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

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
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>General References</td>
<td>5</td>
</tr>
<tr>
<td>II. Theory</td>
<td>6</td>
</tr>
<tr>
<td>Introduction</td>
<td>6</td>
</tr>
<tr>
<td>Background</td>
<td>6</td>
</tr>
<tr>
<td>Point Reactor Kinetics</td>
<td>7</td>
</tr>
<tr>
<td>Neutron Generation Time versus Prompt Neutron Lifetime</td>
<td>11</td>
</tr>
<tr>
<td>Effective Delayed Neutron Fraction</td>
<td>12</td>
</tr>
<tr>
<td>Prompt Jump Approximation.</td>
<td>13</td>
</tr>
<tr>
<td>Feedback Effects</td>
<td>17</td>
</tr>
<tr>
<td>Six Delayed Neutron Groups</td>
<td>17</td>
</tr>
<tr>
<td>Computer Algorithm Equation.</td>
<td>18</td>
</tr>
<tr>
<td>Conclusion</td>
<td>22</td>
</tr>
<tr>
<td>III. Equipment</td>
<td>24</td>
</tr>
<tr>
<td>Introduction</td>
<td>24</td>
</tr>
<tr>
<td>Compensated Ion Chamber</td>
<td>24</td>
</tr>
<tr>
<td>Keithley Model 411 Micro-microammeter</td>
<td>28</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Panel Meter</td>
<td>30</td>
</tr>
<tr>
<td>Mark 80 Microcomputer</td>
<td>30</td>
</tr>
<tr>
<td><strong>IV. Hardware</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>32</td>
</tr>
<tr>
<td>Keithley Micro-microammeter Interface</td>
<td>32</td>
</tr>
<tr>
<td>Digital Panel Meter Interface</td>
<td>36</td>
</tr>
<tr>
<td>Interrupt Timing Circuit</td>
<td>44</td>
</tr>
<tr>
<td>Display Interface</td>
<td>48</td>
</tr>
<tr>
<td>Decoding Circuit</td>
<td>53</td>
</tr>
<tr>
<td>500 KHz Clock Speed</td>
<td>53</td>
</tr>
<tr>
<td>Conclusion</td>
<td>57</td>
</tr>
<tr>
<td><strong>V. Software</strong></td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>59</td>
</tr>
<tr>
<td>Neutron Flux Algorithm</td>
<td>59</td>
</tr>
<tr>
<td>Reactivity Algorithm</td>
<td>59</td>
</tr>
<tr>
<td>Time Interval</td>
<td>64</td>
</tr>
<tr>
<td>Reactimeter Program Introduction</td>
<td>66</td>
</tr>
<tr>
<td>INPUT.</td>
<td>70</td>
</tr>
<tr>
<td>SELECT</td>
<td>73</td>
</tr>
<tr>
<td>HRFC-LRFC</td>
<td>79</td>
</tr>
<tr>
<td>DIRECT</td>
<td>81</td>
</tr>
<tr>
<td>PRECURi</td>
<td>87</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REACT.</td>
<td>95</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>100</td>
</tr>
<tr>
<td>WAIT</td>
<td>117</td>
</tr>
<tr>
<td>START.</td>
<td>119</td>
</tr>
<tr>
<td>Conclusion</td>
<td>119</td>
</tr>
<tr>
<td>VI. Conclusions</td>
<td>123</td>
</tr>
<tr>
<td>Conclusion</td>
<td>123</td>
</tr>
<tr>
<td>Reactimeter Program.</td>
<td>127</td>
</tr>
<tr>
<td>VII. Recommendations</td>
<td>148</td>
</tr>
<tr>
<td>References</td>
<td>151</td>
</tr>
</tbody>
</table>

Appendix

A. Calculation of $\beta_{\text{eff}}$                                      | 152  |
B. Intel 8080 Float Point Package.                                         | 153  |
C. Instrument Documentation.                                               | 177  |
D. Reactimeter Program - Kinetics Equations.                               | 196  |
E. Intel 8080 Float Point Program.                                         | 236  |
F. Reactimeter Program - Inhour Equation                                   | 276  |
Vita                                                                       | 296  |
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Diagram for a Typical Neutron Precursor</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Neutron Flux Change for a Positive Reactivity Insertion</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Block Diagram of Reactimeter</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Compensated Ion Chamber Detector</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Gray Shaft Encoder</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Gray Shaft Encoder Circuit</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Block Diagram of Gray Shaft Encoder</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Keithley Interface Circuit</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Digital Panel Meter Interface Circuit</td>
<td>41</td>
</tr>
<tr>
<td>4.6</td>
<td>Zero Input Interface Circuit</td>
<td>42</td>
</tr>
<tr>
<td>4.7</td>
<td>Connection Diagram for Digital Panel Meter</td>
<td>45</td>
</tr>
<tr>
<td>4.8</td>
<td>0.2 sec Timing Interval Circuit</td>
<td>47</td>
</tr>
<tr>
<td>4.9</td>
<td>Output Interface Circuit</td>
<td>50</td>
</tr>
<tr>
<td>4.10</td>
<td>Decoding Circuit</td>
<td>54</td>
</tr>
<tr>
<td>4.11</td>
<td>Automatic Single Step Mode Circuit</td>
<td>58</td>
</tr>
<tr>
<td>5.1</td>
<td>Main Flowchart</td>
<td>71</td>
</tr>
<tr>
<td>5.2</td>
<td>INPUT Routine</td>
<td>74</td>
</tr>
<tr>
<td>5.3</td>
<td>SELECT Routine</td>
<td>80</td>
</tr>
<tr>
<td>5.4</td>
<td>HRFC Routine</td>
<td>82</td>
</tr>
<tr>
<td>5.5</td>
<td>LRFC Routine</td>
<td>85</td>
</tr>
<tr>
<td>5.6</td>
<td>DIRECT Routine</td>
<td>88</td>
</tr>
<tr>
<td>5.7</td>
<td>PRECURi Routines</td>
<td>91</td>
</tr>
<tr>
<td>Page</td>
<td>Figure Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>5.8</td>
<td>TRANSFER Routine</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>REACT Routine</td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>OUTPUT Routine</td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>CHECK Subroutine</td>
<td></td>
</tr>
<tr>
<td>5.12</td>
<td>WOUT Subroutine</td>
<td></td>
</tr>
<tr>
<td>5.13</td>
<td>WAIT Loop</td>
<td></td>
</tr>
<tr>
<td>5.14</td>
<td>START Routine</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Reactivity versus Reactor Period</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Connection Diagram for Digital Panel Meter- Modified Version</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>100 Hz Timing Circuit for the Digital Panel Meter</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>Interrupt Circuit for the Digital Panel Meter</td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>Main Flowchart - Inhour Program</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>INPUT Routine</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>CMPAR Routine</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>TRACK Routine</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>EXCH Subroutine</td>
<td></td>
</tr>
<tr>
<td>6.10</td>
<td>CALC Routine</td>
<td></td>
</tr>
<tr>
<td>6.11</td>
<td>CHSGN Subroutine</td>
<td></td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Delayed Neutron Precursors</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>$^{235}\text{U}$ Delayed Neutron Data</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>$^{235}\text{U}$ Delayed Neutron Decay Constants</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Current Measured vs Voltage Output</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Gray Code</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>4-Input AND Gate Truth Table for Devices Select Pulses IN 0 through IN 3</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Binary Representation of Output Data</td>
<td>52</td>
</tr>
<tr>
<td>4.4</td>
<td>Truth Table for a 4-Line-to-16-Line Decoder</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>Peripheral Device vs Clock Pulse</td>
<td>56</td>
</tr>
<tr>
<td>5.1</td>
<td>Mantissa Values vs Output Voltage</td>
<td>61</td>
</tr>
<tr>
<td>5.2</td>
<td>Conversion Constants for Neutron Flux Algorithm</td>
<td>63</td>
</tr>
<tr>
<td>5.3</td>
<td>Constants for Reactivity Algorithm</td>
<td>67</td>
</tr>
<tr>
<td>5.4</td>
<td>Device Codes and Peripheral Devices</td>
<td>78</td>
</tr>
<tr>
<td>5.5</td>
<td>ASCII Coded 13-Byte Representation of Floating-Point Decimal Numbers</td>
<td>110</td>
</tr>
<tr>
<td>5.6</td>
<td>Decimal Point Location and Digital Display</td>
<td>114</td>
</tr>
<tr>
<td>6.1</td>
<td>Instrumentation Calibration</td>
<td>125</td>
</tr>
</tbody>
</table>
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 of 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 monitors 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


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

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, \( \rho \), 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, \( \phi \), 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, $\rho$, 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

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

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)],

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

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. In their most conventional form, the point reactor kinetics equations are as follows:

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

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

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),

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

\(\beta_i\) is the delayed neutron fraction of precursor \(i\),

\(\beta\) is the total delayed neutron fraction, and

\(\Lambda\) 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,

\[
\phi(0) = \phi_0,
\]

\[
C_i(0) = C_{i0} = \phi \frac{\beta_i}{\Lambda} \lambda_i,
\]

\[
\rho(0) = \rho_0 = 0.
\]
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 $\bar{C_i}(t)$ be the $i^{th}$ precursor density (precursor of group $i$/cm$^3$), $\bar{v_i}$(cm/sec) be the velocity of the $i^{th}$ precursor, $\lambda_i$(sec$^{-1}$) be the decay constant of the $i^{th}$ precursor, and $\beta_i$ be the delayed neutron fraction of the $i^{th}$ precursor, such that $\beta = \sum_{i=1}^{6} \beta_i$. $\beta$ 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)]
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, $\Lambda$, is used instead of the prompt neutron lifetime, $\lambda$. 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 $\lambda$ were used in equations (2.3) and (2.4) instead of $\Lambda$, $\Lambda$ would be replaced by $\lambda/k$. The prompt neutron lifetime is based on the reciprocal probability of the destruction of the neutron.

$$\lambda = \frac{\text{Total neutron population in the system at time } t}{\text{Rate of neutron loss in the system}} \quad (2.6)$$

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. $\lambda$ is normalized to the fission event and is defined as, $\lambda = 1/k$. 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, $\delta$. It is more convenient to reference fission events than removal processes, and employ parameters based on production, $\rho$, $\delta$, and $\lambda$.

Effective Delayed Neutrons Fraction

Each reactor has a characteristic effective delayed neutron fraction, $\beta_{\text{eff}}$. 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,

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

where $\beta_i^*$ is the effective delayed neutron fraction of the $i^{th}$ group, $\tau_p$ is the Fermi age of the prompt neutrons, and $\tau_i$ is the Fermi age of the $i^{th}$ 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)].

$\beta_{\text{eff}}$ 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 $\beta_{\text{eff}}$ (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 $\beta_{\text{eff}}$ is being used, and $\beta_{\text{eff}} = \sum_{i=1}^{6} \beta_i^*$, and $\beta$ is very close to $\beta_{\text{eff}}$, an approximation for $\beta_i^*$ is used. For U$^{235}$, $\beta_i/\beta$ are tabulated; therefore the approximation for $\beta_i^*$ that is used is,

$$\beta_i^* = (\beta_i/\beta) \beta_{\text{eff}}^*$$

(2.8)

$\beta_i^*$ is not exact, but is precise enough for the simple model employed. As will be shown, the values of $\beta_i^*$ do not need to be calculated. In the development of the computer algorithm later in the chapter, the ratio of $\beta_i^*/\beta_{\text{eff}}$ is used. This value reduces to $\beta_i/\beta$ 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,

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

(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

\[
\phi(t) = \sum_{i=1}^{7} A_j \exp(\omega_j t),
\]

(2.10)

where \(\omega_j\) (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

\[
\rho_0 = \beta + \Lambda \omega - \sum_{i=1}^{6} \frac{\beta_i}{\omega + \lambda_i} .
\]

(2.11)

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

\[
\rho_0 = \Lambda \omega + \sum_{i=1}^{6} \frac{\beta_i \omega}{\omega + \lambda_i} ,
\]

(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. \(\omega_1\) through \(\omega_6\) refer to the delayed neutron groups with \(\omega_1\) representing the delayed neutron groups which has the largest \(\omega\) and \(\omega_6\) representing the delayed neutron group with the smallest \(\omega\).
\( w_7 \) represents the prompt neutrons and is the smallest of all the \( \omega \) roots.

The effect of \( \omega \) can be seen in Figure 2.2. During time intervals of two seconds or less, \( \phi(t) \) has the shape of the term, \( A_7 \exp(w_7 t) \). 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 \( \phi_0 \) to \( \phi_1 \) (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 \( \phi_0 \) to \( \phi_1 \) 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)],

\[ \frac{1}{\lambda} = \frac{1}{\sum \beta_i} \left( \sum \frac{\beta_i}{\lambda_i} \right). \]  

(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 \( a_R = \beta / \lambda \) 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 \( Y_i(t) = C_i(t) \lambda_i / \beta_i \). If one multiplies \( \rho(t) \) by \( \beta / \beta \) and \( \lambda_i C_i(t) \) by \( \lambda \beta_i / \lambda \beta_i \beta \) in equation (2.3) and everything in equation (2.4) by \( \lambda \beta_i / \lambda \beta_i \) and simplifies, (2.3) and (2.4) become

\[
\frac{d\phi}{dt} = a_R [\phi(t)(\rho'(t) - 1) + \sum_{i=1}^{\delta} a_i Y_i(t)], \tag{2.14}
\]
Table 2.1
Delayed Neutron Precursors

<table>
<thead>
<tr>
<th>Group Number</th>
<th>Precursor</th>
<th>Half-life (second)</th>
<th>ln t1/2/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Br(^{87})</td>
<td>54.5</td>
<td>3.99</td>
</tr>
<tr>
<td>2</td>
<td>I(^{137})</td>
<td>24.4</td>
<td>3.19</td>
</tr>
<tr>
<td></td>
<td>Br(^{88})</td>
<td>16.3</td>
<td>2.79</td>
</tr>
<tr>
<td>3</td>
<td>Br(^{(89)})</td>
<td>6.3</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>Rb(^{(93,94)})</td>
<td>~6.</td>
<td>1.79</td>
</tr>
<tr>
<td>4</td>
<td>I(^{139})</td>
<td>2.0</td>
<td>.69</td>
</tr>
<tr>
<td></td>
<td>(Cs, Sb, or Te)</td>
<td>(1.6-2.4)</td>
<td>.47 – .88</td>
</tr>
<tr>
<td></td>
<td>Br(^{(90,91)})</td>
<td>1.6</td>
<td>.47</td>
</tr>
<tr>
<td></td>
<td>Kr(^{93})</td>
<td>~1.5</td>
<td>.41</td>
</tr>
<tr>
<td>5</td>
<td>I(^{140})</td>
<td>0.5</td>
<td>-.69</td>
</tr>
<tr>
<td>6</td>
<td>(Br, Rb, As ?)</td>
<td>0.2</td>
<td>-1.61</td>
</tr>
</tbody>
</table>

*Uncertain quantities are indicated by parentheses. [pg. 99 in (2)]
Table 2.2
\( ^{235}U \) Delayed Neutron Data

<table>
<thead>
<tr>
<th>Group</th>
<th>Decay Constant (sec(^{-1}))</th>
<th>Fractional Yield ( \beta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0127 ± 0.0002</td>
<td>0.000247</td>
</tr>
<tr>
<td>2</td>
<td>0.0317 ± 0.0008</td>
<td>0.001385</td>
</tr>
<tr>
<td>3</td>
<td>0.115 ± 0.003</td>
<td>0.001222</td>
</tr>
<tr>
<td>4</td>
<td>0.311 ± 0.008</td>
<td>0.002645</td>
</tr>
<tr>
<td>5</td>
<td>1.40 ± 0.081</td>
<td>0.000832</td>
</tr>
<tr>
<td>6</td>
<td>3.87 ± 0.369</td>
<td>0.000169</td>
</tr>
</tbody>
</table>

\[
\beta = \sum_{i=1}^{6} \beta_i = 0.0065 ± 0.0002
\]

[pg. 22 in (8)]
Table 2.3

$^{235}$U Delayed Neutron Decay Constants

for One, Two, and Three Groups

<table>
<thead>
<tr>
<th>Groups</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$\lambda_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 group</td>
<td>$0.102$ sec$^{-1}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 groups</td>
<td>$0.0387$ sec$^{-1}$</td>
<td>$0.399$ sec$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>3 groups</td>
<td>$0.0285$ sec$^{-1}$</td>
<td>$0.202$ sec$^{-1}$</td>
<td>$1.57$ sec$^{-1}$</td>
</tr>
</tbody>
</table>
\[
\frac{dY_i}{dt}(t) = \lambda_i (\phi(t) - Y_i(t)), \quad i = 1 \text{ to } 6. \quad (2.15)
\]

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

\[
\begin{align*}
\phi(0) &= \phi_0, \\
Y_i(0) &= Y_{i0} = \phi_0, \\
\rho'(0) &= \rho'_0 = 0.
\end{align*} \quad (2.16)
\]

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

\[
\rho'(t) = \frac{\phi(t)}{\varphi(t)} - \sum_{i=1}^{6} a_i \frac{\phi(t)}{\varphi(t)} Y_i(t). \quad (2.17)
\]

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,

\[
Y_i(t) = \phi_0 \exp(-\lambda_i t) + \lambda_i \int_0^t \phi(t) \exp(-\lambda_i (t - \tau)) \, d\tau. \quad (2.18)
\]

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 $2.5 \times 10^2$ to $2.5 \times 10^{10}$ neutrons/cm$^2$-sec, in fields where very high gamma radiation
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 \( K = 2.5 \times 10^{13} \text{ nvt} / \text{amperes} \) (nv = neutrons/cm²/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: \(^{10}\text{B}(\text{n}, \alpha)^7\text{Li}\). 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

[p. 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: $1 \times 10^{-11}$, $3 \times 10^{-11}$, $1 \times 10^{-10}$, $3 \times 10^{-10}$, $1 \times 10^{-9}$, $\ldots$, $3 \times 10^{-4}$, and $1 \times 10^{-3}$ amperes. The lowest range setting, $1 \times 10^{-11}$, is not used because values of neutron flux at this setting are of no concern. There is a $\pm 2\%$ accuracy for the ranges from $1 \times 10^{-3}$ to $3 \times 10^{-7}$ and a $\pm 4\%$ 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.

<table>
<thead>
<tr>
<th>Range Setting</th>
<th>Current Range Limits</th>
<th>Current Measured (amps)</th>
<th>Voltage Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3 \times 10^{-10}$</td>
<td>$1 \times 10^{-10}$ $3 \times 10^{-10}$</td>
<td>$2.0 \times 10^{-10}$</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1.5 \times 10^{-10}$</td>
<td>25.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$2.5 \times 10^{-10}$</td>
<td>75.0</td>
</tr>
<tr>
<td>$1 \times 10^{-9}$</td>
<td>$3 \times 10^{-10}$ $1 \times 10^{-9}$</td>
<td>$4.0 \times 10^{-10}$</td>
<td>14.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$6.5 \times 10^{-10}$</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$8.0 \times 10^{-10}$</td>
<td>71.5</td>
</tr>
<tr>
<td>$3 \times 10^{-9}$</td>
<td>$1 \times 10^{-9}$ $3 \times 10^{-9}$</td>
<td>$2.0 \times 10^{-9}$</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1.1 \times 10^{-9}$</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1.6 \times 10^{-9}$</td>
<td>30.0</td>
</tr>
<tr>
<td>$1 \times 10^{-5}$</td>
<td>$3 \times 10^{-6}$ $1 \times 10^{-5}$</td>
<td>$5.5 \times 10^{-6}$</td>
<td>35.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$6.5 \times 10^{-6}$</td>
<td>50.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$7.5 \times 10^{-6}$</td>
<td>64.3</td>
</tr>
<tr>
<td>$3 \times 10^{-5}$</td>
<td>$1 \times 10^{-5}$ $3 \times 10^{-5}$</td>
<td>$1.0 \times 10^{-5}$</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$1.5 \times 10^{-5}$</td>
<td>25.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$2.5 \times 10^{-5}$</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$3.0 \times 10^{-5}$</td>
<td>100.0</td>
</tr>
</tbody>
</table>
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 80R 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 80R 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). 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 Gray 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 Gray 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

<table>
<thead>
<tr>
<th>Position</th>
<th>Gray Code</th>
<th>Binary Equivalent Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>1111</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>1000</td>
<td>8</td>
</tr>
</tbody>
</table>
ALL HOLES DIA.=.20 CM.
DRILLED COMPLETELY
THROUGH.

ALL MEASUREMENTS ARE
IN CENTIMETERS.

