
                ;      ARITH IS THE BEGINNING ADDRESS OF THE
                ;      ARITHEMATIC AND UTILITY PACKAGE OF THE FLOATING
                ;      POINT SYSTEM
                ;      SCR IS THE BEGINNING ADDRESS OF THE
                ;      RAM USED AS SCRATCHPAD FOR THE SYSTEM.
                ;      RAM LOCATION USED BY THE BINARY
                ;      FLOATING POINT SYSTEM.
                ;OVER           EQU      56Q; OVERFLOW FLAG
                ;ACCE           EQU      60Q; ACCUMULATOR EXPONENT
                ;ACCS           EQU      ACCE+1; ACCUMULATOR SIGN
                ;ACC1           EQU      ACCS+1; ACCUMULATOR 1ST FRCTN
0033            ACC2            EQU      ACC1+1; SCCUMULATOR 2ND FRTCN
0034            ACC3            EQU      ACC2+1; ACCUMULATOR 3RD FRCTN
                ;SF      EQU        ACC3+1; SUBTRACTION FLAG
0036            ADRL          EQU      SF+1; CHARACTER STRING WORD


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 15
```

```
0037          ADRH          EQU       ADRL+1; TEMPORARY STORAGE
0038          TMP1          EQU       ADRH+1; TEMPORARY STORAGE
0039          TMP2          EQU       TMP1+1; TEMPORARY STORAGE
003A          TMP3          EQU       TMP2+1; VALUE EXPONENT
003B          VALE          EQU       TMP3+1; VALUE EXPONENT
003C          VAL1          EQU       VALE+1;  VALUE 1ST FRACTION
003D          VAL2          EQU       VAL1+1;  VALUE 2ND FRACTION
003E          VAL3          EQU       VAL2+1;  VALUE 3RD FRACTION
003F          TMP4          EQU       VAL3+1; TEMPORARY STORAGE
              ;   ADDRESS IN THE ARITHMETIC AND UTILITY
              ;   PACKAGE REFERENCE BY THE FORMAT CONVERSION
              ;   PACKAGE.
              ; STR   EQU       ARITH+76Q
              ; ZRO   EQU       ARITH+106Q
              ; ABS   EQU       ARITH+120Q
              ; TST   EQU       ARITH+131Q
              ; LCD   EQU       ARITH+156Q
              ; MUL   EQU       ARITH+214Q
              ; DIV   EQU       ARITH+264Q
              ; AD    EQU       ARITH+327Q
              ; ADD10           EQU       ARITH+553Q
              ; LSH   EQU     ARITH+674Q
              ; RSH   EQU       ARITH+711Q
              ; COMP1           EQU       ARITH+764Q
              ;       SUBROUTINE TO CONVERT FROM FIXED
              ;       POINT TO FLOATING POINT FORMAT.
              ;       ARBITRARY BCD CHARACTER CODES, (OFFSET 30H
                                      ; FROM ASCII):
              ;           SPACE - 0F0H    DECIMAL POINT - 0FEH
              ;           PLUS  - 0FBH    LETTER E      - 015H
```

```
                  ;            MINUS - 0FDH
04FF   6B         FLT: MOV        L,E;        INPUT EXPONENT
0500   5A              MOV        E,D;        4TH INPUT
0501   51              MOV        D,C;        3RD INPUT
0502   48              MOV        C,B;        2ND INPUT FRACTION
0503   47              MOV        B,A;        1ST INPUT FRACTION
0504   7D              MOV        A,L;        INPUT EXPONENT
0505   EE80            XRI        200Q;       APPLY EXPONENT BIAS
0507   2601            MVI        H,SCRB;     TO ADDRESS SCRATCH BANK
0509   2E30            MVI        L,ACCE;     TO ADDRESS ACCUM EXPONENT
050B   77              MOV        M,A;        ACCUMULATOR EXPONENT
050C   2C              INR        L;          TO ADDRESS ACCUM SIGN
050D   3680            MVI        M,200Q;     SET ACCUM SIGN POSITIVE
050F   2C              INR        L;          TO ADDR ACCUM 1ST FRCTN
0510   78              MOV        A,B;        1ST INPUT FRACTION
0511   A7              ANA        A           ; SET SIGN BIT
0512   17              RAL        ;           INPUT SIGN TO CARRY
0513   C36B03          JMP        ADD10;      COMPLETE CONVERSION
                  ;    SUBROUTINE TO CONVER FROM FLOATING
                  ;    POINT TO FIXED POINT FORMAT.
0516   2601       FIX: MVI        H,SCRB;     TO ADDRESS SCRATCH BANK
```

```
0518   2E30           MVI         L,ACCE;     TO ADDR ACCUM EXPONENT
051A   7E              MOV        A,M;        ACCUMULATOR EXPONENT
051B   A7              ANA        A           ; SET CONTROL BITS
051C   CA4405          JZ         FIX1;       IF ACCUMULATOR IS ZERO
051F   7B              MOV        A,E;        INPUT EXPONENT
```

```
0520   C67F              ADI      177Q;          APPLY BIAS - 1
0522   96                SUB      M;             SHIFT COUNT -1
0523   D8                RC       ;              RETURN IF ACCUM TOO LARGE
0524   FE1F              CPI      037Q;          COMPARE TO LARGE SHIFT
0526   D24405            JNC      FIX1;          IF ACCUMULATOR TOO SMALL
0529   C601              ADI      1;             SHIFT COUNT
052B   2E32              MVI      L,ACC1;        TO ADDR ACCUM 1ST FRCTN
052D   46                MOV      B,M;           ;ACCUMULATOR 1ST FRACTION
052E   2C                INR      L;             TO ADDR ACCUM 2ND FRCTN
052F   4E                MOV      C,M;           ACCUMULATOR 2ND FRCTN
0530   2C                INR      L;             TO ADDR ACCUM 3RD FRCTN
0531   56                MOV     D,M;            ACCUMULATOR 3RD FRACTION
0532   CDC903            CALL    RSH;            POSITION THE FRACTION
0535   2E31              MVI     L,ACCS;         TO ADDR ACCUM SIGN
0537   7E                MOV     A,M;            ACCUMULATOR SIGN
0538   A7                ANA     A               ; SET CONTROL BITS
0539   F4F403            CP      COMP1;          COMPEMENT FRCTN IF NEG
053C   3E01              MVI     A,1;            NON-ZERO
053E   B0                ORA     B;              SET CONTROL BITS FOR EXIT
053F   78                MOV     A,B;            1ST RESULT
0540   41                MOV     B,C;            2ND RESULT
0541   4A                MOV     C,D;            3RD RESULT
0542   53                MOV     D,E;            4TH RESULT
0543   C9                RET     ;               RETURN TO CALLER
0544   AF          FIX1:         XRA             A; ZERO
0545   47                MOV     B,A;            ZERO
0546   4F                MOV     C,A;            ZERO
0547   57                MOV     D,A;            ZERO
0548   C9                RET     ;               RETURN TO CALLER
0549   00                DB      0;              CHECKSUM WORD
                   ;       INP SUBROUTINE ENTRY POINT
                   ;       INITIALIZE TEMPORARY STORAGE
```

```
 054A    5E          INP: MOV       E,M;           FIRST CHARACTER OF STRING
 054B    CDD406           CALL      SVAD;       SET CHAR ADDR, PNT FLG, EXP
 054E    2C               INR       L;             TO ADDRESS VALUE SIGN
 054F    3680             MVI       M,200Q;        SET VALUE SIGN POSITIVE
 0551    2E30             MVI       L,ACCE;        TO ADDR ACCUM EXPONENT
 0553    72               MOV       M,D;           SET ACCUM TO ZERO
 0554    7B               MOV       A,E;           FIRST CHARACTER
 0555    FEF0             CPI       360Q;          COMPARE TO SPACE
 0557    CA6705           JZ        INP1;          IF SPACE CHARACTER
 055A    FEFB             CPI       373Q;          COMPARE CHAR TO PLUS
 055C    CA6705           JZ        INP1;          IF PLUS SIGN
 055F    FEFD             CPI       375Q;          COMPARE TO MINUS
 0561    C26D05           JNZ       INP2;          IF NOT MINUS SIGN
 0564    2E3A             MVI       L,TMP3;        TO ADDR VALUE SIGN
```