.32 CM. SET SCREW
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

<table>
<thead>
<tr>
<th>Logic State of</th>
<th>Logic State of</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN 0</td>
<td>IN 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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,

$$Hz = \frac{1.443}{(R_A + 2R_B)} C$$  \hspace{1cm} (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 = 000₈ and LO = 050₈ 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

<table>
<thead>
<tr>
<th>Output Data</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 1 1 1</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>9</td>
<td>0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>plus sign</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>minus sign</td>
<td>1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>blank</td>
<td>0 0 0 0 1 1 0 1</td>
</tr>
</tbody>
</table>
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 IN and 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>
<thead>
<tr>
<th>Octal Device Code</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D C B A</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>0 0 0 0</td>
<td>0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>001</td>
<td>0 0 0 1</td>
<td>1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>002</td>
<td>0 0 1 0</td>
<td>1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>003</td>
<td>0 0 1 1</td>
<td>1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>004</td>
<td>0 1 0 0</td>
<td>1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>005</td>
<td>0 1 0 1</td>
<td>1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>006</td>
<td>0 1 1 0</td>
<td>1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>007</td>
<td>0 1 1 1</td>
<td>1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>010</td>
<td>1 0 0 0</td>
<td>1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>011</td>
<td>1 0 0 1</td>
<td>1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>012</td>
<td>1 0 1 0</td>
<td>1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>013</td>
<td>1 0 1 1</td>
<td>1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>014</td>
<td>1 1 0 0</td>
<td>1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1</td>
</tr>
<tr>
<td>015</td>
<td>1 1 0 1</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1</td>
</tr>
<tr>
<td>016</td>
<td>1 1 1 0</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1</td>
</tr>
<tr>
<td>017</td>
<td>1 1 1 1</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0</td>
</tr>
</tbody>
</table>
Table 4.5
Peripheral Device vs Clock Pulse

<table>
<thead>
<tr>
<th>Peripheral Device</th>
<th>Device Code</th>
<th>Clock Pulse</th>
</tr>
</thead>
<tbody>
<tr>
<td>8095 Three-state buffer</td>
<td>000</td>
<td>Negative</td>
</tr>
<tr>
<td>8095 Three-state buffer</td>
<td>001</td>
<td>Negative</td>
</tr>
<tr>
<td>8095 Three-state buffer</td>
<td>002</td>
<td>Negative</td>
</tr>
<tr>
<td>8095 Three-state buffer</td>
<td>003</td>
<td>Negative</td>
</tr>
<tr>
<td>TIL309 Display</td>
<td>004</td>
<td>Negative</td>
</tr>
<tr>
<td>TIL309 Display</td>
<td>005</td>
<td>Negative</td>
</tr>
<tr>
<td>TIL309 Display</td>
<td>006</td>
<td>Negative</td>
</tr>
<tr>
<td>TIL309 Display</td>
<td>007</td>
<td>Negative</td>
</tr>
<tr>
<td>TIL309 Display</td>
<td>010</td>
<td>Negative</td>
</tr>
<tr>
<td>7475 Flip-flop</td>
<td>011</td>
<td>Positive</td>
</tr>
<tr>
<td>Four 7475 Flip-flops</td>
<td>012</td>
<td>Positive</td>
</tr>
<tr>
<td>7490 Decade Counter</td>
<td>013</td>
<td>Negative</td>
</tr>
</tbody>
</table>
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, \( \Delta t \), the reactivity is calculated every 2 \( \Delta t \). The time interval, \( \Delta t \), 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,

\[ ACMAN = 1 + MANT \times \frac{3-1}{100}, \]  

which simplifies to

\[ ACMAN = 1 + MANT \times 0.02. \]  

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,

\[ ACMAN = 3 + MANT \times \frac{10-3}{100}, \]  

(5.2)

(5.3)
### Table 5.1

**Mantissa Value vs Output Voltage**

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

<table>
<thead>
<tr>
<th>Value of Mantissa</th>
<th>Output Voltage (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>000.0</td>
</tr>
<tr>
<td>1.50</td>
<td>25.0</td>
</tr>
<tr>
<td>2.00</td>
<td>50.0</td>
</tr>
<tr>
<td>2.50</td>
<td>75.0</td>
</tr>
<tr>
<td>3.00</td>
<td>100.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value of Mantissa</th>
<th>Output Voltage (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.00</td>
<td>000.0</td>
</tr>
<tr>
<td>4.00</td>
<td>14.3</td>
</tr>
<tr>
<td>5.00</td>
<td>28.6</td>
</tr>
<tr>
<td>6.00</td>
<td>42.9</td>
</tr>
<tr>
<td>7.00</td>
<td>57.2</td>
</tr>
<tr>
<td>8.00</td>
<td>71.5</td>
</tr>
<tr>
<td>9.00</td>
<td>85.8</td>
</tr>
<tr>
<td>10.00</td>
<td>100.0</td>
</tr>
</tbody>
</table>
which simplifies to

\[ \text{ACMAN} = 3 + \text{MANT} \times 0.07. \]  (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, \( K = 2.5 \times 10^{13} \text{ nv/amp} \), from the compensated ion chamber. The neutron flux is equal to the current times \( K \).

\[ \text{FLUX} = \text{Current} \times 2.5 \times 10^{13} \text{ nv/amp}, \]  (5.5)

which is equivalent to

\[ \text{FLUX} = \text{ACMAN} \times 10^J \times 2.5 \times 10^{13} \text{ nv/amp}, \]  (5.6)

where \( J = -11, -10, -9, \ldots -4 \). For simplicity, the constants \( 10^J \) and \( K \) can be defined as a new constant, \( \text{CONn} \). 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 \( \text{CONn} \) and their binary-floating-point values. Equation (5.6) can be simplified to the following,
Table 5.2
Conversion Constants for Neutron Flux Algorithm

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

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

\[ Y_i(t) = \phi_0 \exp(-\lambda_i t) + \lambda_i \int_0^t \phi(\tau) \exp(-\lambda_i(t-\tau)) d(\tau). \] (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 \( Y_i(t_k) \) 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).
\[ Y_1(t_k) = \phi_0 \exp(-\lambda_1 t) + \frac{\lambda_1 \Delta t}{3} \left[ \phi(t_0) \exp(-\lambda_1 (t_k - t_0)) + 4\phi(t_1) \exp(-\lambda_1 (t_k - t_1)) + 2\phi(t_2) \exp(-\lambda_1 (t_k - t_2)) \right] + \ldots + 4\phi(t_{k-1}) \exp(-\lambda_1 (t_k - t_{k-1})) + \phi(t_k) \] (5.8)

\[ Y_1(t_{k+2}) = \phi_0 \exp(-\lambda(t_{k+2})) + \frac{\lambda_1 \Delta t}{3} \left[ \phi(t_0) \exp(-\lambda_1 (t_{k+2} - t_0)) + 4\phi(t_1) \exp(-\lambda_1 (t_{k+2} - t_1)) + 2\phi(t_2) \exp(-\lambda_1 (t_{k+2} - t_2)) \right] + \ldots + 4\phi(t_{k-1}) \exp(-\lambda_1 (t_{k+2} - t_{k-1})) + \phi(t_{k+2}) \] (5.9)

\[ Y_1(t_{k+2}) = \exp(-2\lambda_1 \Delta t) \times \left\{ \phi_0 \exp(-\lambda_1 t) + \frac{\lambda_1 \Delta t}{3} \left[ \phi(t_0) \exp(-\lambda_1 (t_{k+2} - t_0)) + 4\phi(t_1) \exp(-\lambda_1 (t_{k+2} - t_1)) + 2\phi(t_2) \exp(-\lambda_1 (t_{k+2} - t_2)) \right] + \ldots + 4\phi(t_{k-1}) \exp(-\lambda_1 (t_{k+2} - t_{k-1})) + \phi(t_k) \exp(-\lambda_1 (t_{k+2} - t_k)) \right\} \]

\[ + \frac{\lambda_1 \Delta t}{3} \left[ \phi(t_k) \exp(-\lambda_1 (t_{k+2} - t_k)) + 4\phi(t_{k+1}) \exp(-\lambda_1 (t_{k+2} - t_{k+1})) + \phi(t_{k+2}) \right] \] (5.10)

\[ Y_1(t_{k+2}) = Y_1(t_k) \exp(-2\lambda_1 \Delta t) + \frac{\lambda_1 \Delta t}{3} \left[ \phi(t_k) \exp(-2\lambda_1 \Delta t) + 4\phi(t_{k+1}) \exp(-\lambda_1 \Delta t) + \phi(t_{k+2}) \right] \] (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

\[
\text{CEX}_i = \exp(-2\lambda_i \Delta t) \quad (5.12)
\]
\[
\text{CEX2}_i = 4 \times \exp(-\lambda_i \Delta t) \quad (5.13)
\]
\[
\text{CLAM}_i = \frac{\lambda_i \Delta t}{3} \quad (5.14)
\]

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, \(A_i\).

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

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

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Register A, accumulator</td>
</tr>
<tr>
<td>B</td>
<td>Register B</td>
</tr>
<tr>
<td>C</td>
<td>Register C</td>
</tr>
<tr>
<td>D</td>
<td>Register D</td>
</tr>
<tr>
<td>E</td>
<td>Register E</td>
</tr>
<tr>
<td>H</td>
<td>Register H</td>
</tr>
<tr>
<td>L</td>
<td>Register L</td>
</tr>
<tr>
<td>HL</td>
<td>Register pair HL, usually an address with H = HI and L = LO</td>
</tr>
<tr>
<td>(HL)</td>
<td>The data located at the memory location in H and L</td>
</tr>
</tbody>
</table>

Labels

<table>
<thead>
<tr>
<th>Labels</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANT</td>
<td>The 16-bit address of MANT</td>
</tr>
<tr>
<td>(MANT)</td>
<td>The 8-bit value of MANT</td>
</tr>
</tbody>
</table>
Fig. 5.1A  Main Flowchart
Fig. 5.1 B Main Flowchart (cont.)
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**

<table>
<thead>
<tr>
<th>Octal Device Number</th>
<th>Peripheral Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>8095 Three-state buffer from Gray shaft encoder from Keithley</td>
</tr>
<tr>
<td>001</td>
<td>8095 Three-state buffer from tenths digit of the DPM</td>
</tr>
<tr>
<td>002</td>
<td>8095 Three-state buffer from ones digit of the DPM</td>
</tr>
<tr>
<td>003</td>
<td>8095 Three-state buffer from tens digit of the DPM</td>
</tr>
<tr>
<td>004</td>
<td>Numeric Display-latch for ones digit</td>
</tr>
<tr>
<td>005</td>
<td>Numeric Display-latch for tens digit</td>
</tr>
<tr>
<td>006</td>
<td>Numeric Display-latch for hundreds digit</td>
</tr>
<tr>
<td>007</td>
<td>Numeric Display-latch for thousands digit</td>
</tr>
<tr>
<td>010</td>
<td>Numeric Display-latch for ten-thousands digit</td>
</tr>
<tr>
<td>011</td>
<td>7475 flip-flop for sign</td>
</tr>
<tr>
<td>012</td>
<td>Four 7475 flip-flops that latch data from the Keithley and DPM</td>
</tr>
<tr>
<td>013</td>
<td>7475 flip-flop for 7490 decade counter</td>
</tr>
</tbody>
</table>
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. During 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,
HRFC: H1=COH
LO=85H

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 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 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
DIRECT: HI = 00H
LO = A6H

Fig. 5.6A  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 $C Ex_{1i}$, $\exp(-2\lambda_i\Delta t)$. 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 $C Ex_{2i}$, $4\exp(-\lambda_i\Delta t)$. The result of the multiplication appears in the floating-point-accumulator. HOLD, FLUX3, ($\phi(t)$), and $C L A M I$ ($\lambda_i\Delta t/3$) are each loaded as operands, and two additions and one multiplication are performed, respectively $[C L A M I * (FLUX1 * C Ex_{1i} + FLUX2 * C Ex_{2i} + FLUX3)]$. The final result of the above calculation is stored in HOLD while the multiplication of $Y_2Ti$, the previous neutron precursor flux, and $C Ex_{1i}$ is performed. HOLD is added to $Y_2Ti$ to obtain the new neutron precursor flux at time $t$. The result is stored in memory location $Y_2Ti$ 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 $A_i$,
PRECURI: HI = 14H
LO = 00H

Do i = 1, 6

PRECURI Routines

Fig. 5.7A
Fig. 5.7B  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 \( Y_{Ti} \). The \( Y_{Ti} \)'s are summed together in \( \text{REACT} \), which calculates the reactivity.

TRANFER

TRANFER (see Fig. 5.8 for flowchart) is a small routine that assigns the last flux measurement, \( \text{FLUX3} \), to \( \text{FLUX1} \) in preparation for a new calculation in each of the six \( \text{PRECURi} \) 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 \( \text{FLUX3} \) into the floating-point-accumulator and output it to \( \text{FLUX1} \).

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 \( a_iY_i(t) \), labeled in memory as \( Y_{Ti} \), \( Y_{T1} \) is loaded into the floating-point-accumulator and \( Y_{T2} \) is loaded into the operand. The floating-point \( \text{AD} \) subroutine is called, with the result of the operation appearing in the floating-point-accumulator. Similarly, \( Y_{T3}, Y_{T4}, Y_{T5}, \) and \( Y_{T6} \) 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 \( \text{FLUX3} \) is loaded and the floating-point \( \text{DIV} \) 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 $1 \times 10^7$ 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, $0258$ (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
OUTPUT: HI = 16 H
LO = 6D H

A ← (RHO + 9)
B ← FO H

Jump to CHK1

YES
A - B = 0

NO

OTPTO: A ← 0
E ← 1

OTNUM: OUT 4
Cali CHECK
OUT 5
Call CHECK

Fig. 5.10 A OUTPUT Routine
Fig. 5.10B  OUTPUT Routine (cont.)

CHKI: \[ A \leftarrow (RHO+1) \]

\[ B \leftarrow F E H \]

\[ K \]
Fig. 5.10 C  OUTPUT Routine (cont.)
Fig. 5.10 D OUTPUT Routine (cont)
Jump to OTPT2

\[ A - B = 0 \]

YES

NO

\[ HL \leftarrow RHO + 1 \]

\[ A \leftarrow (HL) \]

\[ E \leftarrow 1 \]

Jump to OTNUM

Fig. 5.10 E OUTPUT Routine (cont.)
OTPT5:  
\[ HL \leftarrow RHO + 5 \]
\[ A \leftarrow (HL) \]
\[ E \leftarrow 5 \]
Jump to OTNUM

OTPT4:  
\[ HL \leftarrow RHO + 4 \]
\[ A \leftarrow (HL) \]
\[ E \leftarrow 4 \]
Jump to OTNUM

Fig. 5.10F OUTPUT Routine (cont.)
OTPT 3:  
\[ \text{HL} \leftarrow \text{RHO + 3} \]  
\[ \text{A} \leftarrow (\text{HL}) \]  
\[ \text{E} \leftarrow 3 \]  
Jump to OTNUM

OTPT 2:  
\[ \text{HL} \leftarrow \text{RHO + 2} \]  
\[ \text{A} \leftarrow (\text{HL}) \]  
\[ \text{E} \leftarrow 2 \]  
Jump to OTNUM

Fig. 5.10G OUTPUT Routine (cont.)
Fig. 5.10 H OUTPUT Routine (cont.)
Table 5.5

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

<table>
<thead>
<tr>
<th>ASCII Code in Octal</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>002</td>
<td>2</td>
</tr>
<tr>
<td>003</td>
<td>3</td>
</tr>
<tr>
<td>004</td>
<td>4</td>
</tr>
<tr>
<td>005</td>
<td>5</td>
</tr>
<tr>
<td>006</td>
<td>6</td>
</tr>
<tr>
<td>007</td>
<td>7</td>
</tr>
<tr>
<td>008</td>
<td>8</td>
</tr>
<tr>
<td>009</td>
<td>9</td>
</tr>
<tr>
<td>025</td>
<td>E</td>
</tr>
<tr>
<td>360</td>
<td>space</td>
</tr>
<tr>
<td>374</td>
<td>plus</td>
</tr>
<tr>
<td>375</td>
<td>minus</td>
</tr>
<tr>
<td>376</td>
<td>decimal point</td>
</tr>
</tbody>
</table>
Table 5.5

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

<table>
<thead>
<tr>
<th>Floating Point Number</th>
<th>13-Byte Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1.0 \times 10^{-3}$</td>
<td>360 001 376 000 000 000 000 000 000 025 375 000 003</td>
</tr>
<tr>
<td>$1.5 \times 10^{-2}$</td>
<td>360 001 376 005 000 000 000 000 000 025 375 000 002</td>
</tr>
<tr>
<td>$2.75 \times 10^{-1}$</td>
<td>360 376 002 007 005 000 000 000 000 030 360 360 360</td>
</tr>
<tr>
<td>$4.5 \times 10^{0}$</td>
<td>360 004 376 005 000 000 000 000 000 000 360 360 360</td>
</tr>
<tr>
<td>$-4.5 \times 10^{0}$</td>
<td>375 004 376 005 000 000 000 000 000 000 360 360 360</td>
</tr>
<tr>
<td>$1.2 \times 10^{1}$</td>
<td>360 001 002 376 000 000 000 000 000 000 360 360 360</td>
</tr>
<tr>
<td>$2.9 \times 10^{2}$</td>
<td>360 002 011 000 376 000 000 000 000 000 360 360 360</td>
</tr>
<tr>
<td>$5.85 \times 10^{3}$</td>
<td>360 005 010 005 000 376 000 000 000 000 360 360 360</td>
</tr>
<tr>
<td>$6.789 \times 10^{4}$</td>
<td>360 006 007 010 011 000 376 000 000 360 360 360</td>
</tr>
<tr>
<td>$8.0 \times 10^{5}$</td>
<td>360 010 000 000 000 000 376 000 360 360 360</td>
</tr>
<tr>
<td>$1.5 \times 10^{6}$</td>
<td>360 001 005 000 000 000 000 000 376 360 360 360</td>
</tr>
<tr>
<td>$1.75 \times 10^{7}$</td>
<td>360 001 376 007 005 000 000 000 000 025 374 000 007</td>
</tr>
<tr>
<td>$1.05 \times 10^{12}$</td>
<td>360 001 376 000 005 000 000 000 000 025 374 001 002</td>
</tr>
</tbody>
</table>
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, 025₈ 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

<table>
<thead>
<tr>
<th>Decimal Point Location</th>
<th>Location of Decimal Digit</th>
<th>Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHO + 6</td>
<td>RHO + 5</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>RHO + 4</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>RHO + 3</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>RHO + 2</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>RHO + 1</td>
<td>Ten-thousands</td>
</tr>
<tr>
<td>RHO + 5</td>
<td>RHO + 4</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>RHO + 3</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>RHO + 2</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>RHO + 1</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Ten-thousands</td>
</tr>
<tr>
<td>RHO + 4</td>
<td>RHO + 3</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>RHO + 2</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>RHO + 1</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Ten-thousands</td>
</tr>
<tr>
<td>RHO + 3</td>
<td>RHO + 2</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>RHO + 1</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Ten-thousands</td>
</tr>
<tr>
<td>RHO + 2</td>
<td>RHO + 1</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Ten-thousands</td>
</tr>
<tr>
<td>RHO + 1</td>
<td>Zero</td>
<td>Ones</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Tens</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Hundreds</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Thousands</td>
</tr>
<tr>
<td></td>
<td>Blank</td>
<td>Ten-thousands</td>
</tr>
</tbody>
</table>
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 = 0008 and LO = 0508, which is the starting address of RESTR.
After clearing the decade counter, the routine INPUT regains control.
WAIT: HI = 00H  
   LO = 5CH

   Enable Interrupt

   No Operation

   Jump to WAIT

RESTART: HI = 00H  
   LO = 28H

   OUT 11

   Go to INPUT

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 DMAwT, 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
START: HI=00H
LO=00H

SP ← 1100H

HL ← DMANT

(HL) ← FOH

L ← L + 3

(HL) ← FEH

L ← L + 2

(HL) ← FFH

D ← 7

HL ← Y2TI

J

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.
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

<table>
<thead>
<tr>
<th>$\rho_{\text{act}}$ (pcm)</th>
<th>T (sec)</th>
<th>% $\phi$ in. 0.2 sec</th>
<th>Reading 10% scale</th>
<th>Change of DPM in 0.4 sec</th>
<th>Reading 50% scale</th>
<th>Reading 90% scale</th>
<th>$\rho_{\text{calculated}}$ calculated 10% scale reading</th>
<th>$\rho_{\text{calculated}}$ calculated 50% scale reading</th>
<th>$\rho_{\text{calculated}}$ calculated 90% scale reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>337</td>
<td>5</td>
<td>4.1</td>
<td>0.8</td>
<td>4.0</td>
<td>7.4</td>
<td>335</td>
<td>334</td>
<td>336</td>
<td>336</td>
</tr>
<tr>
<td>211</td>
<td>15</td>
<td>1.3</td>
<td>0.2</td>
<td>1.3</td>
<td>2.3</td>
<td>180</td>
<td>215</td>
<td>205</td>
<td>205</td>
</tr>
<tr>
<td>144</td>
<td>30</td>
<td>0.7</td>
<td>0.1</td>
<td>0.7</td>
<td>1.2</td>
<td>118</td>
<td>148</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>104</td>
<td>50</td>
<td>0.4</td>
<td>0.1</td>
<td>0.4</td>
<td>0.7</td>
<td>0</td>
<td>103</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>78</td>
<td>75</td>
<td>0.3</td>
<td>0</td>
<td>0.3</td>
<td>0.5</td>
<td>0</td>
<td>83</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>60</td>
<td>100</td>
<td>0.2</td>
<td>0</td>
<td>0.2</td>
<td>0.3</td>
<td>0</td>
<td>62</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>53</td>
<td>125</td>
<td>0.16</td>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td>35</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>45</td>
<td>150</td>
<td>0.10</td>
<td></td>
<td>0.1</td>
<td>0.2</td>
<td></td>
<td>35</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>35</td>
<td>200</td>
<td>0.10</td>
<td></td>
<td>0</td>
<td>0.1</td>
<td></td>
<td>0</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>29</td>
<td>250</td>
<td>0.08</td>
<td></td>
<td>0.1</td>
<td></td>
<td>0</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>28</td>
<td>300</td>
<td>0.04</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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.

\[
\rho_0 = A \omega + \sum_{i=1}^{6} \frac{\beta_i}{\omega + \lambda_i} \tag{2.12}
\]

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,

\[
\rho_0 = \frac{A}{T} + \sum_{i=1}^{6} \frac{\beta_i}{1 + \lambda_i T} \tag{6.1}
\]

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 (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
INPUT: HI = 00 H
LO = 28 H

OUT 10

HL ← DMANT + 4

IN 1

(DMANT + 4) ← A

L ← L - 1

L ← L - 1

IN 2

(DMANT + 2) ← A

J

Fig. 6.6 A INPUT Routine
Fig. 6.6 B  INPUT Routine (cont.)

J

L ← L - 1

IN 3

(DMANT + 1) ← A

L ← L - 1

Call INP

HL ← FLX2

Call STR

Go to CMPAR
CMPAR HI = 00H
LO = 28H

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 0000010₂ and MEM equals 10001001₂, 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
EXCH: HI = 00H
LO = 72H

(FLXI) ← (FLX2)

(MEM) ← 0

(MEM1) ← 0

Return

Fig. 6.9 EXCH Subroutine
CALC: H1 = 14
LO = 00

Fig. 6. 10 A  CALC Routine
Fig. 6.10 B  CALC Routine (cont.)
Fig. 6.10 C  CALC Routine (cont.)
Fig. 6. I O D CALC Routine (cont.)
144

Call DIV

\[ N \]

\[ \text{HL} \leftarrow \text{RES} \]

Call AD

\[ \text{HL} \leftarrow \text{CONV} \]

Call MUL

\[ \text{HL} \leftarrow \text{RHO} \]

Call OU

\((\text{RES}) \leftarrow 0\)

Call EXCH

Jump to OUTPT

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 030g, (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 (λ₁, the decay constant) and PER (reactor period) are multiplied together. One is added to this quantity which is divided into BII (β₁, 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 * 10⁵. 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.

147
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 reactometer be based on the use of 8080 A/8085/Z-80 family of chips.
REFERENCES
REFERENCES


APPENDIX A

Calculation of $\beta_{\text{eff}}$

$\beta_{\text{eff}}$ 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 Reactirule sliderule. $\beta_{\text{eff}}$ is read off a scale. The following reactivity and period measurements, from which $\beta_{\text{eff}}$ was determined, were made by Robert Stone, Reactor Supervisor, on April 4, 1977.

<table>
<thead>
<tr>
<th>$\rho$ (per cent mills)</th>
<th>$T$ (seconds)</th>
<th>$\beta_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>179</td>
<td>20.4</td>
<td>.00680 ± .00001</td>
</tr>
<tr>
<td>170</td>
<td>48.0</td>
<td>.00682</td>
</tr>
<tr>
<td>244</td>
<td>11.1</td>
<td>.00676</td>
</tr>
</tbody>
</table>

To determine $\beta_{\text{eff}}$ for the reactor, the average of the three $\beta_{\text{eff}}$'s is used. The average value for $\beta_{\text{eff}}$ 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.
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:

<table>
<thead>
<tr>
<th>First byte</th>
<th>n</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>$\infty$</td>
<td>$2^{-\infty} = 0.00000000$</td>
</tr>
<tr>
<td>170</td>
<td>-8</td>
<td>$2^{-8} = 0.00390625$</td>
</tr>
<tr>
<td>171</td>
<td>-7</td>
<td>$2^{-7} = 0.0078125$</td>
</tr>
<tr>
<td>172</td>
<td>-6</td>
<td>$2^{-6} = 0.015625$</td>
</tr>
<tr>
<td>173</td>
<td>-5</td>
<td>$2^{-5} = 0.03125$</td>
</tr>
<tr>
<td>174</td>
<td>-4</td>
<td>$2^{-4} = 0.0625$</td>
</tr>
<tr>
<td>175</td>
<td>-3</td>
<td>$2^{-3} = 0.125$</td>
</tr>
<tr>
<td>176</td>
<td>-2</td>
<td>$2^{-2} = 0.25$</td>
</tr>
<tr>
<td>177</td>
<td>-1</td>
<td>$2^{-1} = 0.5$</td>
</tr>
<tr>
<td>200</td>
<td>0</td>
<td>$2^0 = 1$</td>
</tr>
<tr>
<td>201</td>
<td>1</td>
<td>$2^1 = 2$</td>
</tr>
</tbody>
</table>
Table 2.2: Powers of 2

<table>
<thead>
<tr>
<th>Number</th>
<th>Power</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$2^1$</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>$2^2$</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>$2^3$</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>$2^4$</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>$2^5$</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>$2^6$</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>$2^7$</td>
<td>256</td>
</tr>
</tbody>
</table>

The largest possible exponential value of $2^n$ is $2^{128} = 3.4028 \times 10^{38}$.

The smallest possible exponential value of $2^n$ is $2^{-128} = 2.9386 \times 10^{-39}$.

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\text{b} and 177\text{b}

the sign is positive. If the second byte is between

200\text{b} and 377\text{b}

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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Weighting Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>$1/2 = 0.5$ [always assumed to be present]</td>
</tr>
<tr>
<td>6</td>
<td>$1/4 = 0.25$</td>
</tr>
<tr>
<td>5</td>
<td>$1/8 = 0.125$</td>
</tr>
<tr>
<td>4</td>
<td>$1/16 = 0.0625$</td>
</tr>
<tr>
<td>3</td>
<td>$1/32 = 0.03125$</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Weighting Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1/512 = 0.001953125</td>
</tr>
<tr>
<td>6</td>
<td>1/1024 = 0.0009765625</td>
</tr>
<tr>
<td>5</td>
<td>1/2048 = 0.00048828125</td>
</tr>
<tr>
<td>4</td>
<td>1/4096</td>
</tr>
<tr>
<td>3</td>
<td>1/8192</td>
</tr>
<tr>
<td>2</td>
<td>1/16384</td>
</tr>
<tr>
<td>1</td>
<td>1/32768</td>
</tr>
<tr>
<td>0</td>
<td>1/65536</td>
</tr>
</tbody>
</table>

d. Fourth byte

More of same. Consider the following table:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Weighting factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1/131,072</td>
</tr>
<tr>
<td>6</td>
<td>1/262,144</td>
</tr>
</tbody>
</table>
The error inherent in the four-byte floating point representation is one part in 16,777,216, or 5.96 \times 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. 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 set to zero COUNT register

Since the number of counts is an integer that is greater than 0, the binary scaling factor should be 0408 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:

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

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, 377₈, 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:

<table>
<thead>
<tr>
<th>Input character string</th>
<th>Output character string</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.141593</td>
<td>3.141593</td>
</tr>
<tr>
<td>-.00000000000001</td>
<td>-1.0000000E-13</td>
</tr>
<tr>
<td>+1.6E5</td>
<td>160000.0</td>
</tr>
<tr>
<td>123456789</td>
<td>1.23456789E+08</td>
</tr>
<tr>
<td>54321E-10</td>
<td>5.432100E-06</td>
</tr>
<tr>
<td>-2718281828</td>
<td>-2.718281828E+09</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Floating point binary</th>
<th>Corresponding string representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>160 000 000 000</td>
<td>360 007 376 006 002 011 003 011 005 025 375 000 006</td>
</tr>
<tr>
<td>170 000 000 000</td>
<td>360 001 376 011 005 003 001 002 005 025 375 000 003</td>
</tr>
<tr>
<td>171 000 000 000</td>
<td>360 003 376 011 000 006 002 005 000 025 375 000 003</td>
</tr>
<tr>
<td>174 000 000 000</td>
<td>360 003 376 001 002 005 000 000 000 025 375 000 002</td>
</tr>
<tr>
<td>175 000 000 000</td>
<td>360 006 376 002 005 000 000 000 000 025 375 000 002</td>
</tr>
<tr>
<td>176 000 000 000</td>
<td>360 376 001 002 005 000 000 000 000 360 360 360 360</td>
</tr>
<tr>
<td>177 000 000 000</td>
<td>360 376 002 005 000 000 000 000 000 360 360 360 360</td>
</tr>
<tr>
<td>200 000 000 000</td>
<td>360 376 005 000 000 000 000 000 000 360 360 360 360</td>
</tr>
</tbody>
</table>
Many of the rules governing string character representations are evident in the listing on the preceding page. Note the following:

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

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

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.

\[
\begin{array}{ccc}
\text{FLOAT} & \text{FLOATQ} & \text{FLOAT} \\
202\ 000\ 000\ 000 & \times & 202\ 000\ 000\ 000 & = & 203\ 000\ 000\ 000 \\
202\ 100\ 000\ 000 & \times & 202\ 100\ 000\ 000 & = & 204\ 040\ 000\ 000 \\
203\ 000\ 000\ 000 & \times & 202\ 000\ 000\ 000 & = & 204\ 000\ 000\ 000 \\
204\ 000\ 000\ 000 & \times & 202\ 000\ 000\ 000 & = & 205\ 000\ 000\ 000 \\
201\ 000\ 000\ 000 & + & 201\ 000\ 000\ 000 & = & 202\ 000\ 000\ 000 \\
\end{array}
\]
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 $040_8$.

<table>
<thead>
<tr>
<th>Decimal number</th>
<th>FIXED</th>
<th>FLOAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>000 000 000 001 040</td>
<td>201 000 000 000</td>
</tr>
<tr>
<td>2.0</td>
<td>000 000 000 002 040</td>
<td>202 000 000 000</td>
</tr>
<tr>
<td>3.0</td>
<td>000 000 000 003 040</td>
<td>202 100 000 000</td>
</tr>
<tr>
<td>7.0</td>
<td>000 000 000 007 040</td>
<td>203 140 000 000</td>
</tr>
<tr>
<td>1.0</td>
<td>000 000 000 002 037</td>
<td>201 000 000 000 .</td>
</tr>
<tr>
<td>1.0</td>
<td>000 000 000 010 035</td>
<td>201 000 000 000</td>
</tr>
<tr>
<td>1.0</td>
<td>000 000 000 020 034</td>
<td>201 000 000 000</td>
</tr>
</tbody>
</table>
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:

- **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 arithmetic 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:

<table>
<thead>
<tr>
<th>Memory locations that refer to scratch pad memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hexadecimal</strong></td>
</tr>
<tr>
<td>02 35</td>
</tr>
<tr>
<td>02 47</td>
</tr>
<tr>
<td>02 52</td>
</tr>
<tr>
<td>02 5A</td>
</tr>
<tr>
<td>02 80</td>
</tr>
<tr>
<td>02 C6</td>
</tr>
<tr>
<td>02 CB</td>
</tr>
<tr>
<td>02 E1</td>
</tr>
<tr>
<td>03 9D</td>
</tr>
<tr>
<td>04 BF</td>
</tr>
<tr>
<td>04 DE</td>
</tr>
<tr>
<td>04 F5</td>
</tr>
<tr>
<td>05 08</td>
</tr>
<tr>
<td>05 17</td>
</tr>
<tr>
<td>05 6C</td>
</tr>
<tr>
<td>05 E6</td>
</tr>
<tr>
<td>06 3B</td>
</tr>
<tr>
<td>06 A1</td>
</tr>
<tr>
<td>06 C6</td>
</tr>
<tr>
<td>06 DA</td>
</tr>
<tr>
<td>06 E2</td>
</tr>
<tr>
<td>07 44</td>
</tr>
<tr>
<td>07 51</td>
</tr>
<tr>
<td>07 57</td>
</tr>
<tr>
<td>07 5D</td>
</tr>
<tr>
<td>07 79</td>
</tr>
<tr>
<td>07 7D</td>
</tr>
<tr>
<td>07 89</td>
</tr>
</tbody>
</table>
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.

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

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.

<table>
<thead>
<tr>
<th>LO memory address</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>MULX4</td>
<td>Operand third fraction</td>
</tr>
<tr>
<td>01</td>
<td>MULP3</td>
<td>Operand second fraction</td>
</tr>
<tr>
<td>05</td>
<td>MULP2</td>
<td>Operand first fraction</td>
</tr>
<tr>
<td>09</td>
<td>MULP1</td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>DIVX5</td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>OP4S</td>
<td>Divisor fourth fraction</td>
</tr>
<tr>
<td>11</td>
<td>OP3S</td>
<td>Divisor third fraction</td>
</tr>
<tr>
<td>15</td>
<td>OP2S</td>
<td>Divisor second fraction</td>
</tr>
<tr>
<td>19</td>
<td>OP1S</td>
<td>Divisor first fraction</td>
</tr>
<tr>
<td>1C</td>
<td>OP4A</td>
<td>Remainder fourth fraction</td>
</tr>
<tr>
<td>1E</td>
<td>DIVX6</td>
<td></td>
</tr>
<tr>
<td>1F</td>
<td>OP3A</td>
<td>Remainder third fraction</td>
</tr>
<tr>
<td>23</td>
<td>OP2A</td>
<td>Remainder second fraction</td>
</tr>
<tr>
<td>27</td>
<td>OP1A</td>
<td>Remainder first fraction</td>
</tr>
<tr>
<td>2A</td>
<td>OP4X</td>
<td>Remainder fourth fraction</td>
</tr>
<tr>
<td>2E</td>
<td>OVER</td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>PREX</td>
<td>Previous exponent</td>
</tr>
<tr>
<td>30</td>
<td>ACCE</td>
<td>Accumulator exponent</td>
</tr>
<tr>
<td>31</td>
<td>ACCS</td>
<td>Accumulator sign</td>
</tr>
<tr>
<td>32</td>
<td>ACC1</td>
<td>Accumulator first fraction</td>
</tr>
<tr>
<td>33</td>
<td>ACC2</td>
<td>Accumulator second fraction</td>
</tr>
<tr>
<td>34</td>
<td>ACC3</td>
<td>Accumulator third fraction</td>
</tr>
<tr>
<td>35</td>
<td>SF</td>
<td>Subtraction flag</td>
</tr>
<tr>
<td>36</td>
<td>ADPL</td>
<td>Character string word</td>
</tr>
<tr>
<td>37</td>
<td>ADRH</td>
<td>Character string bank</td>
</tr>
<tr>
<td>38</td>
<td>TMP1</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>39</td>
<td>TMP2</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>3A</td>
<td>TMP3</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>3B</td>
<td>VALE</td>
<td>Value exponent</td>
</tr>
<tr>
<td>3C</td>
<td>VAL1</td>
<td>Value first fraction</td>
</tr>
<tr>
<td>3D</td>
<td>VAL2</td>
<td>Value second fraction</td>
</tr>
<tr>
<td>3E</td>
<td>VAL3</td>
<td>Value third fraction</td>
</tr>
<tr>
<td>3F</td>
<td>TMP4</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>40</td>
<td>FSQRN</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>FMACX</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>FSQRX</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>FMACS</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>FMACT</td>
<td></td>
</tr>
<tr>
<td>4C</td>
<td>FMACG</td>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>FCSD</td>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>FLOGE</td>
<td></td>
</tr>
<tr>
<td>4E</td>
<td>FSNHD</td>
<td></td>
</tr>
</tbody>
</table>
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
COMPENSATED IONIZATION CHAMBER TYPE 6377

The 6377 compensated ionization chamber is designed to detect thermal neutrons in the range from $2.5 \times 10^2$ to $2.5 \times 10^{10}$ 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 II) 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 $4 \times 10^{-14}$ amperes neutron cm$^{-2}$ second. The gamma sensitivity, when operated uncompensated, is approximately $3 \times 10^{-11}$ amperes R hour.

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

**MECHANICAL**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Diameter</td>
<td>3-1/16 inches</td>
</tr>
<tr>
<td>Maximum Overall Length</td>
<td>23-1/2 inches</td>
</tr>
<tr>
<td>Approximate Sensitive Length</td>
<td>14 inches</td>
</tr>
<tr>
<td>Net Weight</td>
<td>5-3/8 pounds</td>
</tr>
<tr>
<td>Shipping Weight</td>
<td>19 pounds</td>
</tr>
</tbody>
</table>

**MATERIALS**

- Over Case: 3% Al, 97% Mg alloy
- Electrodes: 3% Al, 97% Mg alloy
- Insulation: Stabilized polystyrene
- Neutron Sensitive Material: Boron enriched in B-10
- Thickness: 1 mg/cm$^2$
- Gas Filling: Nitrogen

**IMPEDEANCE**

- Res. (Note 2). Signal Electrode to Case (Minimum): 10$^5$ Ohms
- Res. (Note 2). H.V. Electrode to Case (Minimum): 10$^5$ Ohms
- Res. (Compensating Electrode to Case (Minimum): 10$^5$ Ohms
- Conduction (Note 1). Signal Electrode to Case (Approx.) 275 $\mu$pf
- Conduction (Note 1). H.V. Electrode to Case (Approx.): 315 $\mu$pf
- Conduction (Note 1). Compensating Electrode to Case (Approx.): 125 $\mu$pf

**MAXIMUM RATINGS**

- Voltage Between Electrodes (exc.) 1500 max. Volts
- Temperature 175 max. Degrees F
- External Pressure (Note 3) 100 max. Pounds Inch$^{-2}$
- Thermal Neutron Flux 3 x 10$^{13}$ max. n$\cdot$cm$^{-2}$$\cdot$second$^{-1}$
TYPICAL OPERATION:

Typical Connection: See Figure 1
Operating Voltage: 300 to 800 Volts
Compensating Voltage: See Figure 3
Saturation Characteristics: See Figure 2
Thermal Neutron Plus
Range: 2.5 x 10^6 to 2.5 x 10^9 mV
Thermal Neutron Sensitivity: 4 x 10^-14 Amperes/mV
Gamma Sensitivity: Total Compensation: zero
Uncompensated: 3 x 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

Note: Permissible power supply regulation and ripple will depend upon the particular application. See Section entitled "Ionization Chamber Operation."
FIGURE 2

TYPICAL SATURATION CHARACTERISTICS

Signal Output in Milliamperes

Applied Positive Volts

2.4 x 10^9 nV

1.9 x 10^9 nV

1.2 x 10^9 nV

6 x 10^9 nV

3 x 10^9 nV

WESITNGHOUSE ELECTRIC CORPORATION, ELECTRONIC TUBE DIVISION, ELMIRA, NEW YORK
TYPICAL COMPENSATION CHARACTERISTICS

Positive Voltage
200 Volts

FIGURE 3

WESTINGHOUSE ELECTRIC CORPORATION, ELECTRONIC TUBE DIVISION, ELMIRA, NEW YORK
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>I</td>
</tr>
<tr>
<td>Description</td>
<td>II</td>
</tr>
<tr>
<td>Circuit Description</td>
<td></td>
</tr>
<tr>
<td>Calibration</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>III</td>
</tr>
<tr>
<td>Input Connections</td>
<td></td>
</tr>
<tr>
<td>Input Connections Using Direct Leads</td>
<td></td>
</tr>
<tr>
<td>Grid Current Recording</td>
<td></td>
</tr>
<tr>
<td>Speed of Response</td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td>IV</td>
</tr>
<tr>
<td>Maintenance Adjustments</td>
<td></td>
</tr>
<tr>
<td>Insulation Connector Caps</td>
<td></td>
</tr>
<tr>
<td>Special Instructions for the Model 411-C</td>
<td>V</td>
</tr>
</tbody>
</table>

# 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 amperes.

The features include full-scale voltage drop at the input of less than five millivolts, zero drift of less than 5% 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 10⁻¹ 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 $10 \times 10^{-4}$ amperes to $10 \times 10^{-12}$ amperes are selected by the Amperes Full Scale switch, located left of the Meter. The accuracy of the ranges from $10 \times 10^{-4}$ through $3 \times 10^{-7}$ is within 2%, $10 \times 10^{-9}$ through $10 \times 10^{-12}$ 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 $5 \times 10^{-14}$ amperes, 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 $3 \times 10^{-11}$ to $12 \times 10^{-10}$ 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 63% of final value)

<table>
<thead>
<tr>
<th>Ranges</th>
<th>with no significant external capacitance</th>
<th>with 5000 mmf across the input</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 10^{-11}$</td>
<td>2.0 Seconds</td>
<td>4.0 Seconds</td>
</tr>
<tr>
<td>$3 \times 10^{-11}$</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>$1 \times 10^{-10}$</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>$3 \times 10^{-10}$</td>
<td>0.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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 1175-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 $3 \times 10^{-6}$ to $10^{-11}$ amperes, the accuracy is better than 1%.

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 zeroing the meter if necessary.

INPUT, USING CABLES

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 33-756 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 ±0 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-202-3 is the chassis connector, P-202-CCT is the mating plug. Terminal #1 is ground.

KEITHLEY INSTRUMENTS CLEVELAND, OHIO
Table III gives resistance to be used in series with one and five milliamperes recording millimeters, to make the recorder full-scale deflection equal the panel meter full-scale deflection.

<table>
<thead>
<tr>
<th>Recorder</th>
<th>Series Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 m.a.</td>
<td>8.3 to 87 K</td>
</tr>
<tr>
<td>5 m.a.</td>
<td>1920 to 1240</td>
</tr>
</tbody>
</table>

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 5000 ohm divider will not draw too much output current and will be sufficiently low impedance to connect to amplifier inputs.

The Speed of Response, 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.

+216 Volts. 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 CB2 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.

KEITHLEY INSTRUMENTS

CLEVELAND, OHIO
SECTION IV MAINTENANCE

The Keithley Model 411 Micro-micammeter 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 \(10 \times 10^{-4}\) range and, with \(7 \times 10^{-4}\) amperes 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 2% of full scale. On the \(3 \times 10^{-4}\) to \(10 \times 10^{-3}\) amperes range the range resistors are accurate to 1% and, providing the calibration was accurately done on the \(10 \times 10^{-4}\) range, the overall accuracy will be 2%. From \(3 \times 10^{-3}\) to \(10 \times 10^{-12}\) 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 2Y5386-5 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.

- IV-1 -

KEITHLEY INSTRUMENTS

CLEVELAND, OHIO
ANALOG DEVICES

3½ Digit AC Line Powered DPM

General Description:
The AD2009 is a low cost 3½ digit AC line powered DPM designed for general purpose DPM applications. The AD2009 features bipolar input voltages over full scale ranges of either ±9.999V or ±19.9mV, with an accuracy of ±1% reading ±199.9mV. Full scale voltages over full range.

Display blanking is provided.

Applications:
General Purpose DPM Applications Requiring AC Power and a High Visibility Display

Data Logging and Digital Feedback Control Systems

Features:
AC Line Powered
Built for Reliability

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 inverts 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 seven segment displays, a separate power supply is provided to continually illuminate the seven segment display.

Design and manufacturing techniques are chosen to insure reliability in the AD2009. Consistent 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, the AD2009 is fully tested for electrical specifications, calibrated, and given one full week of failure free burn-in before shipment.