```
 0566    72               MOV       M,D;           SET VALUE OF SIGN NEGATIVE
                 ;        ANALYZE NEXT CHARACTER IN STRING
 0567    CDE106      INP1:          CALL           CHAD; CALL CHAR ADDR SBRTN
 056A    7E               MOV       A,M;           NEXT CHARACTER
 056B    2601             MVI       H,SCRB;        TO ADDRESS SCRATCH BANK
 056D    0600        INP2:          MVI            B,0; DIGIT 2ND WD OR DEC EXP
 056F    FEFE             CPI       376Q;          COMPARE TO DECIMAL POINT
 0571    CAAA05           JZ        INP3;          IF DECIMAL POINT
 0574    FE15             CPI       025Q;          COMPARE TO EXPONENT SIGN
 0576    CAB405           JZ        INP4;          IF EXPONENT SIGN
 0579    FE0A             CPI       12Q;           SET CARRY IF CHAR IS DIGIT
 057B    D2E505           JNC       INP8;          IF CHAR IS NOT A DIGIT
```

```
057E    2E3F              MVI      L,TMP4;        TO ADDR CURRENT DIGIT
0580    77                MOV      M,A;           SAVE CURRENT DIGIT
0581    21EC06            LXI      H,FTEN;        TO ADDR FLOATING TEN
0584    CD8C02            CALL     MUL;           MULTIPLY BY TEN
0587    2E3B              MVI      L,VALE;        TO ADDR VALUE
0589    CD3E02            CALL     STR;           STORE OLD VALUE TIMES TEN
058C    2C                INR      L;             TO ADDR CURRENT DIGIT
058D    7E                MOV      A,M;           CURRENT DIGIT
058E    0600              MVI      B,0;           CLEAR 2ND WORD OF DIGIT
0590    48                MOV      C,B;           CLEAR 3RD WORD OF DIGIT
0591    50                MOV      D,B;           CLEAR 4TH WORD OF DIGIT
0592    1E08              MVI      E,010Q;        INDICATE DIGIT IS IN REG A
0594    CDFF04            CALL     FLT;           CONVERT DIGIT TO FLOATING PNT
0597    2E3B              MVI      L,VALE;        TO ADDR VALUE
0599    CDD702            CALL     AD;            ADD OLD VALUE TIMES TEN
059C    2E39              MVI      L,TMP2;        TO ADDR DEC PNT FLAG
059E    7E                MOV      A,M;           DECIMAL POINT FLAG
059F    A7                ANA      A              ; SET CONTROL BITS
05A0    CA6705            JZ       INP1;          IF NO DEC PNT ENCOUNTERED
05A3    2D                DCR      L;             TO ADDR INPUT EXPONENT
05A4    46                MOV      B,M;           INPUT EXPONENT
05A5    05                DCR      B;             DECREMENT INPU EXPONENT
05A6    70                MOV      M,B;           UPDATE INPUT EXPONENT
05A7    C36705            JMP      INP1;          TO GET NEXT CHARACTER

05AA    2E39      INP3:            MVI            L,TMP2; TO ADDR DEC PNT FLAG
05AC    AE                XRA      M;             ZERO IF FLAG SET
05AD    77                MOV      M,A;           SET DEC PNT FLAG
05AE    C26705            JNZ      INP1;          IF FLAG NOT ALREADY SET
05B1    C3E505           JMP      INP8;          IF 2ND DEC PNT
                  ;     PROCESS DECIMAL EXPONENT
05B4    CDE106    INP4:            CALL           CHAD; CALL CHAR ADDR SBRTN
```