Food and Trigger Control Signals

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input Std

Input STD
**SPECIFICATIONS**

**DISPLAY OUTPUT**
- Beckman Seven Segment Gas Discharge Display. 0-55" high (144 mm) for three Data Digits. 100% Overrange and Negative Polarity indicating. 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: ±1.9992V or ±1999.9mV (see option)
- Auto-Polarity
- Input Impedance: 100MΩDC
- Bias Current, Both Range: InA @ 2V FS, 20mA @ 200mV FS
- Overvoltage Protection, Both Range: 200V DC Sustained

**ACCURACY**
- ±0.1% Digit
- Resolution: 1mV or 100µV (S option)
- Temperature Range: -25°C to +85°C Operating
- ±1°C to +85°C Storage
- Temperature Coefficient: Gain (Both Ranges) = ±0.003%/°C
- Zero Offset (1V Input) = ±1µV/°C
- (200mV Input) = ±2µV/°C
- Warm-Up Time to Rated Accuracy: 15 minutes
- Setting Time to Rated Accuracy: 0.3 sec

**NORMAL MODE REJECTION**
- 18dB @ 60Hz

**COMMON MODE REJECTION (1kHz source imbalance @ 50-60Hz with shielded shielded transformer)**
- 2V Input = 100dB
- 200mV Input = 80dB

**COMMON MODE VOLTAGE**
- 500VDC (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
  - Logic "0" <0.4V
  - Logic "1" >2.4V

**CONTROL INPUTS**
- Display Blank (1 TTL 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 (1 TTL Load). Logic "0" or grounding disables either the external or internal trigger and the last conversion is held and displayed.
- External Trigger (1 TTL Load). Positive pulse (10µsec max width) will initiate conversion.
- Decimal Points (Not TTL Compatible). Grounding will eliminate the desired decimal point. External drive circuitry must be capable of withstanding 100V when the decimal points are turned off.

**DATA OUTPUTS**
- JHCD Digits (Drives 6 TTL Loads): Positive true, unblanked.
- JHCD Digits (Drives 6 TTL Loads): Unblanked, Logic "0" indicates overrange (2000).
- Polarity (Drives 6 TTL Loads). Latched, Logic "1" indicates positive polarity
- Status (Drives 10 TTL 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 internal 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" x 1.93" (106 x 49 x 112mm)
- 225-21524-<>111 (117) or Equivalent

**WEIGHT**
- 15 ounces (425 grams)

**OPTIONS**
- ORDERING GUIDE
- AC Power Inputs (50-60Hz)
- AD2009 = 117VAC
- AD2009/6 = 220VAC
- AD2009/Q = 100VAC
- AD2009/H = 240VAC

- AD2009 = 1999VDC Full Scale
- AD2009/5 = 1999VDC 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
- 225-21524-<>111 (117) or Equivalent
- Optional: Order AC2611 or 54.50

**PRICING**
- $5.00 unit quantity

1Guaranteed to +25°C
2Guaranteed
3Not to be used when the AD2009 is floating on common mode voltages
4Use one input range and one power input may be specified
5Lens 7 is supplied in all less options as specified
6Specifications subject to change without notice
INTERFACING THE AD2009

Input Connections

The AD2009 has a single ended input with common analog and digital grounds. When digital control lines and DCI 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 ensure 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 1). Blankling the display has no effect on the output data of 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 TTL/74 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 8). For external triggering, a positive trigger pulse (≤500µs width) should be applied to the Trigger input (Pin 8). 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 paneling 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 warm up 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 ±1900V (+190.0mV on AD2009/5) and adjust the gain pot until the meter reads 1900 exactly.

Figure 1. AD2009 Timing Diagram

Figure 2. AD2009 Mechanical Outline

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

<table>
<thead>
<tr>
<th>PIN REF</th>
<th>PIN FUNCTION</th>
<th>PIN REF</th>
<th>PIN FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTERNAL TRIGGER OUT(^1)</td>
<td>A</td>
<td>NO CONNECTION</td>
</tr>
<tr>
<td>2</td>
<td>SIGNAL INPUT</td>
<td>B</td>
<td>EXTERNAL TRIGGER IN(^1)</td>
</tr>
<tr>
<td>3</td>
<td>STATUS (PRINT)</td>
<td>C</td>
<td>OVERLOAD</td>
</tr>
<tr>
<td>4</td>
<td>POLARITY</td>
<td>D</td>
<td>HOLD</td>
</tr>
<tr>
<td>5</td>
<td>BCD 8</td>
<td>E</td>
<td>BCD 1</td>
</tr>
<tr>
<td>6</td>
<td>BCD 2</td>
<td>F</td>
<td>BCD 4</td>
</tr>
<tr>
<td>7</td>
<td>BCD 80</td>
<td>G</td>
<td>BCD 10</td>
</tr>
<tr>
<td>8</td>
<td>BCD 20</td>
<td>H</td>
<td>BCD 40</td>
</tr>
<tr>
<td>9</td>
<td>BCD 800</td>
<td>I</td>
<td>BCD 100</td>
</tr>
<tr>
<td>10</td>
<td>SIGNAL GROUND</td>
<td>J</td>
<td>DP3/XXX</td>
</tr>
<tr>
<td>11</td>
<td>BCD 400</td>
<td>K</td>
<td>DP2/XX.X</td>
</tr>
<tr>
<td>12</td>
<td>BCD 200</td>
<td>L</td>
<td>DIGITAL GROUND</td>
</tr>
<tr>
<td>13</td>
<td>DISPLAY BLANK</td>
<td>M</td>
<td>DP1/XXX</td>
</tr>
<tr>
<td>14</td>
<td>OVERRIDE</td>
<td>N</td>
<td>SHIELD (EARTH GROUND)</td>
</tr>
<tr>
<td>15</td>
<td>AC LINE HI</td>
<td>O</td>
<td>AC LINE LO</td>
</tr>
</tbody>
</table>

\(^1\) 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
Maxmum Temperatures/Humidity
- 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

VCEO Collector to Emitter Sustaining Voltage: 20 V
Maxmum Current
- IC Collector Current: 25 mA

ELECTRICAL CHARACTERISTICS (25°C)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>CHARACTERISTICS</th>
<th>TYP</th>
<th>UNITS</th>
<th>TEST CONDITIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>lCEO</td>
<td>Collector Dark Current:Cell</td>
<td>4.0</td>
<td>nA</td>
<td>VCE = 5.0 V</td>
</tr>
<tr>
<td>lCE(L)</td>
<td>Photo Current</td>
<td>200</td>
<td>µA</td>
<td>VCE = 5.0 V, H = 5 mW/cm²</td>
</tr>
<tr>
<td>lCE(L)</td>
<td>Photo Current</td>
<td>1.75</td>
<td>mA</td>
<td>VCE = 5.0 V, H = 10 mW/cm²</td>
</tr>
<tr>
<td>lCE(L)</td>
<td>Photo Current</td>
<td>2.25</td>
<td>mA</td>
<td>VCE = 5.0 V, H = 5 mW/cm²</td>
</tr>
<tr>
<td>Smin</td>
<td>Matching Factor</td>
<td>0.5</td>
<td>—</td>
<td>VCE = 5.0 V</td>
</tr>
<tr>
<td>Smax</td>
<td>Photo Current</td>
<td>200</td>
<td>µA</td>
<td>VCE = 5.0 V, H = 5 mW/cm²</td>
</tr>
<tr>
<td>tr</td>
<td>Light Current Rise Time</td>
<td>4.0</td>
<td>µs</td>
<td>GaAs, IC = 2.0 mA</td>
</tr>
<tr>
<td>tf</td>
<td>Light Current Fall Time</td>
<td>—</td>
<td>—</td>
<td>RL = 100Ω, VCC = 5.0 V</td>
</tr>
<tr>
<td>VCE(sat)</td>
<td>Collector-Emitter Saturation Voltage</td>
<td>0.16</td>
<td>V</td>
<td>IC = 500 µA</td>
</tr>
<tr>
<td>VCEO(sus)</td>
<td>Collector-Emitter Sustaining Voltage</td>
<td>20</td>
<td>V</td>
<td>IC = 1.0 mA (Pulsed)</td>
</tr>
<tr>
<td>BVECO</td>
<td>Emitter-Collector Breakdown Voltage</td>
<td>7.0</td>
<td>V</td>
<td>IEC = 100 µA</td>
</tr>
</tbody>
</table>

*Planar is a patented Fairchild process

TYPICAL APPLICATION

![Typical Application Diagram](image-url)
APPENDIX D

Reactimeter Program
NEPRI MACRO CEX1I, CEX2I, CLAMI, Y2TI

LODE
LXI H, CEX1I ; LOAD H L REGISTER PAIR WITH ADDRESS OF CEX1I
CALL MUL ; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, HOLD ; LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD
STRIN
LXI H, FLUX2 ; LOAD H L REGISTER PAIR WITH ADDRESS OF FLUX2
LODE
LXI H, CEX2I ; LOAD H L REGISTER PAIR WITH ADDRESS OF CEX2I
CALL MUL ; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, HOLD ; LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD
CALL AD ; CALL IFPP AD SUBROUTINE TO ADD
LXI H, FLUX3 ; LOAD H L REGISTER PAIR WITH ADDRESS OF FLUX3
CALL AD ; CALL IFPP AD SUBROUTINE TO ADD
LXI H, CLAMI ; LOAD H L REGISTER PAIR WITH ADDRESS OF CLAMI
CALL MUL ; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, HOLD ; LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD
STRIN
LXI H, Y2TI ; LOAD H L REGISTER PAIR WITH ADDRESS OF Y2TI
;LOAD H L REGISTER PAIR WITH ADDRESS OF CEXII
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD
CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
LXI H,Y2TI ;LOAD H L REGISTER PAIR WITH ADDRESS OF Y2TI
STRI N ENDM
NEPR2 MACRO AI,YTI
LXI H, AI ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,YTI ;LOAD H L REGISTER PAIR WITH ADDRESS OF YTI
STRI N ENDM
LODE MACRO
CALL LOD ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR
ENDM

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 2

STRI N MACRO ;CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
GIVEN BY THE H L REGISTER PAIR

CALL IFPP AD SUBROUTINE TO AD
POP H
; POP THE HIGH AND LOW ADDRESS, STORED BY
; SELECT OFF THE STACK AND LOAD IN THE H L
; REGISTER PAIR
INX H
; INCREMENT THE H L REGISTER PAIR
INX H
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE LOW ADDRESS BYTE OF CONI
MOV E, M
; MOVE THE LOW ADDRESS BYTE OF CONI TO E
INX H
; INCREMENT H L REGISTER PAIR
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE HIGH ADDRESS BYTE OF CONI
MOV H, M
; MOVE THE HIGH ADDRESS BYTE OF CONI TO H
MOV L, E
; MOVE THE LOW ADDRESS BYTE OF CONI TO L
; THE H L REGISTER PAIR CONTAIN THE ADDRESS
; OF CONI
CALL MUL
; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, FLUX
; LOAD H L REGISTER PAIR WITH THE ADDRESS
; OF FLUX

; GIVEN BY THE H L REGISTER PAIR

CALL IFPP AD SUBROUTINE TO AD
POP H
; POP THE HIGH AND LOW ADDRESS, STORED BY
; SELECT OFF THE STACK AND LOAD IN THE H L
; REGISTER PAIR
INX H
; INCREMENT THE H L REGISTER PAIR
INX H
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE LOW ADDRESS BYTE OF CONI
MOV E, M
; MOVE THE LOW ADDRESS BYTE OF CONI TO E
INX H
; INCREMENT H L REGISTER PAIR
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE HIGH ADDRESS BYTE OF CONI
MOV H, M
; MOVE THE HIGH ADDRESS BYTE OF CONI TO H
MOV L, E
; MOVE THE LOW ADDRESS BYTE OF CONI TO L
; THE H L REGISTER PAIR CONTAIN THE ADDRESS
; OF CONI
CALL MUL
; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, FLUX
; LOAD H L REGISTER PAIR WITH THE ADDRESS
; OF FLUX

; GIVEN BY THE H L REGISTER PAIR

CALL IFPP AD SUBROUTINE TO AD
POP H
; POP THE HIGH AND LOW ADDRESS, STORED BY
; SELECT OFF THE STACK AND LOAD IN THE H L
; REGISTER PAIR
INX H
; INCREMENT THE H L REGISTER PAIR
INX H
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE LOW ADDRESS BYTE OF CONI
MOV E, M
; MOVE THE LOW ADDRESS BYTE OF CONI TO E
INX H
; INCREMENT H L REGISTER PAIR
; THE H L REGISTER PAIR CONTAIN THE MEMORY
; LOCATION OF THE HIGH ADDRESS BYTE OF CONI
MOV H, M
; MOVE THE HIGH ADDRESS BYTE OF CONI TO H
MOV L, E
; MOVE THE LOW ADDRESS BYTE OF CONI TO L
; THE H L REGISTER PAIR CONTAIN THE ADDRESS
; OF CONI
CALL MUL
; CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H, FLUX
; LOAD H L REGISTER PAIR WITH THE ADDRESS
; OF FLUX
THE FOLLOWING LABELS ARE FOR THE FLOATING POINT ROUTINES THAT ARE IN EPROM AND M ATICAL OPERATIONS. THESE ROUTINES ARE IN ANOTHER PRINT OUT AND ARE GIVEN HERE AS DUMMY PROGRAMS TO BE REFERENCE BY THE CROSS ASSEMBLER.

;THE FOLLOWING LABELS ARE FOR THE FLOATING POINT ROUTINES THAT ARE IN EPROM AND M ATICAL OPERATIONS. THESE ROUTINES ARE IN ANOTHER PRINT OUT AND ARE GIVEN HERE AS DUMMY PROGRAMS TO BE REFERENCE BY THE CROSS ASSEMBLER.

026E 00 LOD: NOP
026F C9 RET
028C 00 MUL: NOP
028D C9 RET
0284 00 DIV: NOP
0285 C9 RET
02D4 00 SB: NOP
02D5 C9 RET
02D7 00 AD: NOP
02D8 C9 RET
054A ORG 054AH

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 3

054A 00 INP: NOP
054B C9 RET
360C 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 M,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
0024  C35C00 JMP WAIT ;JUMP TO WAIT AND BEGIN PROGRAM
0028  INTRO: ORG 0028H
0028  D308  RESTR: OUT 11 ;GENERATE A DEVICE SELECT PULSE
002A  D30A  INPUT: OUT 10 ;GENERATE A DEVICE SELECT PULSE
           ;TO DEVICE TEM, 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  ;READ ONES DIGIT FROM DEVICE 2 INTO
0033 2D   DCR L  ;ACCUMULATOR
0034 DB02  IN 2
0036 77   MOV M,A
0037 2D   DCR L  ;READ TENS DIGIT FROM DEVICE 3 INTO
0038 DB03  IN 3  ;ACCUMULATOR
003A 77   MOV M,A  ;MOVE TENS DIGIT TO MEMORY ADDRGS BY H L

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 4

003B 0800  IN 0  ;REGISTER PAIR
003D 07   RLC  ;READ GRAY
003E 07   RLC  ;ROTATE ACCUMULATOR LEFT
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,MANT  ;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 REGISTERS WITH ADDRESS OF MANT
                      1  +
004C 1 CD3E02 + STRIN  ;CALL IFPP STR SUBROUTINE THAT STORES THE
                      1  +
                      1  +  ;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
                      +  ;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 LRFC OR LRFC AND THE CONSTANT CONI

0053  6E  MOV L,M ;MOVE KEITHLEY CODE INTO L
0054  2601  MVI H,001 ;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  03  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  C35COO  JMP WAIT ;JUMP TO WAIT
0061  21C610  LRFC: LXI H,MANT ;LOAD H L REGISTER PAIR WITH THE ADDRESS
;OF MANT

1  +  LODE ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
0064  1 CD6E02  +  CALL LOD ;POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
1  +  ;BY H L REGISTER PAIR
1  +
0067  216431  LXI H,CONT2 ;LOAD THE H L REGISTER PAIR WITH THE AD-
006A   C08C02   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

0070   1C0D702   +   CALL AD   ;CALL IFPP AD SUBROUTINE TO AD

0073   1E1     +   POP H   ;POP THE HIGH AND LOW ADDRESS, STORED BY
1        +   ;SELECT OFF THE STACK AND LOAD IN THE H L
1        +   ;REGISTER PAIR

0074   123     +   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   15E     +   MOV E,M   ;MOVE THE LOW ADDRESS BYTE OF CONI TO E

0076   123     +   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   166     +   MOV H,M   ;MOVE THE HIGH ADDRESS BYTE OF CONI TO H

0078   168     +   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   1C08C02   +   CALL MUL   ;CALL IFPP MUL SUBROUTINE TO MULTIPLY

007C   121CA10   +   LXI H,FLUX   ;LOAD H L REGISTER PAIR WITH THE ADDRESS
1        +   ;OF FLUX
2        +   STRIN

007F   2C03E02   +   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
0088 1 CD6E02 + CALL LOD ; Call IFPP LOD subroutine to load floating point accumulator from the address given by H L Register pair
008B 216801 LXI H, CONT7 ; Load the H L Register pair with the address 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
0094 1 C0D702 + CALL AD ; Call IFPP AD subroutine to AD
0097 1 E1 + POP H ; Pop the high and low address, stored by SELECT OFF the stack and load in the H L Register pair
0098 1 23 + INX H ; Increment the H L Register pair
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
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
; THE H L REGISTER PAIR CONTAIN THE ADDRESS
; 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 ; CALL IFPP STR SUBROUTINE THAT STORES THE
; 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 ; CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
; 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 ; CALL IFPP STRK SUBROUTINE THAT STORES THE
; 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 ; CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
; POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
1  +  
00C2  21A610  LXI H,FLUX3  ;LOAD H L REGISTER PAIR WITH THE ADDRESS OF FLUX3
1  +  
00C5  1C03E02  STR   CALL STR  ;CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR
1  +  
00CB  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  C0DF00  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
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1403</td>
<td>2 C6E02</td>
<td>CALL LOD: CALL IFPP LOD SUBROUTINE TO LOAD FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR</td>
</tr>
<tr>
<td>1406</td>
<td>1 216C01</td>
<td>LXI H,CEXI1: LOAD H L REGISTER PAIR WITH ADDRESS OF CEXI1</td>
</tr>
<tr>
<td>1409</td>
<td>1 C08C02</td>
<td>CALL MUL: CALL IFPP MUL SUBROUTINE TO MULTIPLY</td>
</tr>
<tr>
<td>140C</td>
<td>1 21C210</td>
<td>LXI H,HOLD: LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD</td>
</tr>
<tr>
<td>140F</td>
<td>2 CD3E02</td>
<td>CALL STR: CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR</td>
</tr>
<tr>
<td>1412</td>
<td>1 21A210</td>
<td>LXI H,FLUX2: LOAD H L REGISTER PAIR WITH ADDRESS OF FLUX2</td>
</tr>
<tr>
<td>1415</td>
<td>2 C6E02</td>
<td>CALL LOD: CALL IFPP LOD SUBROUTINE TO LOAD FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR</td>
</tr>
<tr>
<td>1418</td>
<td>1 218401</td>
<td>LXI H,CEX21: LOAD H L REGISTER PAIR WITH ADDRESS OF CEX21</td>
</tr>
<tr>
<td>141B</td>
<td>1 C08C02</td>
<td>CALL MUL: CALL IFPP MUL SUBROUTINE TO MULTIPLY</td>
</tr>
<tr>
<td>141E</td>
<td>1 21C210</td>
<td>LXI H,HOLD: LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD</td>
</tr>
<tr>
<td>1421</td>
<td>1 CD702</td>
<td>CALL AD: CALL IFPP AD SUBROUTINE TO ADD</td>
</tr>
<tr>
<td>1424</td>
<td>1 21A610</td>
<td>LXI H,FLUX3: LOAD H L REGISTER PAIR WITH ADDRESS OF FLUX3</td>
</tr>
<tr>
<td>1427</td>
<td>1 CD702</td>
<td>CALL AD: CALL IFPP AD SUBROUTINE TO ADD</td>
</tr>
<tr>
<td>142A</td>
<td>1 219C01</td>
<td>LXI H,CLAM1: LOAD H L REGISTER PAIR WITH ADDRESS OF CLAM1</td>
</tr>
</tbody>
</table>
CALL MUL
;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD
;LOAD H L REGISTER PAIR WITH ADDRESS OF
HOLD
2
STRIN
;CALL IFPP STR SUBROUTINE THAT STORES THE
FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
GIVEN BY THE H L REGISTER PAIR
LXI H,Y2T1
;LOAD H L REGISTER PAIR WITH ADDRESS OF
Y2TI
2
LODE
;CALL IFPP LOO SUBROUTINE TO LOAD FLOATING
-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
BY H L REGISTER PAIR
LXI H,CEx11
;LOAD H L REGISTER PAIR WITH ADDRESS OF
CEx11
CALL MUL
;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD
;LOAD H L REGISTER PAIR WITH ADDRESS OF
HOLD
CALL AD
;CALL IFPP AD SUBROUTINE TO ADD
LXI H,Y2T1
;LOAD H L REGISTER PAIR WITH ADDRESS OF
Y2TI
2
STRIN
;CALL IFPP STR SUBROUTINE THAT STORES THE
FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
GIVEN BY THE H L REGISTER PAIR
NEPR2 A1,YT1
LXI H, AI ; Load H L register pair with address of AI
CALL MUL ; Call IFPP MUL subroutine to multiply
LXI H, YTI ; Load H L register pair with address of YTI
+ STRIN ; Call IFPP STR subroutine that stores the floating-point-accumulator in the address given by the H L register pair
LXI H, FLUX2 ; Load H L register pair with address of FLUX1
+ NEPR1 CEX12, CEX22, CLAM2, Y2T2
+ LODR ; Load floating-address given by the H L register pair
CALL LOD ; Call IFPP LOD subroutine to load floating-point-accumulator from the address given by the H L register pair
LXI H, CEX12 ; Load H L register pair with address of CEX11
CALL MUL ; Call IFPP MUL subroutine to multiply
LXI H, HOLD ; Load H L register pair with address of HOLD
+ STRIN ; Call IFPP STR subroutine that stores the floating-point-accumulator in the address given by the H L register pair
LXI H, FLUX2 ; Load H L register pair with address of FLUX2
+ LODR ; Load floating-address given by the H L register pair
CALL LOD ; Call IFPP LOD subroutine to load floating-point-accumulator from the address given by the H L register pair
LXI H, CEX22 ; Load H L register pair with address of CEX21
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1 ;HOLD
CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
LXI H,FLUX3 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
FLUX3

CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
LXI H,CLAM2 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
CLAMI
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF
HOLD
STRI ;CALL IFPP STR SUBROUTINE THAT STORES THE
FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
GIVEN BY THE H L REGISTER PAIR
LXI H,Y2TI ;LOAD H L REGISTER PAIR WITH ADDRESS OF
Y2TI
LODE ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
BY H L REGISTER PAIR
LXI H,CFX12 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
CFX12
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF
CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
LXI H,Y2T2 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
;Y2T2
CALL STR ;CALL IFPP STR SUBROUTINE THAT STORES THE
;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
;GIVEN BY THE H L REGISTER PAIR
NEPR2 A2,YT2
LXI H,A2 ;LOAD H L REGISTER PAIR WITH ADDRESS OF AI
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,YT2 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
;YTI
CALL STR ;CALL IFPP STR SUBROUTINE THAT STORES THE
;FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
;GIVEN BY THE H L REGISTER PAIR
LXI H,FLUX1 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
;FLUX1
NEPR1 CEX13,CEX23,CLAM3,Y2T3
LODE ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
;POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
;BY H L REGISTER PAIR
LXI H,CEX13 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
;CEX11
CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF
;HOLD
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  + ;LOAD
14C9 2 C06E02 + 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  + ;CEX23
14CF 1 C08C02 + 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 C0D702 + 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 C0D702 + CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
14DE 1 21A401 + LXI H,CLAM3 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1  + ;CLAM3
14E1 1 C08C02 + CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14E4 1 21C210 + LXI H,HOLD ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1  + ;HOLD
14E7 2 C03E02 + 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 + LOAD ;Y2TI
2 + CALL LOD ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
14ED 2 CD6E02 + POINT-ACCUMULATOR FROM THE ADDRESS GIVEN
14F0 2 217401 + BY H L REGISTER PAIR
14F3 1 CD8C02 + CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
14F6 1 21C210 + LXI H,CEX13 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
14F9 1 CDD702 + ;CSEXII
14FC 1 218E10 + CALL AD ;CALL IFPP AD SUBROUTINE TO ADD
14FF 2 CD3E02 + LXI H,Y2T3 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
14FF 2 + Y2TI ;Y2TI
1 + STRIN ;CALL IFPP STR SUBROUTINE THAT STORES THE
2 + FLOATING-POINT-ACCUMULATOR IN THE ADDRESS
1 + GIVEN BY THE H L REGISTER PAIR
1502 1 21BC01 + LXI H,A3 ;LOAD H L REGISTER PAIR WITH ADDRESS OF A
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
150A 2 CD3E02 + YTI ;YTI
2 + STRIN ;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 + NEPR1 CEX14,CEX24,CLAM4,Y2T4
          2 + LODE
1511 2 C06E02 + 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 + ;CEX14
1517 1 C08C02 + 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 C03E02 + 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 C06E02 + 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 + ;CEX24
1529 1 C08C02 + 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 C0D702 + 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 C0D702 + 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          ;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          +          LODD          ;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,CFX11         ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1          +          ;CFX11
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       CD0702  +          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          ;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 208C02 + CALL MUL ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1562  1 21B610 + LXI H,YT4 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
2 + YT1
2 + STRIN
1565  2 2D3E02 + 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  2 219E10 PREC5: LXI H,FLUX1 ;LOAD H L REGISTER PAIR WITH ADDRESS OF
1 + ;FLUX1
2 + NEPR1 CEX15,CEX25,CLAM5,YZT5
2 + LODE
156B  2 2D6E02 + 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 + ;CEX15
1571  1 208C02 + 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 2D3E02 + 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 2D6E02 + 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
<table>
<thead>
<tr>
<th>Line</th>
<th>Opcode 1</th>
<th>Opcode 2</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1583</td>
<td>C08C02</td>
<td>CALL MUL</td>
</tr>
<tr>
<td>1</td>
<td>1586</td>
<td>21C210</td>
<td>LXI H, HOLD</td>
</tr>
<tr>
<td>1</td>
<td>1589</td>
<td>CDD702</td>
<td>CALL AD</td>
</tr>
<tr>
<td>1</td>
<td>158C</td>
<td>21A610</td>
<td>LXI H, FLUX3</td>
</tr>
<tr>
<td>1</td>
<td>158F</td>
<td>CDD702</td>
<td>CALL AD</td>
</tr>
<tr>
<td>1</td>
<td>1592</td>
<td>21AC31</td>
<td>LXI H, CLAM5</td>
</tr>
</tbody>
</table>

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 13

<table>
<thead>
<tr>
<th>Line</th>
<th>Opcode 1</th>
<th>Opcode 2</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1595</td>
<td>C08C02</td>
<td>CALL MUL</td>
</tr>
<tr>
<td>1</td>
<td>1598</td>
<td>21C210</td>
<td>LXI H, HOLD</td>
</tr>
<tr>
<td>1</td>
<td>1598</td>
<td>2CD3E02</td>
<td>STRIN</td>
</tr>
<tr>
<td>1</td>
<td>1598</td>
<td>2CD3E02</td>
<td>CALL STR</td>
</tr>
<tr>
<td>1</td>
<td>159E</td>
<td>219610</td>
<td>LXI H, Y2T5</td>
</tr>
<tr>
<td>1</td>
<td>15A1</td>
<td>2CD6E02</td>
<td>LOD</td>
</tr>
<tr>
<td>1</td>
<td>15A4</td>
<td>217C01</td>
<td>LXI H, CEX15</td>
</tr>
<tr>
<td>1</td>
<td>15A7</td>
<td>C08C02</td>
<td>CALL MUL</td>
</tr>
</tbody>
</table>
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
15BO 1 219610 + LXI H, Y2T5 ; LOAD H L REGISTER PAIR WITH ADDRESS OF
2 + Y2T5
2 + STRIN
15B3 2 CD3E02 + CALL STR ; CALL IFPP STR SUBROUTINE THAT STORES THE
2 + FLOATING-POINTER-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 + YT5
2 + STRIN
15BF 2 CD3E02 + CALL STR ; CALL IFPP STR SUBROUTINE THAT STORES THE
2 + FLOATING-POINTER-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
1 + FLUX1
2 + NEPR1 CEX16, CEX26, CLAM6, Y2T6
15C5 2 CD6E02 + CALL LOD ; CALL IFPP LOD SUBROUTINE TO LOAD FLOATING
2 + POINTER-ACCUMULATOR FROM THE ADDRESS GIVEN
2 + BY H L REGISTER PAIR
15CB 1 218001 + LXI H, CEX16 ; LOAD H L REGISTER PAIR WITH ADDRESS OF
1 + CEX16
15CB 1 CDB0C2 + 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
FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR

LOAD H L REGISTER PAIR WITH ADDRESS OF

2 + 2
15D4 1 21A210 + LXI H, FLUX2

1 + LODE
2 + CALL LOD
15D7 2 CD6E02 + CALL IFPP LOD SUBROUTINE TO LOAD FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR
2 + LXI H, CEX26
15DA 1 219801 + LOAD H L REGISTER PAIR WITH ADDRESS OF CEX26
1 +
15DD 1 CD8C02 + CALL MUL
15E0 1 21C210 + CALL IFPP MUL SUBROUTINE TO MULTIPLY
1 + LXI H, HOLD
15E6 1 21A610 + LOAD H L REGISTER PAIR WITH ADDRESS OF HOLD
1 + LXI H, FLUX3
15E9 1 CDD702 + LOAD H L REGISTER PAIR WITH ADDRESS OF FLUX3
1 +
15EC 1 21B001 + CALL AD
15EF 1 CD8C02 + CALL IFPP AD SUBROUTINE TO ADD
1 + LXI H, CLAM6
15F2 1 21C210 + LOAD H L REGISTER PAIR WITH ADDRESS OF CLAM6
1 +
15F5 2 CO3E02 + STRIN
2 + CALL STR
2 + CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR
15F8 1 219A10 + LXI H,Y2T6 ;LOAD H L REGISTER PAIR WITH ADDRESS OF Y2TI
1 +
2 + LODE
15F8 2 C06E02 + CALL LOD ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR
2 +
15FE 1 2180J1 + LXI H,CEX16 ;LOAD H L REGISTER PAIR WITH ADDRESS OF CEX11
1 +
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 HOLD
1 +
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 Y2TI
1 +
2 + STRIN
160D 2 C03E02 + CALL STR ;CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR
2 +
1610 1 21C801 + LXI H,A6 ;LOAD H L REGISTER PAIR WITH ADDRESS OF A6
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 YT1
1 +
2 + STRIN
1619 2 CD3E02 + CALL STR ;CALL IFPP STR SUBROUTINE THAT STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR
2 +
161C 21A610 TRANS: LXI H,FLUX3 ;LOAD H L REGISTER PAIR WITH THE ADDRESS

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 15
1 + LODE
161F 1 CD6E02 + CALL LOD ;CALL IFPP LOD SUBROUTINE TO LOAD FLOATING POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR
1 +
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 FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY THE H L REGISTER PAIR
1 +
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 POINT-ACCUMULATOR FROM THE ADDRESS GIVEN BY H L REGISTER PAIR
1 +
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
1649  C0D702  CALL AD  ;CALL IFPP AD SUBROUTINE TO ADD
164C  21A610  LXI H,FLUX3  ;LOAD H L REGISTER PAIR WITH THE ADDRESS
       OF FLUX3
164F  C0D402  CALL DIV  ;CALL IFPP DIV SUBROUTINE TO DIVIDE
1652  C04D02  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  C0D702  CALL AD  ;CALL IFPP AD SUBROUTINE TO ADD
165B  21D401  LXI H,TEN5  ;LOAD H L REGISTER PAIR WITH THE ADDRESS
       OF TEN5
165E  C08C02  CALL MUL  ;CALL IFPP MUL SUBROUTINE TO MULTIPLY
1661  214001  LXI H,OFSET  ;LOAD H L REGISTER PAIR WITH ADDRESS OF
       OFSET
1664  C0D702  CALL AD  ;CALL IFPP AD SUBROUTINE TO ADD
1667  210010  LXI H,RHO  ;LOAD H L REGISTER PAIR WITH THE ADDRESS
       OF RHO
166A  C0DC06  CALL OU  ;CALL THE IFPP OU SUBROUTINE THAT STORES
       THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS BY THE H L REGISTER PAIR IN BCD

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 16

166D  3AD910  OUTPT: LDA RHO+9  ;FORMAT
       ;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
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
1673 CA9316 JZ CHK1 ; JUMP TO CHK1 IF THE RESULT IS ZERO
1676 3E00 OTPT0: 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 COD800 CALL CHECK ; CALL SUBROUTINE CHECK
167F D305 OUT 5 ; OUTPUT ACCUMULATOR TO DEVICE 5
1681 COD800 CALL CHECK ; CALL SUBROUTINE CHECK
1684 D306 OUT 6 ; OUTPUT ACCUMULATOR TO DEVICE 6
1686 COD800 CALL CHECK ; CALL SUBROUTINE CHECK
1689 D307 OUT 7 ; OUTPUT ACCUMULATOR TO DEVICE 7
168B COD800 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 OTPT0 ; JUMP TO OTPT0
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
<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16B4</td>
<td>CAOB16</td>
<td>JZ OTPT3</td>
<td>Jump to OTPT3</td>
</tr>
<tr>
<td>16B7</td>
<td>3AD310</td>
<td>LDA RH0+3</td>
<td>Load accumulator with RH0+3</td>
</tr>
<tr>
<td>16BA</td>
<td>06FE</td>
<td>MVI B,376Q</td>
<td>Load B with 376Q (Code for decimal point)</td>
</tr>
<tr>
<td>16BC</td>
<td>B8</td>
<td>CMP B</td>
<td>Compare B with accumulator</td>
</tr>
<tr>
<td>16BD</td>
<td>CAE416</td>
<td>JZ OTPT2</td>
<td>Jump to OTPT2</td>
</tr>
<tr>
<td>16C0</td>
<td>21D110</td>
<td>LXI H,RH0+1</td>
<td>Load H L register pair with RH0+1</td>
</tr>
<tr>
<td>16C3</td>
<td>7E</td>
<td>MOV A,M</td>
<td>Move the data from the memory location addressed by the H L register pair to the accumulator</td>
</tr>
<tr>
<td>16C4</td>
<td>1E01</td>
<td>MVI E,1</td>
<td>Set register equal to one</td>
</tr>
<tr>
<td>16C6</td>
<td>C37A16</td>
<td>JMP OTNUM</td>
<td>Jump to OTNUM</td>
</tr>
<tr>
<td>16C9</td>
<td>21D510</td>
<td>OTPT5: LXI H,RH0+5</td>
<td>Load H L register pair with RH0+5</td>
</tr>
<tr>
<td>16CC</td>
<td>7E</td>
<td>MOV A,M</td>
<td>Move the data from the memory location addressed by the H L register pair to the accumulator</td>
</tr>
</tbody>
</table>

8080 Macro Assembler, Ver 2.4
Errors = 0 Page 17

<table>
<thead>
<tr>
<th>Address</th>
<th>Opcode</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>16CD</td>
<td>1E05</td>
<td>MVI E,5</td>
<td>Set register equal to five</td>
</tr>
<tr>
<td>16CF</td>
<td>C37A16</td>
<td>JMP OTNUM</td>
<td>Jump to OTNUM</td>
</tr>
<tr>
<td>16D2</td>
<td>21D410</td>
<td>OTPT4: LXI H,RH0+4</td>
<td>Load H L register pair with RH0+4</td>
</tr>
<tr>
<td>16D5</td>
<td>7E</td>
<td>MOV A,M</td>
<td>Moving the data from the memory location addressed by the H L register pair to the accumulator</td>
</tr>
<tr>
<td>16D6</td>
<td>1E04</td>
<td>MVI E,4</td>
<td>Set register equal to four</td>
</tr>
<tr>
<td>16D8</td>
<td>C37A16</td>
<td>JMP OTNUM</td>
<td>Jump to OTNUM</td>
</tr>
<tr>
<td>16DB</td>
<td>21D310</td>
<td>OTPT3: LXI H,RH0+3</td>
<td>Load H L register pair with RH0+3</td>
</tr>
<tr>
<td>16DE</td>
<td>7E</td>
<td>MOV A,M</td>
<td>Set register equal to three</td>
</tr>
<tr>
<td>16DF</td>
<td>1E03</td>
<td>MVI E,3</td>
<td>Jump to OTNUM</td>
</tr>
<tr>
<td>16E1</td>
<td>C37A16</td>
<td>JMP OTNUM</td>
<td>Jump to OTNUM</td>
</tr>
<tr>
<td>16E4</td>
<td>21D210</td>
<td>OTPT2: LXI H,RH0+2</td>
<td>Load H L register pair with RH0+2</td>
</tr>
</tbody>
</table>
16E7 7E MOVL A,M
16E8 1E02 MVI E,2 ;SET REGISTER EQUAL TO E
16EA C37A16 JMP OTNUM ;JUMP TO OTNUM
16ED 3A0010 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 CA0000 JZ ZERO ;JUMP TO ZERO
16F6 3A901 LDA MINUS ;LOAD ACCUMULATOR WITH MINUS
16F9 0309 OUT 9 ;OUTPUT ACCUMULATOR TO DEVICE 9
16FB C35C00 JMP WAIT ;JUMP TO WAIT
1080 RWMEM: ORG 1080H
1080 0MANT: 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 ;TEMPORARY HOLDING SPACE FOR A MULTIPLICATION
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
10DD 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 4601 DW CON3 ;ADDRESS OF CON3
011C 8500 DW HRFC ;ADDRESS OF HRFC
011E 4601 DW CON3 ;ADDRESS OF CON3
0120 8500 DW HRFC ;ADDRESS OF HRFC
0122 6301 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 5001 DW CON7 ;ADDRESS OF CON7
<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>012C</td>
<td>8500</td>
<td>DW HRFC; ADDRESS OF HRFC</td>
</tr>
<tr>
<td>012E</td>
<td>5C01</td>
<td>DW CON7; ADDRESS OF CON7</td>
</tr>
<tr>
<td>0130</td>
<td>6100</td>
<td>DW LRFC; ADDRESS OF LRFC</td>
</tr>
<tr>
<td>0132</td>
<td>5401</td>
<td>DW CON5; ADDRESS OF CON5</td>
</tr>
<tr>
<td>0134</td>
<td>8500</td>
<td>DW HRFC; ADDRESS OF HRFC</td>
</tr>
<tr>
<td>0136</td>
<td>5401</td>
<td>DW CON5; ADDRESS OF CON5</td>
</tr>
<tr>
<td>0138</td>
<td>8500</td>
<td>DW HRFC; ADDRESS OF HRFC</td>
</tr>
<tr>
<td>013A</td>
<td>5801</td>
<td>DW CON6; ADDRESS OF CON6</td>
</tr>
<tr>
<td>013C</td>
<td>6100</td>
<td>DW LRFC; ADDRESS OF LRFC</td>
</tr>
<tr>
<td>013E</td>
<td>5801</td>
<td>DW CON6; ADDRESS OF CON6</td>
</tr>
<tr>
<td>0140</td>
<td>83</td>
<td>OFFSET: DB 203Q; NUMBER VALUE IS 5.66</td>
</tr>
<tr>
<td>0141</td>
<td>35</td>
<td>DB 065Q</td>
</tr>
<tr>
<td>0142</td>
<td>1E</td>
<td>DB 036Q</td>
</tr>
<tr>
<td>0143</td>
<td>88</td>
<td>DB 270Q</td>
</tr>
<tr>
<td>0144</td>
<td>88</td>
<td>CON1: DB 210Q; NUMBER VALUE IS 2.5000 * 10**2</td>
</tr>
<tr>
<td>0145</td>
<td>7A</td>
<td>DB 172Q</td>
</tr>
<tr>
<td>0146</td>
<td>00</td>
<td>DB 000Q</td>
</tr>
<tr>
<td>0147</td>
<td>00</td>
<td>DB 000Q</td>
</tr>
<tr>
<td>0148</td>
<td>8C</td>
<td>CON2: DB 214Q; NUMBER VALUE IS 2.5000 * 10**3</td>
</tr>
<tr>
<td>0149</td>
<td>1C</td>
<td>DB 034Q</td>
</tr>
<tr>
<td>014A</td>
<td>40</td>
<td>DB 100Q</td>
</tr>
<tr>
<td>014B</td>
<td>00</td>
<td>DB 000Q</td>
</tr>
<tr>
<td>014C</td>
<td>8F</td>
<td>CON3: DB 217Q; NUMBER VALUE IS 2.5000 * 10**4</td>
</tr>
<tr>
<td>014D</td>
<td>43</td>
<td>DB 103Q</td>
</tr>
<tr>
<td>014E</td>
<td>50</td>
<td>DB 120Q</td>
</tr>
<tr>
<td>014F</td>
<td>00</td>
<td>DB 000Q</td>
</tr>
<tr>
<td>0150</td>
<td>92</td>
<td>CON4: DB 222Q; NUMBER VALUE IS 2.5000 * 10**5</td>
</tr>
<tr>
<td>0151</td>
<td>74</td>
<td>DB 164Q</td>
</tr>
<tr>
<td>0152</td>
<td>24</td>
<td>DB 044Q</td>
</tr>
<tr>
<td>0153</td>
<td>00</td>
<td>DB 000Q</td>
</tr>
<tr>
<td>0154</td>
<td>96</td>
<td>CON5: DB 226Q; NUMBER VALUE IS 2.5000 * 10**6</td>
</tr>
<tr>
<td>0155</td>
<td>18</td>
<td>DB 030Q</td>
</tr>
</tbody>
</table>
8080 MACRO ASSEMBLER, VER 2.4  
ERRORS = 0  PAGE 19

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  07  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 203Q  ;NUMBER VALUE IS 0.9949329
016D  7E  DB 176Q
016E  B3  DB 263Q
016F  FD  DB 355Q
0170  80  CEX12: DB 200Q ;NUMBER VALUE IS 0.9874001
0171  7C  
0172  C6  
0173  43  
0174  80  CEX13: DB 200Q ;NUMBER VALUE IS 0.9550420
0175  74  
0176  7D  
0177  A1  
0178  80  CEX14: DB 200Q ;NUMBER VALUE IS 0.8830266
0179  62  
017A  0E  
017B  0A  
017C  80  CEX15: DB 200Q ;NUMBER VALUE IS 0.5712091
017D  12  
017E  3A  
017F  C2  
0180  7E  CEX16: DB 176Q ;NUMBER VALUE IS 0.2126729
0181  59  
0182  C6  
0183  ED  
0184  82  CEX21: DB 202Q ;NUMBER VALUE IS 3.989853
0185  7F  
0186  59  
0187  C0  
0188  82  CEX22: DB 202Q ;NUMBER VALUE IS 3.974720

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 20

0189  7E  
0190  176Q
CEX23: DB 202Q ;NUMBER VALUE IS 3.909050
CEX24: DB 202Q ;NUMBER VALUE IS 3.758780
CEX25: DB 202Q ;NUMBER VALUE IS 3.023135
CEX26: DB 201Q ;NUMBER VALUE IS 1.844659
CLAM1: DB 166Q ;NUMBER VALUE IS 8.466667 * 10**-4
CLAM2: DB 170Q ;NUMBER VALUE IS 2.113333 * 10**-3
CLAM3: DB 171Q ;NUMBER VALUE IS 7.666667 * 10**-3
CLAM4: DB 173Q ;NUMBER VALUE IS 2.073333 * 10**-2
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 0.026000
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  00  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