```
  05B7    7E                MOV       A,M;            NEXT CHARACTER OF STRING
  05B8    47                MOV       B,A;            CURRENT CHARACTER
  05B9    D6FD              SUI       375Q;           COMPARE TO MINUS CHAR
  05BB    5F                MOV       E,A;            CHAR = MINUS SIGN
  05BC    CAC505            JZ        INP5;           IF MINUS SIGN
  05BF    C602              ADI       2;              COMPARE TO PLUS CHAR
  05C1    78                MOV       A,B;            CURRENT CHARACTER
```

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 18
```

```
  05C2    C2C705            JNZ       INP6;           IF NOT PLUS SIGN

  05C5    2C          INP5:           INR             L; TO ADDRESS NEXT CHAR
  05C6    7E                MOV       A,M;            NEXT CHARACTER OF STRING
  05C7    0600        INP6:           MVI             B,0; POSSIBLE DEC EXPONENT
  05C9    FE0A              CPI       12Q;            SET CARRY IF CHAR IS DIGIT
  05CB    D2E505            JNC       INP8;           IF CHAR IS NOT A DIGIT
  05CE    47                MOV       B,A;            DEC EXP EQUAL DIGIT
  05CF    2C                INR       L;              TO ADDRESS NEXT CHAR
  05D0    7E                MOV       A,M;            NEXT CHARACTER OF STRING
  05D1    FE0A              CPI       12Q;            SET CARRY IS NOT A DIGIT
  05D3    D2DE05            JNC       INP7;           IF CHAR IS NOT A DIGIT
                      ;      FORM COMPLETE DECIMAL EXPONENT
  05D6    4F                MOV       C,A;            LS DIGIT OF DEC EXP
  05D7    78                MOV       A,B;            MS DIGIT OF DEC EXP
  05D8    87                ADD       A;              2 * MS DIGIT
  05D9    87                ADD       A;              4 * MS DIGIT
  05DA    80                ADD       B;              5 * MS DIGIT
  05DB    87                ADD       A;              10 * MS DIGIT
```

```
05DC   81              ADD     C;              10 * MS + LS DIGIT
05DD   47              MOV     B,A;            DECIMAL EXPONENT
05DE   7B        INP7:         MOV             A,E; SIGN OF DEC EXPONENT
05DF   A7              ANA     A               ; SET CONTROL BITS
05E0   D2E505          JNC     INP8;           IF SIGN PLUS
05E3   90              SUB     B;              COMPLEMENT DEC EXP
05E4   47              MOV     B,A;            DECIMAL EXPONENT
05E5   2601      INP8:         MVI             H,SCRB; TO ADDRESS SCRATCH BAND
05E7   2E3A            MVI     L,TMP3;         TO ADDRESS INPUT SIGN
05E9   4E              MOV      C,M;           INPUT SIGN
05EA   2E31            MVI     L,ACCS;         TO ADDRESS ACCUM SIGN
05EC   71              MOV     M,C;            ACCUMULATOR SING
05ED   78              MOV     A,B;            DECIMAL EXPONENT
                 ;       CONVERT DECIMAL EXPONENT TO BINARY.
05EE   2E38      INP9:         MVI             L,TMP1; TO  ADDRESS DEC EXPONENT
05F0   86              ADD     M;              ADJUST DECIMAL EXPONENT
05F1   CA5902          JZ      TST;            YN DEC EXP IS ZERO
05F4   77              MOV     M,A;            CURRENT DECIMAL EXPONENT
05F5   21EC06          LXI     H,FTEN;         TO ADDR FLOATING TEN
05F8   F20306          JP      INP10;          IF MULTIPLY REQUIRED
05FB   CDB402          CALL    DIV;            DIVIDE BY TEN
05FE   3E01            MVI     A,1;            TO INCREMENT DEC EXP
0600   C3EE05          JMP     INP9;           TO TEST FOR COMPLETTION
0603   CD8C02    INP10:        CALL            MUL; MULTIPLY BY TEN
0606   D8              RC      ;               RETURN IF OVERFLOW
0607   3EFF            MVI     A,377Q;         TO DECREMENT DEC EXP
0609   C3EE05          JMP     INP9;           TO TEST FOR COMPLETION
                 ;       OUT SUBROUTINE ENTRY POINT.
                 ;         SAVE CHARACTER ADDRESS AND ACCUMULATOR.
060C   2D        OU:   DCR     L;              DECREMENT CHARACTER ADDRESS
060D   CDD406          CALL    SVAD;           SET CHAR ADDR, DIG CNT, DEC EXP
0610   CD5902          CALL    TST;            LOAD ACCUM TO REGISTERS
```

```
 0613    2E3B             MVI       L,VALE;         TO ADDR ACCUM SAVE AREA
 0615    CD3E02           CALL      STR;            CALL REG STR SUBROUTINE
                   ;       OUTPUT SIGN CHARACTER.
 0618    CDE106           CALL      CHAD;           CALL CHAR ADDR SBRTN
 061B    36F0             MVI       M,360Q;         STORE SPACE CHARACTER
 061D    A7               ANA       A               ; SET CONTROL BITS
 061E    CA3A06           JZ        OUT3;           IF ACCUMULATOR IS ZERO
 0621    5F               MOV       E,A;            ACCUMULATOR EXPONENT
 0622    78               MOV       A,B;            ACCUM SIGN AND 1ST FRCTN
 0623    A7               ANA       A               ; SET CONTROL BITS
 0624    7B               MOV       A,E;            ACCUMULATOR EXPONENT
 0625    F22A06           JP        OUT1;           IF ACCUM IS POSITIVE
 0628    36FD             MVI       M,375Q;         CHANGE SIGN TO MINUS

                   ;       SCALE ACCUMULATOR TO .1 = 1. RANGE/
 062A    FE7E      OUT1:            CPI             175Q; COMPARE TO SMALL EXPONENT
 062C    21EC06    OUT2:            LXI             H,FTEN; TO ADDR FLOATING TEN
 062F    DA4406           JC        OUT4;           IF EXPONENT TOO SMALL
 0632    FE81             CPI       201Q;           COMPARE TO LARGE EXP
 0634    DA4F06           JC        OUT5;           IF EXP SMALL ENOUGH
 0637    CDB402           CALL      DIV;            DIVIDE BY TEN
 063A    2601      OUT3:            MVI             H,SCRB; TO ADDRESS SCRATCH BANK
 063C    2E39             MVI       L,TMP2;         TO ADDR DECIMAL EXPONENT
 063E    5E               MOV       E,M;            DECIMAL EXPONENT
 063F    1C               INR       E;              INCREMENT DECIMAL EXPONENT
 0640    73               MOV       M,E;            DECIMAL EXPONENT
```

```
0641   C32C06          JMP      OUT2;         TO TEST FOR SCALING COMPLETE
0644   CD8C02   OUT4:           CALL          MUL; MULTIPLY BY TEN
0647   2E39            MVI      L,TMP2;       TO ADDR DECIMAL EXPONENT
0649   5E              MOV      E,M;          DECIMAL EXPONENT
064A   1D              DCR      E;            DECREMENT DECIMAL EXPONENT
064B   73              MOV      M,E;          DECIMAL EXPONENT
064C   C32A06          JMP      OUT1;         TO TEST FOR SCALING COMPLETE
                ;       ROUND THE VALUE BY ADDING .00000005
064F   CD5002   OUT5:           CALL          ABS; SET ACCUM POSITIVE
0652   21F006          LXI      H,RNDO;       TO ADDRES ROUNDER
0655   CDD702          CALL     AD;           ADD THE ROUNDER
0658   FE81            CPI      201Q;         CHECK FOR OVERFLOW
065A   D22C06          JNC      OUT2;         IF EXP TOO LARGE
                ;       SET DIGIT COUNTS.
065D   2E39            MVI      L,TMP2;       TO ADDR DECIMAL EXPONENT
065F   7E              MOV      A,M;          DECIMAL EXPONENT
0660   5F              MOV      E,A;          DIGITS BEFORE DECIMAL POINT
0661   FE08            CPI      010Q;         COMPARE TO LARGE EXP
0663   DA6806          JC       OUT6;         IF EXPONENT IN RANGE
0666   1E01            MVI      E,1;          DIGITS BEFORE DEC POINT
0668   93       OUT6:           SUB           E; ADJUST DEC EXPONENT
0669   77              MOV      M,A;          DECIMAL EXPONENT
066A   3E07            MVI      A,7;          TOTAL NUMBER OF DIGITS
066C   93              SUB      E;            DIGITS AFTER DECIMAL PNT
066D   2C              INR      L;            TO ADDR 2ND DIGIT CNT


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 20


066E   77              MOV      M,A;          DIGITS AFTER DECIMAL POINT
```

```