NO PROGRAM ERRORS

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0  PAGE 22

SYMBOL TABLE
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0007</td>
<td>A1</td>
<td>01B4</td>
<td>A2</td>
</tr>
<tr>
<td>A4</td>
<td>01C0</td>
<td>A5</td>
<td>01C4</td>
<td>A6</td>
</tr>
<tr>
<td>A7</td>
<td>AD</td>
<td>02B7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0000</td>
<td>BLANK</td>
<td>01D8</td>
<td>C</td>
</tr>
<tr>
<td>CEX12</td>
<td>0170</td>
<td>CEX13</td>
<td>0174</td>
<td>CEX14</td>
</tr>
<tr>
<td>CEX15</td>
<td>0188</td>
<td>CEX21</td>
<td>0184</td>
<td>CEX22</td>
</tr>
<tr>
<td>CEX23</td>
<td>018C</td>
<td>CEX24</td>
<td>0190</td>
<td>CEX25</td>
</tr>
<tr>
<td>CHK1</td>
<td>1693</td>
<td>CHS</td>
<td>024D</td>
<td>CLAM1</td>
</tr>
<tr>
<td>CLAM2</td>
<td>01A0</td>
<td>CLAM3</td>
<td>01A4</td>
<td>CLAM4</td>
</tr>
<tr>
<td>CON1</td>
<td>0144</td>
<td>CON2</td>
<td>0148</td>
<td>CON3</td>
</tr>
<tr>
<td>CON4</td>
<td>0150</td>
<td>CON5</td>
<td>0154</td>
<td>CON6</td>
</tr>
<tr>
<td>CON7</td>
<td>015C</td>
<td>CONT2</td>
<td>0164</td>
<td>CONT7</td>
</tr>
<tr>
<td>CONT8</td>
<td>0160</td>
<td>D</td>
<td>0002</td>
<td>DATA</td>
</tr>
<tr>
<td>CEX26</td>
<td>0198</td>
<td>CHECK</td>
<td>00D8</td>
<td></td>
</tr>
<tr>
<td>CON9</td>
<td>0160</td>
<td>E</td>
<td>0003</td>
<td></td>
</tr>
<tr>
<td>CON10</td>
<td>016C</td>
<td>F</td>
<td>0004</td>
<td></td>
</tr>
<tr>
<td>CON11</td>
<td>0170</td>
<td>G</td>
<td>0006</td>
<td></td>
</tr>
<tr>
<td>CON12</td>
<td>0174</td>
<td>GRAYC</td>
<td>00CE</td>
<td></td>
</tr>
<tr>
<td>INITZ</td>
<td>0018</td>
<td>INP</td>
<td>054A</td>
<td>INPUT</td>
</tr>
<tr>
<td>INT</td>
<td>022F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTRO</td>
<td>0027</td>
<td>L</td>
<td>0305</td>
<td>LOAD</td>
</tr>
<tr>
<td>LODIR</td>
<td>008C</td>
<td>LRFC</td>
<td>0061</td>
<td>M</td>
</tr>
<tr>
<td>MANT</td>
<td>10C6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM</td>
<td>10CF</td>
<td>MINUS</td>
<td>01D9</td>
<td>MUL</td>
</tr>
<tr>
<td>NEPR1</td>
<td>01CC</td>
<td>NEPR2</td>
<td>0292</td>
<td>NEUCC</td>
</tr>
<tr>
<td>OFSET</td>
<td>0140</td>
<td>ON</td>
<td>014C</td>
<td></td>
</tr>
<tr>
<td>OPTNUM</td>
<td>167A</td>
<td>OPTPT0</td>
<td>1676</td>
<td>OPTPT2</td>
</tr>
<tr>
<td>OPTPT3</td>
<td>16DB</td>
<td>OPTPT4</td>
<td>1602</td>
<td>OPTPT5</td>
</tr>
<tr>
<td>OUT</td>
<td>060C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTPT</td>
<td>166D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLUS</td>
<td>01DA</td>
<td>PREC1</td>
<td>1400</td>
<td>PREC2</td>
</tr>
<tr>
<td>PREC3</td>
<td>14B4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PREC4</td>
<td>150A</td>
<td>PREC5</td>
<td>1568</td>
<td>PREC6</td>
</tr>
<tr>
<td>PSEW</td>
<td>0006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REACT</td>
<td>1628</td>
<td>RESTR</td>
<td>0028</td>
<td>RHO</td>
</tr>
<tr>
<td>SB</td>
<td>02D4</td>
<td>SELECT</td>
<td>0050</td>
<td>SIGN</td>
</tr>
<tr>
<td>SP</td>
<td>0006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>063E</td>
<td>STR</td>
<td>023E</td>
<td>STRIN</td>
</tr>
<tr>
<td>TEN5</td>
<td>01D4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THREE</td>
<td>01D0</td>
<td>TRANS</td>
<td>161C</td>
<td>WAIT</td>
</tr>
<tr>
<td>WOUT</td>
<td>00DF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y2T1</td>
<td>1086</td>
<td>Y2T2</td>
<td>108A</td>
<td>Y2T3</td>
</tr>
<tr>
<td>Y2T4</td>
<td>1092</td>
<td>Y2T5</td>
<td>1096</td>
<td>Y2T6</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0200</td>
<td>JRG 200H</td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>ARTh</td>
<td>EQU § SHR 8 AND OFFH</td>
</tr>
<tr>
<td>0200</td>
<td>ARITH</td>
<td>EQU $</td>
</tr>
<tr>
<td>0100</td>
<td>SCR EQU</td>
<td>100H</td>
</tr>
<tr>
<td>0101</td>
<td>SCRb EQU</td>
<td>SCR SHR 8 AND OFFH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLANK NUMBER OF SCRATCHPAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8080 BINARY FLOATING POINT SYSTEM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARITHMETIC AND UTILITY PACKAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROGRAMMER CAL OHME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATE 26 DECEMBER 1973</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APITH IS THE BEGINNING ADDRESS OF THE ARITHMETIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AND UTILITY PACKAGE OF THE FLOATING POINT SYSTEM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCR IS THE BEGINNING ADDRESS OF THE RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USED AS SCRATCHPAD FOR THE SYSTEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>THE RAM MULTIPLY AND DIVIDE SUBROUTINES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ARE MOVED FROM ROM TO RAM BY SUBROUTINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT AND ARE EXECUTED IN RAM ONLY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAM MULTIPLY SUBROUTINE.</td>
</tr>
<tr>
<td>0100</td>
<td>MULX4 EQU</td>
<td>§-ARITH*SCR</td>
</tr>
<tr>
<td>0200</td>
<td>C609 ADI</td>
<td>0; ADDUPEAND 3RD FRACTION</td>
</tr>
<tr>
<td>0001</td>
<td>MULP3 EQU</td>
<td>§-1-ARITH</td>
</tr>
<tr>
<td>0202</td>
<td>5F MOV</td>
<td>E,A; 4TH PARTIAL PRODUCT</td>
</tr>
<tr>
<td>0203</td>
<td>7A MOV</td>
<td>A,D; 3RD PARTIAL PRODUCT</td>
</tr>
<tr>
<td>0204</td>
<td>C600 ACI</td>
<td>0; ADD UPERAND 2ND FRACTION</td>
</tr>
<tr>
<td>0005</td>
<td>MULP2 EQU</td>
<td>§-1-ARITH</td>
</tr>
<tr>
<td>0206</td>
<td>57 MOV</td>
<td>U,A; 3RD PARTIAL PRODUCT</td>
</tr>
<tr>
<td>0207</td>
<td>79 MOV</td>
<td>A,C; 2ND PARTIAL PRODUCT</td>
</tr>
</tbody>
</table>
0200 CE00 AC1 0; ADD OPERAND 1ST FRACTION
0009 MULP1 MULX5; TO ROM CODE
020A C37304 JMP EQU RAM DIVIDE SUBROUTINE;
0100 DIVX5 EQU $-1-ARITH
020D D63B SUI 0; SUB DIVISOR 4TH FRACTION
020E OP45 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 0F RET 0; RETURN TO ROM
011E DIVX6 EQU $-1-ARITH

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 2

021E C600 ADI 0; ADD DIVISOR 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 O; ADD DIVISOR 2ND FRACTION
0023 OP2A EQU $-1-ARITH
0224 67 MOV H,A; REMAINDER 2ND FRACTION
0225 7B MOV A,F; REMAINDER 1ST FRACTION
0226 CE00 ACI O; ADD DIVISOR 1ST FRACTION
0027 OP1A EQU $-1-APITH
0228 5F MOV E,A; REMAINDER 1ST FRACTION
0229 3E00 MVI A,0; REMAINDER 4TH FRACTION
002A OP4X EQU $-1-ARITH
022B C30F04 JMP DIVX2; TO ROM CODE
; RAM LOCATIONS USED BY THE BINARY
; FLOATING POINT SYSTEM
002E OVER EQU $-ARITH
022E 00 DB O; INITIALLY CLEAR
002F PREFIX EQU OVER+1; PREVIOUS EXPONENT
0030 ACCF EQU PREFIX+1; ACCUMULATOR EXPONENT
0031 ACCS EQU ACCF+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 262F INIT: MVI L,PREFIX; TO ADDR LAST WD TO MOVE
0231 2602 INIT1: MVI H,AKTH; TO ADDRESS RGM COPY
0233 5F MOV E,M; CURRENT WORD OF RGM 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
023A C9 RET ; RETURN TO CALLER
; SIX SUBROUTINE ENTRY POINT,
023C 73 STR0:  MOV M,E; STORE 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 79 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,SCRK; TO ADDRESS SCRATCH BLANK
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 AVI H,SCRK; TO ADDRESS SCRATCH BLANK
0253 2E31 MVI L,ACC;  TO ADDRESS ACCUM SIGN
0255 A6 ANA M;    COMPLEMENT OF SIGN
0256 2E80 XRI 200Q;  COMPLEMENT THE SIGN BIT
0258 77 MOV  M,A;  ACCUMULATOR SIGN
; FLOATING POINT TEST ENTRY POINT
0259 2601 TST: MVI H,SCRCL; TO ADDRESS SCRATCH BLANK
025B 2E30 TST1: MVI L,ACCL; TO ADDR OF ACCUM
025D 7E MOV A,M; ACCUMULATOR EXPONENT
025E A7 ANA A; SET CONTROL BITS
J25F 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 FRACTN
0267 2C INR L; TO ADDR ACCUM 2ND FRACTN
0268 4E MOV C,M; ACCUMULATOR 2ND FRACTN
0269 2C INR L; TO ADDR ACCUM 3RD FRACTN
026A 56 MOV D,M; ACCUMULATOR 3RD FRACTN
026B C37A03 JMP ADD12; TO SET EXIT CONDITIONS

; FLOATING POINT LOAD ENTRY POINT
026F 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 CP SIGN AND 1ST
0275 7E MOV A,M; OPERAND SIGN AND 1ST FRACTN
0276 2C INR L; TO ADDRESS OPERAND 2ND FRACTN
0277 4E MOV C,M; OPERAND 2ND FRACTN
0278 2C INR L; TO ADDRESS OPERAND 3RD FRACTN
0279 56 MOV D,M; OPERAND 3RD FRACTN

; STORE THE OPERAND IN THE ACCUMULATOR.
027A 6F MOV L,A; OPERAND SIGN AND 1ST FRACTN
J27B F08J LOD1: CKI ZRO; ACCUMULATOR 1ST FRACTN
027D 47 MOV B,A; ACCUMULATOR 1ST FRACTN
027E AD XRA L; ACCUMULATOR SIGN
027F 2601 MVI H,SCRBL; TO ADDRESS SCRATCH BLANK
0281 2F30 MVI L,ACCE; TO ADD 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 78 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 EN. PNT.
028C 7E MUL: MOV A,M; OPERAND EXPONENT
028D A7 ANA A; SET CONTROL BITS
028E C49503 CNZ MDEL; READ OPERAND IF NOT ZERO
0291 CA4602 JZ ZRC; IF ZERO OR UNDERFLOW
0294 D4AO2 JC OVERF; IF OVERFLOW
0297 CD4D04 CALL MULX; CALL FIXED MULT SUBRTP
; NORMALIZE IF NECESSARY.
029A 73 MOV A,B; 1ST PRODUCT
029B A7 ANA A; SET CONTROL
029C F4A902 JM ENDA; IF NO NORMALIZATION REQUIRED
029F 2E30 MVI L,ACCE; TO ADD ACCUM EXPONENT
02A1 7E MOV A,M; ACCUMULATOR EXPONENT
02A2 DE31 SBI 1; DECREMENT ACCUMULATOR EXPONENT
02A4 77 MOV M,A; ACCUMULATOR EXPONENT
02A5 68 RZ ; RETURN TO CALLER IS UNDERFLOW
02A6 CD8C03 CALL LSH; CALL LEFT SHIFT SUBROUTINE
ROUND IF NECESSARY.

02A9 CD3004 RND: CALL RND; CALL ROUNING SUBROUTINE
02AC DACA02 JC OVERF; IF OVERFLOW
02AF 47 MOV B,A; ACCUM SIGN AND 1ST FRACTION
02B0 F601 ORI I; 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 I; SET CARRY IF DIVISION BY ZERO
02B8 D49503 CMI; READ OPERAND IF NOT ZERO
02BA 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 C90004 CALL DIVX; CALL FIXED DIV SUBR TN
02C5 2601 MVI H,SCRB; TO ADDRESS SCRATCH BANK
02C7 DAA902 JC RND: CALL RND.

; SET OVERFLOW FLAG.
02CA 2601 OVERF: MVI H,SCRB; TO ADDRESS SCRATCH
02CC 2F2E MVI L,OVER; TO ADD OVERFLOW FLAG
02CE 3EFF MVI A,377Q; OVERFLOW FLAG
02D0 77 MOV M,A; OVERFLOW FLAG
02D1 07 RCL ; SET CARRY BIT FOR EXIT
02D2 C9 RET ; RETURN TO CALLER
02D3 0J DB O; 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.
0208  5E  MOV  E,M;  OPERAND EXPONENT
0209  2C  INR  L;  TO ADD UP SIGN, 1ST FRACTN

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 5

02DA  AE  XRA  N;  OPERAND SIGN AND 1ST FRACTN
02DB  47  MOV  B,A;  OPERAND SIGN AND 1ST FRACTN
02DC  2C  INR  L;  TO ADDR OPERAND 2ND
02DD  4E  MOV  C,M;  OPERAND 2ND FRACTION
02DE  2C  INR  L;  TO ADDR OPERAND 3RD FRACTN
02DF  56  MOV  D,A;  OPERAND 3RD FRACTION

; SAVE INITIAL EXPONENT.
02E0  2601  MVI  H,SCR8;  TO ADDRESS SCRATCH BANK
02E2  2E30  MVI  L,ACCE;  TO ADDR ACCUM EXPONENT
02E4  7E  MOV  A,N;  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
02EB  47  ANA  A;  SET CONTROL BITS
02E9  CA5602  JZ  TST1;  IF OPERAND IS ZERO

; GENERATE SUBTRACTION FLAG, RESTORE
; SUPPRESSED FRACTION BIT.
02EC  68  MOV  L,B;  OPERAND SIGN AND 1ST FRACTN
02ED  78  MOV  A,B;  OPERAND SIGN AND 1ST FRACTION
02EE  F680  ORI  2004;  OPERAND 1ST FRACTION
02F0  47  MOV  B,A;  OPERAND 1ST FRACTN
02F1  AD  XRA  L;  OPERAND SIGN
02F2  2E31  MVI  L,ACCS;  TO ADDRESS ACCUMULATOR SIGN
J2F4  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  DAGE03  JC  ADD2;  IF ACCUM SMALLER THAN OP
; CHECK FOR INSIGNIFICANT OPERAND
0303  FA5B02  JM  TST1;  IF THE OPERAND IS INSIGNIFICANT
0306  FE19  CPI  03LQ;  COMPARE SHIFT COUNT TO 25
0308  DA2D03  JC  ADD3;  JOIN EACH PATH IF OP SIGNIF
030B  C35B02  JMP  TST1;  OPERAND IS INSIGNIFICANT
; CHECK 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, ACCE;  TO ADDRESS THE ACCUMULATOR

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 6
; SIGN
0310 AE XRA M; OPERAND SIGN
031E 77 MOV M,A; ACCUMULATOR SIGN
031F AF XRA A; ZERO
0320 93 SUB E; COMPLEMENT 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 ADDR ACCUM 2ND FRACTION
0326 5E MOV E,M; ACCUMULATOR 2ND FRACTION
0327 71 MOV M,C; OPERAND 2ND FRACTION
0328 48 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,LF; TO ADDRESS SUBTRACTION FLAG
0332 7E MOV A,M; SUBTRACTION FLAG
0333 A7 ANA A; SET CONTROL BITS
0334 2E34 MVI L,ACC3; TO ADDR ACCUM 3RD FRACT
0336 FA5D03 JM ADD9; IF SUBTRACTION 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; AUGEND 2ND FRACTION
8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 7

033E 89  ADC C;  ADDEND 2ND FRACTION
033F 4F  MOV C,A;  SUB 2ND FRACTION
0340 20  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 FRACTN = LOW BIT OF 3RD
0350 5F  MOV F,A;  SUM 4TH FRACTION
0351 2E30  MVI L,ACCE;  TO ADDRESS ACCUMULATOR
  ;  EXPONENT

0353 7E  MOV A,M;  ACCUMULATOR EXPONENT
0354 C601  ADI 1;  INCREMENT ACCUMULATOR EXPONENT
0356 DACA02  JC OVERFLOW;  IF OVERFLOW
0359 71  MOV M,A;  ACCUMULATOR EXPONENT
035A C37403  JMP ADD11;  TO ROUND FRACTION
  ;  SUBTRACT SUBTRAHEND FROM MINUEND
; SUBTRACT MINUEND FROM SUBTRAHEND

035D 036E 036F 0370 0371 0372 0373 0374 0375 0376 0377 0378 0379 037A 037B 037C 037D 037E 037F 0380 0381 0382 0383 0384 0385 0386 0387 0388
AF 93 5F 7E 9A 57 20 7E 99 4F 20 7E 99 47 DCE03 E F40204 F24802 C03004 DAOA02 2E2F 7A 96 6F 7A F601 7B 5D C9
ADD9: SUB E, A; MOV E, A; MOV A, M; SUBB D; MOV D, A; MOV A, M; SUBB C; MOV C, A; MOV A, M; D pérdida: CP NORM; JP ZRCL; CALL RND; JC OVERF; MOv B, A; MOV L, PREX; MOV M; SUB L, A; MOV A, B; MOV A, E; MOV E, L; RET
XRA "SUBMINUEND 4TH FPCTN IS ZERO" SUBTRAHEND 4TH FRACTION DIFFERENCE 4TH FRACTION MINUEND 3RD FRACTION SUBTRAHEND 3RD FRACTION DIFFERENCE 3RD FRACTION TO ADDRESS MINUEND 2ND MINUEND 2ND FRACTION DIFFERENCE 2ND FRACTION SUBTRAHEND 2ND FRACTION MINUEND 1ST FRACTION DIFFERENCE 1ST FRACTION COMP; COMPLEMENT IF NEGATIVE NORMALIZED IF NECESSARY IF UNDERFLOW OR ZERO CALL RND; CALL ROUNDED SUBROUTINE IF OVERFLOW B, A; ACCUM SIGN AND 1ST FPCTN TO ADDRESS PREV EXPONENT ACCUMULATOR EXPONENT DIFFERENCE IN EXPONENTS ACCUM SIGN AND 1ST FPCTN SET SIGN BIT FOR EXIT ACCUMULATOR EXPONENT SIGNIFICANT INDEX

; LOAD ACCUMULATOR WITH THE OPERAND.