066F    1D            DCR       E;          DECREMENT DIGIT COUNT
0670    7B            MOV       A,E;        DIGITS BEFORE DEC PNT
                ;      OUTPUT SIGNIFICANT DIGITS.
0671    2E38    OUT7:           MVI         L,TMP1; TO ADDR DIGIT COUNT
0673    86            ADD       M;          ADJUST DIGIT COUNT
0674    77            MOV       M,A;        NEW DIGIT COUNT
0675    FA9206        JM        OUT8;       IF COUNT RUN OUT
0678    21EC06        LXI       H,FTEN;     TO ADDR FLOATING TEN
067B    CD8C02        CALL      MUL;        MULTIPLY BY TEN
067E    1E08          MVI       E,10Q;      TO PLACE DIGIT IN RE A
0680    CD1605        CALL      FIX;        CONVERT TO FIXED FORMAT
0683    CDE106        CALL      CHAD;       CALL CHAR ADDR SBRT
0686    77            MOV       M,A;        OUTPUT DECIMAL DIGIT
0687    AF            XRA       A;          CLEAR CURRENT DIGIT
0688    1E08          MVI       E,010Q;     BINARY SCALING FACTOR
068A    CDFF04        CALL FLT;             RESTORE VALUE MINUS DIGIT
068D    3EFF          MVI       A,377Q;     TO ADJUST DIGIT CNT
068F    C37106        JMP       OUT7;       YLOOP FOR NEXT DIGIT
0692    2E3A    OUT8:           MVI         L,TMP3; TO ADDR 2ND DIGIT CNT
0694    7E            MOV       A,M;        DIGITS AFTER DECIMAL PNT
0695    36FF          MVI       M,377Q;     SET 2ND COUNT NEG
0697    A7            ANA       A           ; SET CONTROL BITS
0698    FAA506        JM        OUT9;       IF 2ND COUNT RAN OUT
069B    CDE106        CALL      CHAD;       CALL CHAR ADDR SBRTN
069E    36FE          MVI       M,376Q;     STORE DECIMAL POINT
06A0    2601          MVI       H,SCRB;     TO ADDRESS SCRATCH BANK
06A2    C37106        JMP       OUT7;       LOOP FOR NEXT DIGIT
06A5    2D      OUT9:           DCR         L; TO ADDR DECIMAL EXP
06A6    A6            ANA       M           ; DECIMAL EXPONENT
06A7    CACC06        JZ        OUT13;      IF DECIMAL EXPONENT IS ZERO
                ;      OUTPUT DECIMAL EXPONENT.
06AA    06FB          MVI       B,373Q;     PLUS CHARACTER
```

```
06AC    F2B406          JP          OUT10;         IF EXPONENT IS POSITIVE
06AF    06FD            MVI         B,375Q;        CHANGE SIGN TO MINUS
06B1    4F              MOV         C,A;           NEGATIVE EXPONENT
06B2    AF              XRA         A;             ZERO
06B3    91              SUB         C;             COMPLEMENT EXPONENT
06B4    0EFF     OUT10:             MVI            C,377Q; EMBRYO TENS DIGIT

06B6    57       OUT11:             MOV            D,A; UNITS DIGIT
06B7    0C              INR         C;             INCREMENT TENS DIGIT
06B8    D60A            SUI         012Q;          REDUCE REMAINDER
06BA    D2B606          JNC         OUT11;         IF MORE TENS
06BD    3E15            MVI         A,025Q;        EXPONENT SIGN
06BF    CDE106   OUT12:             CALL           CHAD; CALL CHAR ADDR SBRTN
06C2    CD3E02          CALL        STR;           STORE LAST 4 CHARACTER
06C5    2601            MVI         H,SCRB;        TO ADDRESS SCRATCH BANK
06C7    2E3B            MVI         L,VALE;        TO ADDRESS ACCUM SAVE AREA
06C9    C36E02          JMP         LOD;           RESTORE ACCUM AND EXIT
                 ;      OUTPUT 4 SPACES IF EXPONENT IS ZERO.


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 21


06CC    3EF0     OUT13:             MVI            A,360Q; SPACE CHARACTER
06CE    47              MOV         B,A;           SPACE CHARACTER
06CF    4F              MOV         C,A;           SPACE CHARACTER
06D0    57              MOV         D,A;           SPACE CHARACTER
06D1    C3BF06          JMP         OUT12;         TO STORE CHARACTERS
                 ;      SUBROUTINE TO SAVE CHARACTER STRING ADDR.

06D4    7D       SVAD:              MOV            A,L; CHARACTER STRING WORD
```

```
  06D5    44                MOV       B,H;              CHARACTER STRING BANK
  06D6    0E00              MVI       C,0;              INPUT EXP OR DIGIT
  06D8    51                MOV       D,C;              DEC PNT FLAG OR DEC EXP
  06D9    2601              MVI       H,SCRB;           TO ADDRESS SCRATCH BANK
  06DB    2E36              MVI       L,ADRL;           TO ADDR CHAR STRING WORD
  06DD    CD3E02            CALL      STR;              STROE A, B, C, AND D
  06E0    C9                RET       ;                 RETURN T CALLER
                       ;   SUBROUTINE TO OBTAIN NEXT CHARACTER ADDR.
  06E1    2601       CHAD:            MVI               H,SCRB; TO ADDRESS SCRATCH BANK
  06E3    2E36              MVI       L,ADRL;           TO ADDR CHAR STRING WORD
  06E5    5E                MOV       E,M;              CHARACTER STRING WORD
  06E6    1C                INR       E;                TO ADDR NEXT CHARACTER
  06E7    73                MOV       M,E;              UPDATE CHAR STRING WORD
  06E8    2C                INR       L;                TO ADDR CHAR STRING BANK
  06E9    66                MOV       H,M;              CHARACTER STRING WORD
  06EA    6B                MOV       L,E;              CHARACTER STRING WORD
  06EB    C9                RET       ;                 RETURN TO CALLER

  06EC    84200000 FTEN:               DB               204Q,040Q,0,0; FLOATING TEN
  06F0    6856BFAD RNDO:              DB                150Q,126Q,277Q,255Q; .00000005
  06F4    00                DB        0;                CHECKSUM WORD
                            END
NO PROGRAM ERRORS


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 22


                            SYMBOL TABLE

 * 01
```

```
A       0007      ABS     0250      ACC1    0032      ACC2    0033
ACC3    0034      ACCE    0030      ACCS    0031      AD      02D7
ADD10   036B      ADD11   0374      ADD12   037A      ADD17   0386
ADD2    030E      ADD3    032D      ADD9    0350      ADRH    0037
ADRL    0036      ARITH   0200      ARTHB   0002      B       0000
C       0001      CHAD    06E1      CHS     024D *    COMP    03EF
COMP1   03F4      D       0002      DIV     02B4      DIVX    0490
DIVX1   04DB      DIVX2   04DF      DIVX3   04F2      DIVX4   04F6
DIVX5   010D      DIVX6   011E      E       0003      FIX     0516
FIX1    0544      FLT     04FF      FTEN    06EC      H       0004
INIT    022F *    INIT1   0231      INP     054A *    INP1    0567
INP10   0603      INP2    056D      INP3    05AA      INP4    05B4
INP5    05C5      INP6    05C7      INP7    05DE      INP8    05E5
INP9    05EE      L       0005      LOD     026E      LOD1    0278 *
LSH     03BC      LSH1    03BF *    M       0006      MDEX    0395
MUL     023C      MULP1   0009      MULP2   0005      MULP3   0001
MULX    044D      MULX1   0465      MULX2   0468      MULX3   0479
MULX4   0100      MULX5   0473      NORM    0402      NORM1   0404
NORM2   0415      NORM3   0422      OP1A    0027      OP1S    0019
OP2A    0023      OP2S    0015      OP3A    001F      OP3S    0011
OP4A    001C      OP4S    000E      OP4X    0024      OU      060C *
OUT1    062A      OUT10   06B4      OUT11   06B6      OUT12   06BF
OUT13   06CC      OUT2    062C      OUT3    063A      OUT4    0644
OUT5    064F      OUT6    0668      OUT7    0671      OUT8    0692
OUT9    06A5      OVER    002E      OVERF   02CA      OVUN    03B8
PREX    002F      PSW     0006      RND0    06F0      RNDA    02A9
RNDR    043F      ROND    0430      RSH     03C9      RSH0    03CB *
RSH1    03CD      RSH2    03DA      RSH3    03DD      SB      02D4 *
SCR     0100      SCRB    0001      SF      0035      SP      0006
STR     023E      STR1    023F      STR0    023C      SVAD    06D4
TMP1    0038      TMP2    0039      TMP3    003A      TMP4    003F
```

```
TST     0259       TST1    025B       VAL1    003C       VAL2    003D
VAL3    003E       VALE    003B       ZRO     0246       ZR01    0248
```

APPENDIX F

Reactimeter Program

Inhour Equation

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 1


       1           PART   MACRO LAMI,BII
       1                  LXI H,LAMI    ;LOAD ADDRESS OF LAMI  IN HL
       1                  LODE
       1                  LXI H, PER    ;LOAD ADDRESS OF PER IN HL
       1                  CALL MUL      ;MULTIPLY
       1                  LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
       1                  CALL AD       ;AD
       1                  LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
       1                  STRIN
       1                  LXI H,BII     ;LOAD ADDRESS OF BII IN HL
       1                  LODE
       1                  LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
       1                  CALL DIV      ;DIVIDE
       1                  LXI H,RES     ;LOAD ADDRESS OF RES IN HL
       1                  CALL AD       ;ADD
       1                  LXI H,RES     ;LOAD ADDRESS OF RES IN HL
       1                  STRIN
                          ENDM
       1           LODE   MACRO
       1                  CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
       1                                ;FROM THE ADDRESS GIVEN IN HL
                          ENDM
       1           STRIN  MACRO
       1                  CALL STR      ;STR--STORES THE FLOATING-POINT-
       1                                ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
                          ENDM
  022F                    ORG 022FH
  022F   00        INT:   NOP
```

```
0230   C9                RET
023E                     ORG 023EH
023E   00        STR:    NOP
023F   C9                RET
0250                     ORG 0250H
0250   00        ABS:    NOP
0251   C9                RET
024D                     ORG 024DH
024D   00        CHS:    NOP
024E   C9                RET
026E                     ORG 026EH        ;THE FOLLOWING LABELS ARE FOR THE FLOAT-
026E   00        LOD:    NOP              ;ING POINT ROUTINES THAT ARE IN EPROM AND
026F   C9                RET              ;ARE BY CERTAIN ROUTINES TO PERFORM MATHE
028C                     ORG 028CH        ;MATICAL OPERATIONS.  THESE ROUTINES ARE
028C   00        MUL:    NOP              ;IN ANOTHER PRINT OUT AND ARE GIVEN HERE
028D   C9                RET              ;AS DUMMY PROGRAMS TO BE REFERENCE BY THE
02B4                     ORG 02B4H        ;CROSS ASSEMBLER.
02B4   00        DIV:    NOP
02B5   C9                RET
02D4                     ORG 02D4H
02D4   00        SB:     NOP
02D5   C9                RET
02D7                     ORG 02D7H