0386 0387
2E35 7E
ADD17: MVI L, SF; TO ADDR SUBTRACTION SUBTRACTION FLAG

248
0389  2E31  MVI  L,ACCS;  TO ADD ACCUMULATOR SIGN
0388  AE   XRA  M;    OPERAND SIGN
038C  2D   DCR  L;    TO ADDR ACCUM EXPONENT
038D  CD3C02 CALL STRO;  SET THE ACCUMULATOR
0390  A8   XRA  R;    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  2E   INR  L;    TO ADDR UP SIGN, 1ST FRCTN
0397  4E   MOV  C,M;  OPERAND SIGN AND 1ST FRACTION
0398  2C   INR  L;    TO ADDRESS OPERAND 2ND FRCTN

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 8

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  2E01 MOV  H,SRBB;  TO ADDRESS SCRATCH RANK
039E  2E3D MOV  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   KAR   ;    CARRY TO SIGN
03A6  A8   XRA  B;    CARRY AND SIGN MUST DIFFER
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>03A7 78</td>
<td>MOV A,B</td>
<td>RESULT EXPONENT PLUS BIAS</td>
<td></td>
</tr>
<tr>
<td>03A8 0680</td>
<td>MVI R,200Q</td>
<td>EXP BIAS, SIGN MASK, MS BIT</td>
<td></td>
</tr>
<tr>
<td>03AA F2B803</td>
<td>JP OVUN;</td>
<td>IF OVERFLOW OR UNDERFLOW</td>
<td></td>
</tr>
<tr>
<td>03AD 90</td>
<td>SUB B</td>
<td>REMOVE EXCESS EXP BIAS</td>
<td></td>
</tr>
<tr>
<td>03AE C8</td>
<td>RZ</td>
<td>RETURN IF UNDERFLOW</td>
<td></td>
</tr>
<tr>
<td>03AF 77</td>
<td>MOV M,A;</td>
<td>RESULT EXPONENT</td>
<td></td>
</tr>
<tr>
<td>03B0 2C</td>
<td>INR L</td>
<td>TO ADDRESS ACCUMULATOR SIGN</td>
<td></td>
</tr>
<tr>
<td>03B1 7E</td>
<td>MOV A,M;</td>
<td>ACCUMULATOR SIGN</td>
<td></td>
</tr>
<tr>
<td>03B2 A9</td>
<td>XRA C</td>
<td>RESULT SIGN IN SIGN BIT</td>
<td></td>
</tr>
<tr>
<td>03B3 A0</td>
<td>ANA B;</td>
<td>RESULT SIGN</td>
<td></td>
</tr>
<tr>
<td>03B4 77</td>
<td>MOV M,A;</td>
<td>RESULT SIGN</td>
<td></td>
</tr>
<tr>
<td>03B5 79</td>
<td>MOV A,C;</td>
<td>OPERAND SIGN AND 1ST FRCTN</td>
<td></td>
</tr>
<tr>
<td>03B6 B0</td>
<td>ORA B;</td>
<td>OPERAND 1ST FRACTION</td>
<td></td>
</tr>
<tr>
<td>03B7 C9</td>
<td>RET ;</td>
<td>RETURN TO CALLER</td>
<td></td>
</tr>
<tr>
<td>03B8 07</td>
<td>OVUN:RCL;</td>
<td>SET CARRY BIT IF OVERFLOW</td>
<td></td>
</tr>
<tr>
<td>03B9 D8</td>
<td>RC ;</td>
<td>RETURN IF OVERFLOW</td>
<td></td>
</tr>
<tr>
<td>03BA AF</td>
<td>XRA A;</td>
<td>ZERO</td>
<td></td>
</tr>
<tr>
<td>03BB C9</td>
<td>RET ;</td>
<td>RETURN IF UNDERFLOW</td>
<td></td>
</tr>
<tr>
<td>;</td>
<td>SUBROUTINE TO LEFT SHIFT THE B,C, D, AND E REGISTERS ONE BIT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03BC 74</td>
<td>LSH: MOV A,E;</td>
<td>ORIGINAL CONTENTS OF E</td>
<td></td>
</tr>
<tr>
<td>03BD 17</td>
<td>RAL ;</td>
<td>LEFT SHIFT E</td>
<td></td>
</tr>
<tr>
<td>03BE 5F</td>
<td>MOV E,A;</td>
<td>RESTORE CONTENTS OF REGISTER E</td>
<td></td>
</tr>
<tr>
<td>03BF 7A</td>
<td>LSH1: MOV A,D;</td>
<td>ORIGINAL CONTENTS OF D</td>
<td></td>
</tr>
<tr>
<td>; REGISTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03C0 17</td>
<td>RAL ;</td>
<td>LEFT SHIFT D</td>
<td></td>
</tr>
<tr>
<td>03C1 57</td>
<td>MOV D,A;</td>
<td>RESTORE CONTENTS OF D REGISTER</td>
<td></td>
</tr>
<tr>
<td>03C2 79</td>
<td>MOV A,C;</td>
<td>ORIGINAL CONTENTS OF C</td>
<td></td>
</tr>
<tr>
<td>; REGISTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03C3 17</td>
<td>RAL ;</td>
<td>LEFT SHIFT C</td>
<td></td>
</tr>
<tr>
<td>03C4 4F</td>
<td>MOV C,A;</td>
<td>RESTORE CONTENTS OF C REGISTER</td>
<td></td>
</tr>
<tr>
<td>03C5 78</td>
<td>MOV A,B;</td>
<td>ORIGINAL CONTENTS OF B</td>
<td></td>
</tr>
</tbody>
</table>
REG01; REGISTER

LEFT SHIFT B

RESTORE CONTENTS OF B REGISTER

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

OPERAND 4TH FRACTN IS ZERO
L, O10Q; EACH REG IS 8 BITS OF
SHIFT

L; COMPARE SHIFT COUNT TO 8

IF REG SHIFT LESS THAN 8

OPERAND 4TH FRACTION

OPERAND 3RD FRACTION

OPERAND 2ND FRACTION

OPERAND 1ST FRACTION IS ZERO

REDUCE SHIFT COUNT BY 1 REG

IF MORE SHIFTS REQUIRED

SHIFT OPERAND RIGHT BY - SHIFT COUNT-

BITS

A; SET CNTROL BITS

RETURN IF SHIFT IS COMPLETE

SHIFT COUNT

A; CLEAR CARRY BIT

OPERAND 1ST FRACTION
030F 1F    RAR    ;     RIGHT SHIFT OP 1ST FRACTN
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    DCR    L;     TO ADDR ACCUM SIGN
                  03F0 7E    MOV    A,M;     ACCUMULATOR SIGN
                  03F1 EF80  XRI    2000;     CHANGE SIGN
                  03F3 77    MOV    M,A;     ACCUMULATOR SIGN
                  03F4 AF    COMPL:  XRA    A;     ZER0
                  03F5 6F    MOV    L,A;     ZER0
                  03F6 93    SUB    B;     COMPLIEMENT 4TH FRACTION
                  03F7 5F    MOV    E,A;     4TH FRACTION
                  03F8 7D    MOV    A,L;     ZER0
                  03F9 9A    SUB    D;     COMPLIEMENT 3RD FRACTION
                  03FA 57    MOV    D,A;     3RD FRACTION
                  03FB 7D    MOV    A,L;     ZER0
                  03FC 99    SBB    C;     COMPLIEMENT 2ND FRACTN
                  03FD 4F    MOV    C,A;     2ND FRACTION
                  03FE 7D    MOV    A,L;     ZER0
03FF 98    SBB    B;     COMPLIEMENT 1ST FRACTN
MOV B,A;
; 1ST FRACTION
RET
; RETURN TO CALLER

; NORMALIZED THE REGISTERS.

MVI L,0400;
; MAX NORMALIZING SHIFT
MOV A,B;
; 1ST FRACTION

ANA A;
; SET CONTROL BITS

MOV A,B;
; 1ST FRACTION

MOV C,D;

MOV D,E;
; 3RD FRACTION

MOV F,A;
; ZERO 4TH FRACTION

MOV A,L;
; NORMALIZING SHIFT COUNT
SUI O100;
; REDUCE SHIFT COUNT

MOV L,A;
; NORMALIZING SHIFT COUNT

MOV NORM1;
; IF FRACTION NONZERO

RET
; IF FRACTION IS ZERO

BCK
; DECREMENT SHIFT COUNT

MOV A,E;
; ORIGINAL CONTENTS OF E

RAL
; LEFT SHIFT E

MOV F,A;
; RESTORE CONTENTS OF E REGISTER

MOV A,D;
; ORIGINAL CONTENTS OF D

RAL
; LEFT SHIFT D

MOV D,A;
; RESTORE CONTENTS OF D REGISTER

MOV A,C;
; ORIGINAL CONTENTS OF C

RAL
; LEFT SHIFT C

MOV C,A;
; RESTORE CONTENTS OF C REGISTER

MOV A,B;
; ORIGINAL CONTENTS OF B
0420 8F AOC 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 D020 SU1 040Q; REMOVE BIAS
0428 2E30 MVI L,ACCE; TO ADD 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 A=; A; SET SIGN TO IND. UNDERFLOW
042F C9 RET ; RETURN TO CALLER

; SUBROUTINE TO ROUND THE B,C,D REGISTERS.
0430 2E30 RND: 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 FC3FO4 CM RNDR; CALL 2ND LEVEL ROUNDER
0438 08 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; RETURNING THRU STORE SUBR.

; SECOND LEVEL ROUNding SUBROUTINE.
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
0447 5F MOV E,A; NEW ACCUMULATOR EXPONENT
0448 0680 MOV B,200; NEW 1ST FRACTION
0449 77 MOV M,A; NEW ACCUM EXONENT
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 MULXI; MULTIPLY BY ACCUM 2ND FRCTN
0463 2E32 MVI L,ACC1; TO ADDRESS 1ST FRCTN

; MULTIPLY BY ONE ACCUMULATOR WORD.
0465 7A MULXI: MOV A,D; 5TH PARTIAL PRODUCT
0466 59 MOV E,C; 4TH PARTIAL PRODUCT
0467 59 MOV D,B; 3RD PARTIAL PRODUCT
J468  46  MULX2:   MOV   B,M; MULTIPLIER
J469  6F  MOV   L,A;   5TH PARTIAL PRODUCT
J46A  AF  XRA   A;   ZERO
J46B  4F  MOV   C,A;   2ND PARTIAL PRODUCT
J46C  90  SUB   B;   SET CARRY BIT FOR EXIT FLAG
J46D  DA7904  JC   MULX3;   IF MULTIPLIER IS NOT ZERO
J470  4A  MOV   C,D;   2ND PARTIAL PRODUCT
J471  53  MOV   D,E;   3RD PARTIAL PRODUCT
J472  C9  RET   ;   MUL BY ZERO COMPLETE

; COMPLETE ADDITION OF MULTIPLICAND
J473  4F  MULXS:  MOV   C,A; 2ND PARTIAL PRODUCT
J474  D27904  JNC   MULX3;   IF NO CARRY TO 1ST PRODUCT
J477  04  INR   B;   ADD CARRY TO 1ST PRODUCT
J478  A7  ANA   A;   CLEAR CARRY BIT

; LOOP FOR EACH BIT OF MULTIPLIER WORD.
J479  7D  MULX3:  MOV   A,L; 5TH PARTIAL PRODUCT, EXIT
                  ;  FLAG

8080 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 PARTIAL PRODUCT, EXIT FLAG
047D  7B  MOV   A,E;   4TH PARTIAL PRODUCT
047E  17  RAL   ;   SHIFT 4TH PARTIAL PRODUCT
047F  5F  MOV   F,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 PRODUCT AND MULTIPLIER
0487  17  RAL  ;  SHIFT 1ST PRODUCT AND MULTIPLIER
0488  47  MOV  B,A;  1ST PART PRODUCT AND MULTIPLIER
0489  D27904  JNC  MULX3;  IF NO ADDITION REQUIRED
;  ADD THE MULTIPLICAND TO THE PRODUCT
;  IF THE MULTIPLIER BIT IS ONE.
048C  73  MOV  A,E;  4TH PARTIAL PRODUCT
048D  C30001  JMP  MULX4;  TO RAM CODE
;  FIXED POINT DIVIDE SUBROUTINE.
;  SUBTRACT DIVIDEND 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,F;  REMAINDER 3RD FRACTION
0495  2D  DCR  L;  TO ADDRESS ACCUM 2ND FRACTION
0496  7E  MOV  A,M;  ACCUMULATOR 2ND FRACTION
0497  9A  SUB  U;  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  SUB  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  RAK  ;  HALF OF DIVISOR 1ST FRACTION
+ 2000 TO CORRECT QUOTIENT
L,OP1S; TO ADDRESS 1ST SUBTRACT ; DIVISOR

04A1 2E19 MVI M,A; 1ST SUBTRACT DIVISOR
04A3 77 MOV M,A; 1ST ADD 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,O; INIT QUOTIENT 1ST FRCTN
04B9 7B 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
046D 77 MOV M,A; 4TH SUBTRACT DIVISOR
046E 2E1C MOV L,OP4A; TO ADDRESS 4TH ADD DIVISOR
046F 77 MOV M,A; 4TH ADD DIVISOR
0470 2E2A MOV L,OP4X; TO ADDRESS 4TH ADD DISION
0471 77 MOV M,A; 4TH ADD DIVISOR

; LOAD 1ST REMAINDER, CHECK SIGN
0472 2E32 MV5 l,ACCl; TO ADDR REMAINDER 1ST FRCTN
0473 7E MOV A1H; REMAINDER 1ST FRACTION
0474 2C INR L; TO ADDR REMAINDER 2ND FRCTN
0475 56 MOV D1H; REMAINDER 2ND FRACTION
0476 2C INR L; TO ADDR REMAINDER 3RD FRCTN
0477 58 MOV E1H; REMAINDER 3RD FRACTION
0478 A7 ANA A; SET CONTROL BITS
0479 FAF604 JM DIVX4; IF REMAINDER IS NEGATIVE

; ADJUST EXPONENT, POSITION REMAINDER
; AND INITIALIZE THE QUOTIENT,
047A 2E30 MV5 l,ACCE; TO ADDRESS ACCUMULATOR
; EXPONENT
047B 4E MOV C1H; QUOTIENT EXPONENT
047C 0C INCR C; INCREMENT QUOTIENT EXPONENT
047D CB RZ ; RETURN IF OVERFLOW
047E 71 MOV M1C; QUOTIENT EXPONENT
047F 6B MOV L1F; REMAINDER 3RD FRACTION
0480 62 MOV H1D; REMAINDER 2ND FRACTION
0481 5F MOV E1A; REMAINDER 1ST FRACTION
0482 1601 MV5 D1I; INITIALIZE QUOT 3RD FRCTN
0483 48 MOV C1B; INITIALIZE QUOT 2ND FRCTN

; SUBTRACT THE DIVISOR FROM THE REMAINDER
; IF IT IS POSITIVE.
0484 AF DIVX1: XRA A; REMAINDER 4TH FRCTN IS ZERO
0485 CD0001 CALL DIVX5; CALL RAM SECTION
DIVX2:  RLC  ;  SHIFT REM 4TH FRACTN TO CY
          ;  SHIFT THE REMAINDER LEFT ONE BIT
04E7  70  MOV  A,<P;
          ;  QUOTIENT 1ST FRACTION

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 14

04E1  17  RAL  ;  4S BIT OF QUOTIENT TO CY
04E2  08  RRC  ;  IF DIVISION COMPLETE
04E3  1F  RAR  ;  REMAINDER 4TH FRACTN TO CY
04E4  7D  MOV  A,U;
          ;  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 FRACTN
04E9  67  MOV  H,A;
          ;  REMAINDER 2ND FRACTION
04EA  CDBC03  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  DADBO4  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  C31003  JMP  DIVX6;
          ;  TO REM CODE
                    ;  POSITION THE REMAINDER AND INITIALIZE
          ;  THE QUOTIENT.
04F6  68  DIVX4:  MCV  L,F;
          ;  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     SCK SHR 8 AND OFFH
; ARITH EQU     BANK NUMBER OF SCRATCH PAD

; 8080 BINARY FLOATING POINT SYSTEM
; FORMAT CONVERSION 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.
; RAM LOCATION USED BY THE BINARY
; FLOATING POINT SYSTEM.
; OVER     EQU     560; OVERFLOW FLAG
; ACC1 EQU     600; ACCUMULATOR EXPONENT
; ACCS EQU     ACC1+1; ACCUMULATOR SIGN
; ACCS+1; ACCUMULATOR 1ST FRCTN
0033   ACC2 EQU     ACC2+1; ACCUMULATOR 2ND FRCTN
0034   ACC3 EQU     ACC3+1; ACCUMULATOR 3RD FRCTN
; SF EQU     ACC3+1; SUBTRACTION FLAG
0036   ADRL EQU     SF+1; CHARACTER STRING WORD

8080 MACRO ASSEMBLER, VER 2.4
FRNDRS = 0 PAGE 15
<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0037</td>
<td>ADRH</td>
<td>EQU ADRH+1; TEMPORARY STORAGE</td>
</tr>
<tr>
<td>0038</td>
<td>TMP1</td>
<td>EQU ADRH+1; TEMPORARY STORAGE</td>
</tr>
<tr>
<td>0039</td>
<td>TMP2</td>
<td>EQU TMP1+1; TEMPORARY STORAGE</td>
</tr>
<tr>
<td>003A</td>
<td>TMP3</td>
<td>EQU TMP2+1; VALUE EXPONENT</td>
</tr>
<tr>
<td>003B</td>
<td>VAL1</td>
<td>EQU VALF+1; VALUE 1ST FRACTION</td>
</tr>
<tr>
<td>003C</td>
<td>VAL2</td>
<td>EQU VAL1+1; VALUE 2ND FRACTION</td>
</tr>
<tr>
<td>003D</td>
<td>VAL3</td>
<td>EQU VAL2+1; VALUE 3RD FRACTION</td>
</tr>
<tr>
<td>003E</td>
<td>TMP4</td>
<td>EQU VAL3+1; TEMPORARY STORAGE</td>
</tr>
</tbody>
</table>

; ADDRESS IN THE ARITHMETIC AND UTILITY PACKAGE REFERENCE BY THE FORMAT CONVERSION PACKAGE.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR</td>
<td>EQU ARITH+760</td>
</tr>
<tr>
<td>ZRO</td>
<td>EQU ARITH+1060</td>
</tr>
<tr>
<td>ABS</td>
<td>EQU ARITH+1200</td>
</tr>
<tr>
<td>TST</td>
<td>EQU ARITH+1310</td>
</tr>
<tr>
<td>LCT</td>
<td>EQU ARITH+1550</td>
</tr>
<tr>
<td>MUL</td>
<td>EQU ARITH+2140</td>
</tr>
<tr>
<td>DIV</td>
<td>EQU ARITH+2640</td>
</tr>
<tr>
<td>AD</td>
<td>EQU ARITH+3270</td>
</tr>
<tr>
<td>ADD10</td>
<td>EQU ARITH+5530</td>
</tr>
<tr>
<td>LSH</td>
<td>EQU ARITH+5740</td>
</tr>
<tr>
<td>KSH</td>
<td>EQU ARITH+7110</td>
</tr>
<tr>
<td>COMPL</td>
<td>EQU ARITH+7640</td>
</tr>
</tbody>
</table>

; SUBROUTINE TO CONVERT FROM FIXED POINT TO FLOATING POINT FORMAT.
; ARBITRARY BCD CHARACTER CODES, (OFFSET 30H FROM ASCII):

; SPACE - 0FH DEcimal Point - OFEH
; PLUS - 0FBH LETTER Z - 015H
;  MINUS - OFDH

J4FF 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 70 MOV A,L; INPUT EXPONENT
0505 EE80 XRI 2000; APPLY EXPONENT BIAS
0507 2601 MVI H,SCRH; TO ADDRESS SCRATCH RANK
0509 2E30 MVI L,ACCE; TO ADDRESS ACCUM EXPONENT
0503 77 MOV M,A; ACCUMULATOR EXPONENT
050C 2C INR L; TO ADDRESS ACCUM SIGN
050D 3680 MVI M,20UQ; SET ACCUM SIGN POSITIVE
050F 2C INR L; TO ADD ACCUM 1ST FRCTN
0510 73 MOV A,B; 1ST INPUT FRACTION
0511 A7 ANA A; SET SIGN BIT
0512 17 RAL; INPUT SIGN TO CARRY
0513 C36003 JMP ADDH.D; COMPLETE CONVERSION
; SUBROUTINE TO CONVERT FROM FLOATING
; POINT TO FIXED POINT FORMAT.
0516 2601 FIX: MVI H,SCRH; TO ADDRESS SCRATCH RANK

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 16

J518 2E30 MVI L,ACCE; TO ADD 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 1771; APPLY BIAS - 1
0522 96 SUB M; SHIFT COUNT -1
0523 D8 RC ; RETURN IF ACCUM TOO LARGE
0524 FE1F CPI O371; COMPARE TO LARGE SHIFT
0526 D24F05 JNC FIX1; IF ACCUMULATOR TOO SMALL
0529 C601 ADI l; SHIFT COUNT
052B 2E32 MVI L,ACCI; 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 FRACTION
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,ACC; TO ADDR ACCUM SIGN
0537 7E MOV A,M; ACCUMULATOR SIGN
0538 A7 ANA A; SET CONTROL BITS
0539 F4F03 CP CMP1; COMPARE FRCTN IF NEG
053C 3E01 MVI A,1; NON-ZERO
053E B0 ORA B; SET CONTROL BITS FOR EXIT
053F 73 MOV A,B; 1ST RESULT
0540 41 MOV B,C; 2ND RESULT
0541 4A MOV C,D; 3RD RESULT
0542 53 MOV D,F; 4TH RESULT
0543 C9 RET ; RETURN TO CALLER
0544 AF FIX1; XRA 4; 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 9; CHECKSUM WORD

; INP SUBROUTINE ENTRY POINT
; INITIALIZE TEMPORARY STORAGE
054A  5E  INP: MOV  E,M;  FIRST CHARACTER OF STRING
054B  C0D406  CALL  SVAD;  SET CHAR ADDR, PNT FLG, EXP
054E  2C  INR  L;  TO ADDRESS VALUE SIGN
054F  3680  MVI  M,2000;  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  3600;  COMPARE TO SPACE
0557  C6705  JZ  INP1;  IF SPACE CHARACTER
055A  FEF8  CPI  3730;  COMPARE CHAR TO PLUS
055C  C6705  JZ  INP1;  IF PLUS SIGN
055F  FEF0  CPI  3750;  COMPARE TO MINUS
0561  C2D05  JNZ  INP2;  IF NOT MINUS SIGN
0564  2E3A  MVI  L,IMP3;  TO ADDR VALUE SIGN

8030 Macro Assembler, Ver 2.4
ERRORS = 0 PAGE 17

0565  72  MOV  M,D;  SET VALUE OF SIGN NEGATIVE
0567  CDE06  INP1:  CALL  CHAO; CALL CHAR ADDR SHRTN
056A  7E  MOV  A,M;  NEXT CHARACTER
056B  2601  MVI  M,SCR0;  TO ADDRESS SCRATCH BANK
056D  9600  INP2:  MVI  B,0;  DIGIT 2ND WD OP DEC EXP
056F  FEF8  CPI  3760;  COMPARE TO DECIMAL POINT
0571  AAA05  JZ  INP3;  IF DECIMAL POINT
0574  FE15  CPI  0250;  COMPARE TO EXPONENT SIGN
0576  CBB405  JZ  INP4;  IF EXPONENT SIGN
0579  FE0A  CPI  120;  SET CARRY IF CHAR IS DIGIT
057B  025505  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, FIFEN; TO ADDR FLOATING TEN
0584 CD8C02 CALL MUL; MULTIPLY BY TEN
0587 2E3B MVI L, LVALE; TO ADDR VALUE
0589 CD3E02 CALL STR; STORE OLD VALUE TIMES TEN
058C 2C INR L; TO ADDR CURRENT DIGIT
058E 7E MOV M,A; CURRENT DIGIT
0590 0600 MVI B,0; CLEAR 2ND WORD OF DIGIT
0590 48 MOV C,B; CLEAR 3RD WORD OF DIGIT
0591 53 MOV D,B; CLEAR 4TH WORD OF DIGIT
0592 1E08 MVI E,010Q; INDICATE DIGIT IS IN REG A
0594 CDFF34 CALL FLT; CONVERT DIGIT TO FLOATING PNT
0597 2E3B MVI L, LVALE; TO ADDR VALUE
0599 CD0702 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 20 DCR L; TO ADD INPUT EXPONENT
05A4 46 MOV B,M; INPUT EXPONENT
05A5 05 DCR b; UPDATE INPUT EXPONENT
05A6 70 MOV M,b; TO GET NEXT CHARACTER