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 2


02D7   00        AD:     NOP
02D8   C9                RET
04FF                     ORG 04FFH
```

```
04FF    00        FLT:    NOP
0500    C9                RET
054A                      ORG 054AH
054A    00        INP:    NOP
054B    C9                RET
060C                      ORG 060CH
060C    00        OU:     NOP
060D    C9                RET
0000              START:  ORG 0000H
0000    31FF10            LXI SP,10FFH ;LOAD STACK POINTER TO HI 10 AND LO FF
0003    210A13            LXI H,DMANT
0006    36F0              MVI M,360Q
0008    2C                INR L
0009    2C                INR L
000A    2C                INR L
000B    36FE              MVI M,376Q
000D    2C                INR L
000E    2C                INR L
000F    36FF              MVI M,377Q
0011    1607              MVI D,7      ;SET D EQUAL TO 7
0013    CD2F02            CALL INT     ;INT--INITIALIZE SCRATCH PAD MEMORY
0016    210213            LXI H,RES    ;LOAD ADDRESS OF RES IN HL
0019    3600              MVI M,0      ;RES = 0
001B    2D                DCR L
001C    3600              MVI M,0      ;MEM = 0
001E    2D                DCR L
001F    3600              MVI M,0      ;MEM1 = 0
0021    C39C00            JMP WAIT     ;JUMP TO WAIT AND BEGIN PROGRAM
0028              INTRO:  ORG 0028H
0028    D30A      INPUT:  OUT 10       ;GENERATE A DEVICE SELECT PULSE
                                       ;TO DEVICE TEN
002A    210E13            LXI H,DMANT+4;ADDRESS OF THE TENTHS DIGIT OF DMANT
```

```
                                        ;ACCUMULATOR
 002D   DB01            IN  1           ;READ TENTHS DIGIT FROM DEVICE 1 INTO
                                        ;ACCUMULATOR
 002F   77              MOV M,A         ;MOVE A TO DMANT + 4
 0030   2D              DCR L
 0031   2D              DCR L
 0032   DB02            IN  2           ;READ ONES DIGIT FROM DEVICE 2 INTO
                                        ;ACCUMULATOR
 0034   77              MOV M,A         ;MOVE A TO DMANT + 2
 0035   2D              DCR L
 0036   DB03            IN  3           ;READ TENS DIGIT FROM DEVICE 3 INTO
                                        ;ACCUMULATOR
 0038   77              MOV M,A         ;MOVE A TO DMANT + 1
 0039   2D              DCR L


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 3


 003A   CD4A05          CALL INP        ;INP--CHANGES A BCD STRING NUMBER INTO
                                        ; A BINARY-FLOATING-POINT NUMBER
 003D   212513          LXI H,FLX2      ;LOAD ADDRESS OF FLX2 INTO HL
      1         +       STRIN
 0040 1 CD3E02  +       CALL STR        ;STR--STORES THE FLOATING-POINT-
      1         +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
 0043   212513  CMPAR:  LXI H,FLX2      ;LOAD ADDRESS OF FLX2 IN HL
      1         +       LODE
 0046 1 CD6E02  +       CALL LOD        ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
      1         +                       ;FROM THE ADDRESS GIVEN IN HL
 0049   212113          LXI H,FLX1      ;LOAD ADDRESS OF FLX1 IN HL
 004C   E5              PUSH H          ;PUSH HL ON TO THE STACK
```

```
0040    CDD402              CALL SB        ;SUBTRACT
0050    CDA100              CALL FLAG      ;CALL SUBROUTINE FLAG
0053    CD5002              CALL ABS       ;ABSOLUTE VALUE
0056    E1                  POP H          ;POP HL OFF OF THE STACK
0057    CDB402              CALL DIV       ;DIVIDE
005A    213C01              LXI H,TH100    ;LOAD ADDRESS OF TH100 IN HL
005D    CDD402              CALL SB        ;SUBRACT
0060    FA8700              JM TRACK       ;JUMP TO TRACK ON NEGATIVE RESULT
0063    214001              LXI H,NTH10    ;LOAD ADDRESS OF NTH10 IN HL
0066    CDD402              CALL SB        ;SUBTRACT
0069    FA0014              JM CALC        ;JUMP TO CALC ON NEGATIVE RESULT
006C    CD7200              CALL EXCH       ;CALL EXCH
006F    C39C00              JMP WAIT       ;JUMP TO WAIT
0072    212513    EXCH:     LXI H,FLX2     ;LOAD ADDRESS OF FLX2 INTO HL
      1          +          LODE
0075  1 CD6E02   +          CALL LOD       ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
      1          +                         ;FROM THE ADDRESS GIVEN IN HL
0078    212113              LXI H,FLX1     ;LOAD ADDRESS OF FLX1 INTO HL
      1          +          STRIN
007B  1 CD3E02   +          CALL STR       ;STR--STORES THE FLOATING-POINT-
      1          +                         ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
007E    210113              LXI H,MEM      ;LOAD ADDRESS OF MEM INTO HL
0081    3600                MVI M,0        ;MEM = 0
0083    2D                  DCR L
0084    3600                MVI M,0        ;MEM1 = 0
0086    C9                  RET
0087    210113    TRACK:    LXI H, MEM     ;LOAD ADDRESS OF MEM IN HL
008A    34                  INR M          ;INCREMENT CONTENTS OF MEM
008B    CA9100              JZ HMEM        ;JUMP TO HMEM IF RESULT IS ONE
008E    C39C00              JMP WAIT       ;JUMP TO WAIT
0091    2D        HMEM:     DCR L          ;DECREMENT L
0092    34                  INR M          ;INCREMENT MEM+1
```

```
0093    3E04              MVI A,004Q     ;A = 4
0095    9E                SBB M          ;SUBTRACT MEM+1 FROM A
0096    C29C00            JNZ WAIT       ;JUMP TO WAIT IS RESULT IS NOT ZERO
0099    C35901            JMP OTPTO      ;JUMPT TO OTPTO
009C    FB         WAIT:  EI             ;ENABLE FLAG INTERRUPT
009D    00                NOP            ;NO OPERATION
                                         ;THE PURPOSE OF THIS LOOP IS TO LET THE
```

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 4
```

```
                                         ;MICROCOMPUTER IDLE WHILE WAITING FOR THE
                                         ;START OF A NEW TIME INTERVAL SIGNAL FROM
                                         ;THE DPM WHICH OCCURS EVERY 0.01 SEC.
009E    C39C00            JMP WAIT       ;JUMP TO WAIT
00A1    212913     FLAG:  LXI H,SGST     ;LOAD ADDRSS OF SGST IN HL
00A4    FAAA00            JM NEGAT       ;JUMP TO NEGAT IF RESULT IS NEGATIVE
00A7    3600              MVI M,0        ;SET M = 0
00A9    C9                RET            ;RETURN FROM SUBROUTINE
00AA    3601       NEGAT: MVI M,001Q     ;SET M = 001Q
00AC    C9                RET
00AD    210613     CHSGN: LXI H,PER      ;LOAD ADDRESS OF PER IN HL
      1            +      LODE
00B0  1 CD6E02     +      CALL LOD       ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
      1            +                     ;FROM THE ADDRESS GIVEN IN HL
00B3    CD4D02            CALL CHS       ;CHANGE OF SIGN
00B6    210613            LXI H,PER      ;LOAD ADDRESS OF PER IN HL
      1            +      STRIN
00B9  1 CD3E02     +      CALL STR       ;STR--STORES THE FLOATING-POINT-
      1            +                     ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
```

```
00BC     C9                  RET            ;RETURN FROM SUBROUTINE
1400                         ORG 1400H
1400     210113     CALC:    LXI H,MEM      ;LOAD ADDRESS OF MEM IN HL
1403     4E                  MOV C,M        ;MOVE CONTENTS FROM MEM TO C
1404     2D                  DCR L          ;DECREMENT L
1405     46                  MOV B,M        ;MOVE CONTENTS FROM MEM TO B
1406     03                  INX B          ;INCREMENT BC PAIR
1407     3E00                MVI A,0        ;A = 0
1409     1600                MVI D,0        ;REGISTER D = 0
140B     1E18                MVI E,030Q     ;REGISTER E = 24
140D     CDFF04              CALL FLT       ;CALL IFPP ROUTINE FLT TO CONVERT BINARY
                                            ;FIXED POINT TO BINARY FLOATING POINT
1410     210613              LXI H, PER     ;LOAD ADDRESS OF PER IN HL
       1            +        STRIN
1413   1 CD3E02     +        CALL STR       ;STR--STORES THE FLOATING-POINT-
       1            +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
1416     212913              LXI H,SGST     ;LOAD ADDRESS OF SGST IN HL
1419     35                  DCR M          ;DECREMENT SGST
141A     CCAD00              CZ CHSGN       ;CALL CHGSN IF THE RESULT IS ZERO
       1            +        PART LAM1,BI1
141D   1 210001     +        LXI H,LAM1     ;LOAD ADDRESS OF LAMI IN HL
       2            +        LODE
1420   2 CD6E02     +        CALL LOD       ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
       2            +                       ;FROM THE ADDRESS GIVEN IN HL
1423   1 210613     +        LXI H, PER     ;LOAD ADDRESS OF PER IN HL
1426   1 CD8C02     +        CALL MUL       ;MULTIPLY
1429   1 213801     +        LXI H, ONE     ;LOAD ADDRESS ON ONE IN HL
142C   1 CDD702     +        CALL AD        ;AD
142F   1 211013     +        LXI H, HOLD    ;LOAD ADDRESS OF HOLD IN HL
       2            +        STRIN
1432   2 CD3E02     +        CALL STR       ;STR--STORES THE FLOATING-POINT-
       2            +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
```

```
  1435 1 211801   +       LXI H,BI1     ;LOAD ADDRESS OF BII IN HL
       2          +       LODE
  1438 2 CD6E02   +       CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
       2          +                     ;FROM THE ADDRESS GIVEN IN HL
  143B 1 211013   +       LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
  143E 1 CD8402   +       CALL DIV      ;DIVIDE
  1441 1 210213   +       LXI H,RES     ;LOAD ADDRESS OF RES IN HL
  1444 1 CDD702   +       CALL AD       ;ADD
  1447 1 210213   +       LXI H,RES     ;LOAD ADDRESS OF RES IN HL
       2          +       STRIN
  144A 2 CD3E02   +       CALL STR      ;STR--STORES THE FLOATING-POINT-
       2          +                     ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
       1          +       PART LAM2,BI2
  144D 1 210401   +       LXI H,LAM2    ;LOAD ADDRESS OF LAMI IN HL
       2          +       LODE
  1450 2 CD6E02   +       CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
       2          +                     ;FROM THE ADDRESS GIVEN IN HL
  1453 1 210613   +       LXI H, PER    ;LOAD ADDRESS OF PER IN HL
  1456 1 CD8C02   +       CALL MUL      ;MULTIPLY
  1459 1 213801   +       LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
  145C 1 CDD702   +       CALL AD       ;AD
  145F 1 211013   +       LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
       2          +       STRIN
  1462 2 CD3E02   +       CALL STR      ;STR--STORES THE FLOATING-POINT-
       2          +                     ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
  1465 1 211C01   +       LXI H,BI2     ;LOAD ADDRESS OF BII IN HL
```

```
      2         +       LODE
 1468 2 CD6E02  +       CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
      2         +                     ;FROM THE ADDRESS GIVEN IN HL
 146B 1 211013  +       LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
 146E 1 CD8402  +       CALL DIV      ;DIVIDE
 1471 1 210213  +       LXI H,RES     ;LOAD ADDRESS OF RES IN HL
 1474 1 CDD702  +       CALL AD       ;ADD
 1477 1 210213  +       LXI H,RES     ;LOAD ADDRESS OF RES IN HL
      2         +       STRIN
 147A 2 CD3E02  +       CALL STR      ;STR--STORES THE FLOATING-POINT-
      2         +                     ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
      1         +       PART LAM3,BI3
 147D 1 210801  +       LXI H,LAM3    ;LOAD ADDRESS OF LAMI  IN HL
      2         +       LODE
 1480 2 CD6E02  +       CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
      2         +                     ;FROM THE ADDRESS GIVEN IN HL
 1483 1 210613  +       LXI H, PER    ;LOAD ADDRESS OF PER IN HL
 1486 1 CD8C02  +       CALL MUL      ;MULTIPLY
 1489 1 213801  +       LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
 148C 1 CDD702  +       CALL AD       ;AD
 148F 1 211013  +       LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
      2         +       STRIN
 1492 2 CD3E02  +       CALL STR      ;STR--STORES THE FLOATING-POINT-
      2         +                     ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
 1495 1 212001  +       LXI H,BI3     ;LOAD ADDRESS OF BII IN HL


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 6


      2         +       LODE
```

```
1498 2 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2          +                       ;FROM THE ADDRESS GIVEN IN HL
149B 1 211013   +         LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
149E 1 CDB402   +         CALL DIV      ;DIVIDE
14A1 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
14A4 1 CDD702   +         CALL AD       ;ADD
14A7 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
     2          +         STRIN
14AA 2 CD3E02   +         CALL STR      ;STR--STORES THE FLOATING-POINT-
     2          +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
     1          +         PART LAM4,BI4
14AD 1 210C01   +         LXI H,LAM4    ;LOAD ADDRESS OF LAMI IN HL
     2          +         LODE
14B0 2 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2          +                       ;FROM THE ADDRESS GIVEN IN HL
14B3 1 210613   +         LXI H, PER    ;LOAD ADDRESS OF PER IN HL
14B6 1 CD8C02   +         CALL MUL      ;MULTIPLY
14B9 1 213801   +         LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
14BC 1 CDD702   +         CALL AD       ;AD
14BF 1 211013   +         LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
     2          +         STRIN
14C2 2 CD3E02   +         CALL STR      ;STR--STORES THE FLOATING-POINT-
     2          +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
14C5 1 212401   +         LXI H,BI4     ;LOAD ADDRESS OF BII IN HL
     2          +         LODE
14C8 2 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2          +                       ;FROM THE ADDRESS GIVEN IN HL
14CB 1 211013   +         LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
14CE 1 CDB402   +         CALL DIV      ;DIVIDE
14D1 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
14D4 1 CDD702   +         CALL AD       ;ADD
14D7 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
```

```
      2           +        STRIN
14DA 2 CD3E02     +        CALL STR      ;STR--STORES THE FLOATING-POINT-
     2            +                      ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
     1            +        PART LAM5,BI5
14DD 1 211001     +        LXI H,LAM5    ;LOAD ADDRESS OF LAMI IN HL
     2            +        LODE
14E0 2 CD6E02     +        CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2            +                      ;FROM THE ADDRESS GIVEN IN HL
14E3 1 210613     +        LXI H, PER    ;LOAD ADDRESS OF PER IN HL
14E6 1 CD8C02     +        CALL MUL      ;MULTIPLY
14E9 1 213801     +        LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
14EC 1 CDD702     +        CALL AD       ;AD
14EF 1 211013     +        LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
     2            +        STRIN
14F2 2 CD3E02     +        CALL STR      ;STR--STORES THE FLOATING-POINT-
     2            +                      ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
14F5 1 212801     +        LXI H,BI5     ;LOAD ADDRESS OF BII IN HL
     2            +        LODE
```

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 7

```
14F8 2 CD6E02     +        CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2            +                      ;FROM THE ADDRESS GIVEN IN HL
14FB 1 211013     +        LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
14FE 1 CDB402     +        CALL DIV      ;DIVIDE
1501 1 210213     +        LXI H,RES     ;LOAD ADDRESS OF RES IN HL
1504 1 CDD702     +        CALL AD       ;ADD
1507 1 210213     +        LXI H,RES     ;LOAD ADDRESS OF RES IN HL
     2            +        STRIN
```

```
150A 2 CD3E02   +         CALL STR      ;STR--STORES THE FLOATING-POINT-
     2          +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
     1          +         PART LAM6,BI6
150D 1 211401   +         LXI H,LAM6    ;LOAD ADDRESS OF LAMI IN HL
     2          +         LODE
1510 2 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2          +                       ;FROM THE ADDRESS GIVEN IN HL
1513 1 210613   +         LXI H, PER    ;LOAD ADDRESS OF PER IN HL
1516 1 CD8C02   +         CALL MUL      ;MULTIPLY
1519 1 213801   +         LXI H, ONE    ;LOAD ADDRESS ON ONE IN HL
151C 1 CDD702   +         CALL AD       ;AD
151F 1 211013   +         LXI H, HOLD   ;LOAD ADDRESS OF HOLD IN HL
     2          +         STRIN
1522 2 CD3E02   +         CALL STR      ;STR--STORES THE FLOATING-POINT-
     2          +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
1525 1 212C01   +         LXI H,BI6     ;LOAD ADDRESS OF BII IN HL
     2          +         LODE
1528 2 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     2          +                       ;FROM THE ADDRESS GIVEN IN HL
152B 1 211013   +         LXI H,HOLD    ;LOAD ADDRESS OF HOLD IN HL
152E 1 CDB402   +         CALL DIV      ;DIVIDE
1531 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
1534 1 CDD702   +         CALL AD       ;ADD
1537 1 210213   +         LXI H,RES     ;LOAD ADDRESS OF RES IN HL
     2          +         STRIN
153A 2 CD3E02   +         CALL STR      ;STR--STORES THE FLOATING-POINT-
     2          +                       ;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
153D   213401             LXI H,PNLF    ;LOAD ADDRESS OF PNLF
     1          +         LODE
1540 1 CD6E02   +         CALL LOD      ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
     1          +                       ;FROM THE ADDRESS GIVEN IN HL
1543   210613             LXI H,PER     ;LOAD ADDRESS OF PER IN HL
```

```
  1546   CD8402             CALL DIV      ;DIVIDE
  1549   210213             LXI H,RES     ;LOAD ADDRESS OF RES IN HL
  154C   CDD702             CALL AD       ;ADD
  154F   213001             LXI H,CONV    ;LOAD ADDRESS OF CONV IN HL
  1552   CD8C02             CALL MUL      ;MULTIPLY
  1555   211413             LXI H,RHO     ;LOAD ADDRESS OF RHO IN HO
  1558   CD0C06             CALL OU       ;OUTPUT IN BCD FORMAT
  155B   210213             LXI H,RES     ;LOAD ADDRESS OF RES IN HL
  155E   3600               MVI M,0       ;SET RES = 0
  1560   CD7200             CALL EXCH     ;CALL EXCH
  1563   C35001             JMP OUTPT     ;JUMP TO OUTPT


8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 8


  0150                      ORG 0150H
  0150   3A1D13    OUTPT:   LDA RHO+9     ;LOAD THE ACCUMULATOR WITH THE VALUE OF
                                          ;RHO+9
                                          ;THIS VALUE IS TESTED TO DETERMINE IF A
                                          ;BLANK OR THE NUMBER 025,WHICH MEANS AN
                                          ;EXPONENT FOLLOWS
  0153   06F0               MVI B,360Q    ;LOAD B WITH THE VALUE 360(CODE FOR BLANK)
  0155   B8                 CMP B         ;COMPARE B WITH THE ACCUMULATOR, TEST FOR
                                          ;BLANK IN RHO+9
  0156   CA7101             JZ  CHK1      ;JUMP TO CHK1 IF THE RESULT IS ZERO
  0159   3E00      OTPTO:   MVI A,0H      ;LOAD A WITH 0
  015B   1E01               MVI E, 1      ;SET REGISTER EQUAL TO 1
  015D   D304      OTNUM:   OUT 4         ;OUTPUT ACCUMULATOR TO DEVICE 4
  015F   CDD201             CALL CHECK    ;CALL SUBROUTINE CHECK
  0162   D305               OUT 5         ;OUTPUT ACCUMULATOR TO DEVICE 5
```

```
0164   CDD201            CALL CHECK   ;CALL SUBROUTINE CHECK
0167   D306              OUT 6        ;OUTPUT ACCUMULATOR TO DEVICE 6
0169   CDD201            CALL CHECK   ;CALL SUBROUTINE CHECK
016C   D307              OUT 7        ;OUTPUT ACCUMULATOR TO DEVICE 7
016E   C38901            JMP SIGN     ;JUMP TO SIGN
0171   3A1513   CHK1:    LDA RHO+1    ;LOAD ACCUMULATOR WITH RHO+1
0174   06FE              MVI B,376Q   ;LOAD REGISTER B WITH 376Q
0176   B8                CMP B        ;COMPARE B WITH ACCUMULATOR
0177   CA5901            JZ  OTPTO    ;JUMP TO OTPTO
017A   3A1913            LDA RHO+5    ;LOAD ACCUMULATOR WITH RHO+5
017D   06FE              MVI B,376Q   ;LOAD B WITH 376Q(CODE FOR DECIMAL POINT)
017F   B8                CMP B        ;COMPARE B WITH ACCUMULATOR
0180   CA9E01            JZ  OTPT4    ;JUMP TO OTPT4
0183   3A1813            LDA RHO+4    ;LOAD ACCUMULATOR WITH RHO+4
0186   06FE              MVI B,376Q   ;LOAD B WITH 376Q(CODE FOR DECIMAL POINT)
0188   B8                CMP B        ;COMPARE B WITH ACCUMULATOR
0189   CAA701            JZ  OTPT3    ;JUMP TO OTPT3
018C   3A1713            LDA RHO+3    ;LOAD ACCUMULATOR WITH RHO+3
018F   06FE              MVI B,376Q   ;LOAD B WITH 376Q (CODE FOR DECIMAL POINT)
0191   B8                CMP B        ;COMPARE B WITH ACCUMULATOR
0192   CAB001            JZ  OTPT2    ;JUMP TO OTPT2
0195   211513            LXI H,RHO + 1;LOAD H L REGISTER PAIR WITH RHO + 1
0198   7E                MOV A,M      ;MOVE THE DATA FROM THE MEMORY LOCATION
                                      ;ADDRESSED BY THE H L REGISTER PAIR TO
                                      ;THE ACCUMULATOR
0199   1E01              MVI E,1      ;SET REGISTER EQUAL TO ONE
019B   C35D01            JMP OTNUM    ;JUMP TO OTNUM
019E   211813   OTPT4:   LXI H,RHO+4  ;LOAD H L REGISTER PAIR WITH RHO+4
01A1   7E                MOV A,M
01A2   1E04              MVI E,4      ;SET REGISTER EQUAL TO FOUR
01A4   C35D01            JMP OTNUM    ;JUMP TO OTNUM
01A7   211713   OTPT3:   LXI H,RHO+3  ;LOAD H L REGISTER PAIR WITH RHO+3
```

```
  01AA    7E                 MOV A,M
  01AB    1E03               MVI E,3        ;SET REGISTER E EQUAL TO THREE
  01AD    C35D01             JMP OTNUM      ;JUMP TO OTNUM
  01B0    211613   OTPT2:    LXI H,RHO+2    ;LOAD H L REGISTER PAIR WITH RHO+2
```

```
8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 9
```

```
  01B3    7E                 MOV A,M
  01B4    1E02               MVI E,2        ;SET REGISTER EQUAL TO E
  01B6    C35D01             JMP OTNUM      ;JUMP TO OTNUM
  01B9    3A1413   SIGN:     LDA RHO        ;LOAD ACCUMULATOR WITH RHO
  01BC    06F0               MVI B,360Q     ;LOAD B WITH 360Q (CODE FOR SPACE)
  01BE    B8                 CMP B          ;COMPARE B WITH ACCUMULATOR
  01BF    CACA01             JZ  ZERO       ;JUMP TO ZERO
  01C2    3A4401             LDA MINUS      ;LOAD ACCUMULATOR WITH MINUS
  01C5    D308               OUT 8          ;OUTPUT ACCUMULATOR TO DEVICE 8
  01C7    C39C00             JMP WAIT       ;JUMP TO WAIT
  01CA    3A4501   ZERO:     LDA BLANK      ;LOAD ACCUMULATOR WITH PLUS SIGN
  01CD    D308               OUT 8          ;OUTPUT ACCUMULATOR TO DEVICE 8
  01CF    C39C00             JMP WAIT       ;JUMP TO WAIT
  01D2    1D       CHECK:    DCR E          ;DECREMENT REGISTER E
  01D3    CCD901             CZ  WOUT       ;CALL SUBROUTINE WOUT IF RESULT IS ZERO
  01D6    2D                 DCR L          ;DECREMENT L
  01D7    7E                 MOV A,M        ;MOVE THE DATA FROM MEMORY LOCATION
                                            ;ADDRESSED BY THE H L REGISTER PAIR TO
                                            ;THE ACCUMULATOR
  01D8    C9                 RET            ;RETURN
  01D9    214601   WOUT:     LXI H,BLANK+1;LOAD H L REGISTER PAIR WITH BLANK + 1
  01DC    1E01               MVI E,1        ;SET REGISTER EQUAL TO 1
```

```
01DE   C9                  RET           ;RETURN
0100             DATA:     ORG 0100H
0100   7A        LAM1:     DB 172Q       ;LAMDA1 = 0.0127
0101   50                  DB 120Q
0102   13                  DB 023Q
0103   6A                  DB 152Q
0104   7C        LAM2:     DB 174Q       ;LAMDA2 = 0.0317
0105   01                  DB 001Q
0106   D7                  DB 327Q
0107   DC                  DB 334Q
0108   7D        LAM3:     DB 175Q       ;LAMDA3 = 0.115
0109   6B                  DB 153Q
010A   85                  DB 205Q
010B   1E                  DB 036Q
010C   7F        LAM4:     DB 177Q       ;LAMDA4 = 0.311
010D   1F                  DB 037Q
010E   3B                  DB 073Q
010F   65                  DB 145Q
0110   81        LAM5:     DB 201Q       ;LAMDA5 = 1.40
0111   33                  DB 063Q
0112   33                  DB 063Q
0113   33                  DB 063Q
0114   82        LAM6:     DB 202Q       ;LAMDA6 = 3.87
0115   77                  DB 167Q
0116   AE                  DB 256Q

8080 MACRO ASSEMBLER, VER 2.4
    ERRORS = 0 PAGE 10

0117   15                  DB 025Q
```

```
0118    75        BI1:   DB 165Q       ;BETA1 = 0.000247
0119    01               DB 001Q
011A    7F               DB 177Q
011B    C8               DB 310Q
011C    77        BI2:   DB 167Q       ;BETA2 = 0.001385
011D    35               DB 065Q
011E    88               DB 210Q
011F    E4               DB 344Q
0120    77        BI3:   DB 167Q       ;BETA3 = 0.001222
0121    20               DB 040Q
0122    2B               DB 053Q
0123    83               DB 203Q
0124    78        BI4:   DB 170Q       ;BETA4 = 0.002645
0125    2D               DB 055Q
0126    57               DB 127Q
0127    BC               DB 274Q
0128    76        BI5:   DB 166Q       ;BETA5 = 0.000832
0129    5A               DB 132Q
012A    1A               DB 032Q
012B    92               DB 222Q
012C    74        BI6:   DB 164Q       ;BETA6 = 0.000169
012D    31               DB 061Q
012E    35               DB 065Q
012F    97               DB 227Q
0130    91        CONV:  DB 221Q       ;CONVERSION FACTOR FROM ABSOLUTE
0131    43               DB 103Q       ;REACTIVITY TO PCM
0132    50               DB 120Q
0133    00               DB 000Q
0134    73        PNLF:  DB 163Q       ;PROMP NEUTRON LIFETEME = 0.0001
0135    51               DB 121Q
0136    B7               DB 267Q
0137    16               DB 026Q
```

```
0138   81      ONE:    DB 201Q        ;NUMBER VALUE IS 1.000
0139   00              DB 000Q
013A   00              DB 000Q
013B   00              DB 000Q
013C   7A      TH100:  DB 172Q        ;NUMBER VALUE EQUALS 0.01
013D   23              DB 043Q
013E   D7              DB 327Q
013F   0A              DB 012Q
0140   7D      NTH10:  DB 175Q        ;NUMBER VALUE IS 0.1
0141   4C              DB 114Q
0142   CC              DB 314Q
0143   CD              DB 315Q
0144   0B      MINUS:  DB 013Q        ;CODE FOR MINUS
0145   0D      BLANK:  DB 015Q        ;CODE FOR BLANK
1300           RWMEM:  ORG 1300H
1300           MEM1:   DS 1           ;HIGH YTE FOR PERIOD COUNTER
1301           MEM:    DS 1           ;LOW BYTE FOR PERIOD COUNTER
1302           RES:    DS 4           ;STORAGE LOCATION FOR SUMMING TERM

8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 11

1306           PER:    DS 4           ;STORAGE FOR REACTOR PERIOD
130A           DMANT:  DS 6           ;STORAGE LOCATION INPUT DATA(BCD FORM)
1310           HOLD:   DS 4           ;TEMPORARY SPACE
1314           RHO:    DS 13          ;REACTIVITY IN PCM
1321           FLX1:   DS 4           ;STORAGE SPACE FOR POWER1
1325           FLX2:   DS 4           ;STORAGE SPACE FOR POWER2
1329           SGST:   DS 1           ;STORAGE SPACE FOR PERIOD SIGN
                       END
```

```
NO PROGRAM ERRORS


8080 MACRO ASSEMBLER, VER 2.4
     ERRORS = 0 PAGE 12


                        SYMBOL TABLE

 * 01

 A       0007     ABS     0250     AD      02D7     B       0000
 BI1     0118     BI2     011C     BI3     0120     BI4     0124
 BI5     0128     BI6     012C     BLANK   0145     C       0001
 CALC    1400     CHECK   01DA     CHK1    0171     CHS     024D
 CHSGN   00AD     CMPAR   0043 *   CONV    0130     D       0002
 DATA    01E7 *   DIV     02B4     DMANT   130A     E       0003
 EXCH    0072     FLAG    00A1     FLT     04FF     FLX1    1321
 FLX2    1325     H       0004     HMEM    0091     HOLD    1310
 INP     054A     INPUT   0028 *   INT     022F     INTRO   0024 *
 L       0005     LAM1    0100     LAM2    0104     LAM3    0108
 LAM4    010C     LAM5    0110     LAM6    0114     LOD     026E
 LODE    0366     M       0006     MEM     1301     MEM1    1300 *
 MINUS   0144     MUL     028C     NEGAT   00AA     NTH10   0140
 ONE     0138     OTNUM   015D     OTPTO   0159     OTPT2   01B0
 OTPT3   01A7     OTPT4   019E     OU      060C     OUTPT   0150
 PART    037D     PER     1306     PNLF    0134     PSW     0006
 RES     1302     RHO     1314     RWMEM   0146 *   SB      02D4
 SGST    1329     SIGN    01B9     SP      0006     START   060E *
 STR     023E     STRIN   034E     TH100   013C     TRACK   0087
 WAIT    009C     WOUT    01E1     ZERO    01D2
```

THE DESIGN, CONSTRUCTION, AND TESTING OF A REACTIMETER

by

Kim Allen Jones

(ABSTRACT)

A reactimeter has been developed to measure the neutron reactivity of the Virginia Polytechnic Institute and State University nuclear research reactor. The reactimeter will be employed in monitoring reactivity changes of samples entering and leaving the reactor.

The reactimeter is comprised of a compensated ion chamber that measures the neutron flux of the reactor and a microcomputer that performs the reactivity calculations. The calculations are based on the six group, point reactor kinetics equations. To simplify the algorithm programming into the microcomputer, the prompt jump approximation is used. The entire reactimeter program can be stored in 2 K of memory, but it requires a separate program of elementary mathematical subroutines. This second program performs all the mathematical operations and requires 1.25 K of memory.