05A7 C36705 JMP INP1; TO ADDR DEC PNT FLAG
05AC AE XFA M; ZERO IF FLAG SET
05AD 77 MOV M,A; SET DEC PNT FLAG
05AE C26705 JNZ INP1; IF FLAG NOT ALREADY SET
05A1 C3E505 JMP INP8; IF 2ND DEC PNT

; PROCESS DECIMAL EXPONENT
05B4 CDF106 IMP4: CALL CHAR; CALL CHAR ADDR SBRTL
05B7 7E MOV A,M; NEXT CHARACTER OF STRING
05B8 47 MOV B,A; CURRENT CHARACTER
05B9 06FD SUI 375Q; COMPARE TO MINUS CHAR
05B8 5F MOV E,A; CHAR = MINUS SIGN
05B8 CAC505 JZ INP5; IF MINUS SIGN
05B8 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 JNL INP6; IF NOT PLUS SIGN
05C5 2C INP5: INR L; TO ADDRESS NEXT CHAR
05C6 7E MOV A,M; NEXT CHARACTER OF STRING
05C7 06D0 INP6: MOV MV1 B,0; POSSIBLE DEC EXPONENT
05C8 F9EA CPI 120; SET CARRY IF CHAR IS DIGIT
05CB 02E505 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 7F MOV A,M; NEXT CHARACTER OF STRING
05D1 F9EA CPI 120; SET CARRY IS NOT A DIGIT
05D3 02DE05 JNC INP7; IF CHAR IS NOT A DIGIT
; FORM COMPLETE DECIMAL EXPONENT
05D6 4F MOV C,A; LS DIGIT OF DEC EXP
05D7 7d MOV A,B; MS DIGIT OF DEC EXP
05D8 87 ADD A; 2 * MS DIGIT
05D9 87 ADD A; 4 * MS DIGIT
05DA 8j ADD B; 5 * MS DIGIT
05DB 87 ADD A; 10 * MS DIGIT
050C 81  ADD   C;   10 # HS + LS DIGIT
050D 47  MOV   B,A;  DECIMAL EXPONENT
050E 78  INP7:  MOV  A,AL;  SIGN OF DEC EXPONENT
050F A7  ANA   A;  ; SET CONTROL BITS
0510 02E505 JNC  INPB;  IF SIGN PLUS
0511 90  SUB   B;  COMPLEMENT DEC EXP
0512 47  MOV   B,A;  DECIMAL EXPONENT
0513 2601 INPB:  MVI   L,TMP3;  TO ADDRESS SCRATCH BAND
0514 2E3A  MVI   L,TMP3;  TO ADDRESS INPUT SIGN
0515 4E  MOV   C,M;  INPUT SIGN
0516 2E31  MVI   L,ACC;  TO ADDRESS ACCUM SIGN
0517 71  MOV   M,C;  ACCUMULATOR SIGN
0518 78  MOV   A,R;  DECIMAL EXPONENT
               ; CONVERT DECIMAL EXPONENT TO BINARY.
051E 2E38  INP9:  MVI   L,TMP1;  TO ADDRESS DEC EXPONENT
051F 8B  ADD   M;  ADJUST DECIMAL EXPONENT
0520 CA5002  JZ  TEST;  YN DEC EXP IS ZERO
0521 77  MOV   N,A;  CURRENT DECIMAL EXPONENT
0522 21EC06 LXI   H,FTEN;  TO ADD FLOATING TEN
0523 F20306 JP   INP10;  IF MULTIPLY REQUIRED
0524 C98402 CALL  DIV;  DIVIDE BY TEN
0525 3C01  MVI   A,1;  TO INCREMENT DEC EXP
0526 C3EE05 JMP   INP9;  TO TEST FOR COMPLETION
0527 C08C02 INP10:  CALL  MUL;  MULTIPLY BY TEN
0528 0D  DCH   RC;  RETURN IF OVERFLOW
0529 3EFF  MVI   A,3770;  TO DECREMENT DEC EXP
052A C3EE05 JMP   INP9;  TO TEST FOR COMPLETION
               ; OUT SUBROUTINE ENTRY POINT.
               ; SAVE CHARACTER ADDRESS AND ACCUMULATOR.
060C 2D  OUT:  DCR   L;  DECREMENT CHARACTER ADDRESS
060D C0D06 CALL  SVAD;  SET CHAK ADDR, DIG CNT, DEC EXP
0610 C05902 CALL  TST;  LOAD ACCUM TO REGISTERS
8086 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 19

0613 2E3B  MVI  L,VALE;  TO ADDR ACCUM SAVE AREA
0615 CD3E02 CALL STR;  CALL RFG STR SUBROUTINE

; OUTPUT SIGN CHARACTER.
061B  CDE106 CALL CHAD;  CALL CHAR ADDR SBRTN
061B  36F0  MVI M,3600;  STORE SPACE CHARACTER
061D  A7   ANA A;  ; SET CONTROL BITS
061E  CA3A06 J2  OUT3;  IF ACCUMULATOR IS ZERO
0621  5F   MOV E,A;  ACCUMULATOR EXPONENT
0622  78   MOV A,X;  ACCUM SIGN AND 1ST FRACT
0623  A7   ANA A;  ; SET CONTROL BITS
0624  78   MOV A,E;  ACCUMULATOR EXPONENT
0625  F22A06 JP OUT1;  IF ACCUM IS POSITIVE
062D  36FD  MVI M,3750;  CHANGE SIGN TO MINUS

; SCALE ACCUMULATOR TO .1 = 1. RANGE/
062A  FE7E  OUT1: CPI 1750; COMPARE TO SMALL EXPONENT
062C  21EC06 OUT2: LXI H,FEN; TO ADDR FLOATING TEN
062F  DA4406 JC OUT4; IF EXPONENT TOO SMALL
0632  FE81  CPI 2014; COMPARE TO LARGE EXP
0634  DA4F06 JC OUT5; IF EXP SMALL ENOUGH
0637  CD8402 CALL DIV; DIVIDE BY TEN
063A  2501  OUT3: MVI H,CHR; TO ADDRESS SCRATCH BANK
063C  2E39  MVI L,TMP2; TO ADDR DECIMAL EXPONENT
063E  5E   MOV E,N;  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 10 DCR E; DECREMENT DECIMAL EXPONENT
064B 73 MOV M,F; 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,RND0; TO ADDRES ROUNDER
0655 C0D702 CALL AD; ADD THE ROUNDER
0658 FE81 CPI 201W; 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 FE00 CPI O100; 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 F; DIGITS AFTER DECIMAL POINT
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 F; DECREMENT DIGIT COUNT
0670 7B MOV A,F; DIGITS BEFORE DEC PNT
; OUTPUT SIGNIFICANT DIGITS.
0671 2E38 OUT7: MVI [TMP1] TO ADDR DIGIT COUNT
0673 96 ADD M; ADJUST DIGIT COUNT
0674 77 MOV M,A; NEW DIGIT COUNT
0675 FA9206 J4 OUT8; IF COUNT RUN OUT
0676 21EC06 LXI H,F,TEN; TO ADDR FLOATING TEN
067B CD8C02 CALL MUL; MULTIPLY BY TEN
067E 1E08 MVI E,10Q; TO PLACE DIGIT IN PARA.
0680 CD1605 CALL FIX; CONVERT TO FIXED FORMAT
0683 CDE106 CALL CHO,D; CALL CHAR ADDR SBRT
0686 77 MOV M,A; OUTPUT DECIMAL DIGIT
0687 AF XRA A; CLEAR CURRENT DIGIT
0688 1E03 MVI E,010Q; BINARY SCALING FACTOR
068A CDFF04 CALL FLT; RESTORE VALUE MINUS DIGIT
068D 3EFF MVI A,7777; TO ADJUST DIGIT CNT
068F C37106 JMP OUT7; YLCOP FOR NEXT DIGIT
0692 2E34 OUT8: MVI L,TMP3; TO ADDR 2ND DIGIT CNT
0694 7E MOV A,M; DIGITS AFTER DECIMAL PNT
0695 3EFF MVI M,3777; SET 2ND COUNT NEG
0697 A7 ANA A; SET CONTROL BITS
0698 FAA306 JF OUT9; IF 2ND COUNT RUN OUT
069B CD106 CALL CHA,D; CALL CHAR ADDR SBRT2
069E 36FE MVI M,3760; STORE DECIMAL POINT
06A0 2691 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 CACCC06 JZ OUT13; IF DECIMAL EXPONENT IS ZERO
; OUTPUT DECIMAL EXPONENT.
06AA 06FB MVI B,3730; 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  0120;  REDUCE REMAINDER
06BA  D2B606  JNC  OUT11;  IF MORE TENS
06BD  3E15  MVI  A,025Q;  EXPONENT SIGN
06BF  CDE106  OUT12:  CALL  CHAD; CALL CHAR ADDR SRTRN
06C2  CD3EO2  CALL  STR;  STORE LAST 4 CHARACTER
06C5  2601  MVI  H,SCRH;  TO ADDRESS SCRATCH BANK
06C7  223B  MVI  L,VALE;  TO ADDRESS ACCUM SAVE AREA
06C9  C36E02  JMP  LCD;  RESTORE ACCUM AND EXIT
;  OUTPUT 4 SPACES IF EXPONENT IS ZERO.

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 21

06CC  3EFO  OUT13:  MVI  A,3600; 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
0605  44  MOV  B,H;  CHARACTER STRING BANK
0606  0E00  MVI  C,O;  INPUT EXP OR DIGIT
0608  51  MOV  D,C;  DEC PNT FLAG OR DEC EXP
0609  2E01  MVI  H,SCR;  TO ADDRESS SCRATCH BANK
060B  2E36  MVI  L,ADR;  TO ADDR CHAR STRING WORD
060D  C03E02  CALL  STR;  STORE A, B, C, AND D
0630  C9  RET  ;  RETURN TO CALLER
0631  2E01  CHAD:  MVI  H,SCR;  TO ADDRESS SCRATCH BANK
0633  2E36  MVI  L,ADR;  TO ADDR CHAR STRING WORD
0635  5E  MOV  F,M;  CHARACTER STRING WORD
0636  1C  INR  E;  TO ADDR NEXT CHARACTER
0637  73  MOV  M,E;  UPDATE CHAR STRING WORD
0638  2C  INR  L;  TO ADDR CHAR STRING BANK
0639  66  MOV  H,M;  CHARACTER STRING WORD
063A  68  MOV  L,E;  CHARACTER STRING WORD
063B  C9  RET  ;  RETURN TO CALLER
063C  84200000  FTE:  DR  2040,0400,000; FLOATING TEN
063D  6856BFAD  RND:  DB  1500,1260,2770,2550; .00000005
063F  09  DB  END  0;  CHECKSUM WORD
0641  00  DB  END  0;

NO PROGRAM ERRORS

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 22

SYMBOL TABLE

* 01
<table>
<thead>
<tr>
<th>A</th>
<th>0007</th>
<th>ABS</th>
<th>0250</th>
<th>ACC1</th>
<th>0032</th>
<th>ACC2</th>
<th>0033</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC3</td>
<td>0034</td>
<td>ACCE</td>
<td>0030</td>
<td>ACCS</td>
<td>0031</td>
<td>AD</td>
<td>0207</td>
</tr>
<tr>
<td>ADD10</td>
<td>036B</td>
<td>ADD11</td>
<td>0374</td>
<td>ADD12</td>
<td>037A</td>
<td>ADD17</td>
<td>0386</td>
</tr>
<tr>
<td>ADD2</td>
<td>033E</td>
<td>ADD3</td>
<td>032D</td>
<td>ADD9</td>
<td>0350</td>
<td>ADRH</td>
<td>0037</td>
</tr>
<tr>
<td>ADRL</td>
<td>0036</td>
<td>ARITH</td>
<td>0200</td>
<td>ARTHB</td>
<td>0002</td>
<td>B</td>
<td>0000</td>
</tr>
<tr>
<td>C</td>
<td>0001</td>
<td>CHAD</td>
<td>06E1</td>
<td>CHS</td>
<td>0240</td>
<td>COMP</td>
<td>03EF</td>
</tr>
<tr>
<td>COMP1</td>
<td>03F4</td>
<td>D</td>
<td>0032</td>
<td>DIV</td>
<td>02B4</td>
<td>DIVX</td>
<td>0490</td>
</tr>
<tr>
<td>DIVX1</td>
<td>040B</td>
<td>DIVX2</td>
<td>04F6</td>
<td>DIVX3</td>
<td>04F2</td>
<td>DIVX4</td>
<td>04F6</td>
</tr>
<tr>
<td>DIVX5</td>
<td>01JD</td>
<td>DIVX6</td>
<td>011E</td>
<td>F</td>
<td>0033</td>
<td>FIX</td>
<td>0516</td>
</tr>
<tr>
<td>FIX1</td>
<td>0544</td>
<td>FLT</td>
<td>04FF</td>
<td>FREN</td>
<td>05EC</td>
<td>H</td>
<td>0004</td>
</tr>
<tr>
<td>INIT</td>
<td>029E</td>
<td>INIT1</td>
<td>0231</td>
<td>INP</td>
<td>054A</td>
<td>INP1</td>
<td>0567</td>
</tr>
<tr>
<td>INP10</td>
<td>0603</td>
<td>INP2</td>
<td>0560</td>
<td>INP3</td>
<td>05A4</td>
<td>INP4</td>
<td>05B4</td>
</tr>
<tr>
<td>INP5</td>
<td>05C5</td>
<td>INP6</td>
<td>05C7</td>
<td>INP7</td>
<td>05D2</td>
<td>INP8</td>
<td>05E5</td>
</tr>
<tr>
<td>INP9</td>
<td>05EE</td>
<td>L</td>
<td>0005</td>
<td>LOD</td>
<td>026E</td>
<td>LOOL</td>
<td>027B</td>
</tr>
<tr>
<td>LSH</td>
<td>03BC</td>
<td>LSH1</td>
<td>03BF</td>
<td>M</td>
<td>0006</td>
<td>MUX</td>
<td>0395</td>
</tr>
<tr>
<td>MUL</td>
<td>023C</td>
<td>MULP1</td>
<td>0009</td>
<td>MULP2</td>
<td>0005</td>
<td>MULP3</td>
<td>0001</td>
</tr>
<tr>
<td>MULX</td>
<td>044D</td>
<td>MULX1</td>
<td>0465</td>
<td>MULX2</td>
<td>0468</td>
<td>MULX3</td>
<td>0479</td>
</tr>
<tr>
<td>MULX4</td>
<td>0100</td>
<td>MULX5</td>
<td>0473</td>
<td>NORM</td>
<td>0402</td>
<td>NORM1</td>
<td>0404</td>
</tr>
<tr>
<td>NORM2</td>
<td>0415</td>
<td>NORM3</td>
<td>0422</td>
<td>OP1A</td>
<td>0027</td>
<td>OP1S</td>
<td>0019</td>
</tr>
<tr>
<td>OP2A</td>
<td>0023</td>
<td>OP2S</td>
<td>0015</td>
<td>OP3A</td>
<td>001E</td>
<td>OP3S</td>
<td>0011</td>
</tr>
<tr>
<td>OP4A</td>
<td>001C</td>
<td>OP4S</td>
<td>000E</td>
<td>OP4X</td>
<td>0024</td>
<td>P</td>
<td>060C</td>
</tr>
<tr>
<td>OUT1</td>
<td>062A</td>
<td>OUT10</td>
<td>0644</td>
<td>OUT11</td>
<td>0686</td>
<td>OUT12</td>
<td>06AF</td>
</tr>
<tr>
<td>OUT13</td>
<td>06CC</td>
<td>OUT2</td>
<td>062C</td>
<td>OUT3</td>
<td>063A</td>
<td>OUT4</td>
<td>0644</td>
</tr>
<tr>
<td>OUT5</td>
<td>064F</td>
<td>OUT6</td>
<td>0658</td>
<td>OUT7</td>
<td>06F1</td>
<td>OUT8</td>
<td>0692</td>
</tr>
<tr>
<td>OUT9</td>
<td>06A5</td>
<td>OVER</td>
<td>032E</td>
<td>OVERF</td>
<td>02CA</td>
<td>OVFN</td>
<td>03B8</td>
</tr>
<tr>
<td>PRX</td>
<td>002F</td>
<td>PSW</td>
<td>0006</td>
<td>RANDO</td>
<td>05F0</td>
<td>RANDA</td>
<td>02A9</td>
</tr>
<tr>
<td>RNDR</td>
<td>043F</td>
<td>ROND</td>
<td>0430</td>
<td>RSH</td>
<td>03C9</td>
<td>RSH0</td>
<td>03CB</td>
</tr>
<tr>
<td>RSH1</td>
<td>03CD</td>
<td>RSH2</td>
<td>030A</td>
<td>RSH3</td>
<td>030D</td>
<td>SP</td>
<td>0204</td>
</tr>
<tr>
<td>SCR</td>
<td>0100</td>
<td>SCR6</td>
<td>0901</td>
<td>SF</td>
<td>0035</td>
<td>SP</td>
<td>0006</td>
</tr>
<tr>
<td>STR</td>
<td>023E</td>
<td>STR1</td>
<td>023F</td>
<td>STR1</td>
<td>023C</td>
<td>SVAD</td>
<td>0604</td>
</tr>
<tr>
<td>TMP1</td>
<td>0038</td>
<td>TMP2</td>
<td>0039</td>
<td>TMP3</td>
<td>0034</td>
<td>TMP4</td>
<td>003F</td>
</tr>
</tbody>
</table>
APPENDIX F

Reactimeter Program

Inhour Equation
PART MACRO LAMI,BII
LXI H,LAMI ;LOAD ADDRESS OF LAMI IN HL
LODE
LXI H, PER ;LOAD ADDRESS OF PER IN HL
CALL MUL ;MULTIPLY
LXI H, ONE ;LOAD ADDRESS OF ONE IN HL
CALL AD ;AD
LXI H, HOLD ;LOAD ADDRESS OF HOLD IN HL
STRIN
LXI H,BII ;LOAD ADDRESS OF BII IN HL
LODE
LXI H,HOLD ;LOAD ADDRESS OF HOLD IN HL
CALL DIV ;DIVIDE
LXI H,RES ;LOAD ADDRESS OF RES IN HL
CALL AD ;ADD
LXI H,RES ;LOAD ADDRESS OF RES IN HL
STRIN
ENDM
LODE MACRO
CALL LOD ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
;FROM THE ADDRESS GIVEN IN HL
ENDM
STRIN MACRO
CALL STR ;STR--STORES THE FLOATING-POINT-
;ACCUMULATOR IN THE ADDRESS GIVEN BY HL
ENDM

022F ORG 022FH
022F 00 INT: NOP
THE FOLLOWING LABELS ARE FOR THE FLOATING POINT ROUTINES THAT ARE IN EPROM AND ARE BY CERTAIN ROUTINES TO PERFORM MATHEMATICAL OPERATIONS. THESE ROUTINES ARE IN ANOTHER PRINT OUT AND ARE GIVEN HERE AS DUMMY PROGRAMS TO BE référence BY THE CROSS ASSEMBLER.

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 2
04FF 00 FLT: NOP
0500 C9 RET
054A 00 ORG 054AH
054A 00 INP: NOP
054B C9 RET
060C 00 ORG 060CH
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 36FO MVI M, 360Q
0008 2C INR L
0009 2C INR L
000A 2C INR L
0008 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 D30A INPUT: OUT 10 ;GENERATE A DEVICE SELECT PULSE
0028 INTRO: ORG 0028H ;TO DEVICE TEN
002A 210E13 LXI H, DMANT+4;ADDRESS OF THE TENTHS DIGIT OF DMANT
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
CALL SB ;SUBTRACT
CALL FLAG ;CALL SUBROUTINE FLAG
CALL ABS ;ABSOLUTE VALUE
POP H ;POP HL OFF OF THE STACK
CALL DIV ;DIVIDE
LXI H, TH100 ;LOAD ADDRESS OF TH100 IN HL
CALL SB ;SUBTRACT
JMP TRACK ;JUMP TO TRACK ON NEGATIVE RESULT
LXI H, NTH10 ;LOAD ADDRESS OF NTH10 IN HL
CALL SB ;SUBTRACT
JMP CALC ;JUMP TO CALC ON NEGATIVE RESULT
CALL EXCH ;CALL EXCH
JMP WAIT ;JUMP TO WAIT
LXI H, FLX2 ;LOAD ADDRESS OF FLX2 INTO HL
LOD
CALL LOD ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
LXI H, FLX1 ;LOAD ADDRESS OF FLX1 INTO HL
STR
CALL STR ;STR--STORES THE FLOATING-POINT-
LXI H, MEM ;LOAD ADDRESS OF MEM INTO HL
MVI M, 0 ;MEM = 0
DCR L
MVI M, 0 ;MEM1 = 0
RET
LXI H, MEM ;LOAD ADDRESS OF MEM IN HL
INR M ;INCREMENT CONTENTS OF MEM
JZ HMEM ;JUMP TO HMEM IF RESULT IS ONE
JMP WAIT ;JUMP TO WAIT
DCR L ;DECREMENT L
INR M ;INCREMENT MEM+1
; A = 4
; SUBTRACT MEM+1 FROM A
; JUMP TO WAIT IF RESULT IS NOT ZERO
; JUMPT TO OTPTO
; ENABLE FLAG INTERRUPT
; 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
; THE DPM WHICH OCCURS EVERY 0.01 SEC.
; JUMP TO WAIT
; LOAD ADDRESS OF SGST IN HL
; JUMP TO NEGAT IF RESULT IS NEGATIVE
; SET M = 0
; RETURN FROM SUBROUTINE
; SET M = 001Q
; LOAD ADDRESS OF PER IN HL
; LOD--LOADS FLOATING-POINT-ACCUMULATOR
; FROM THE ADDRESS GIVEN IN HL
; CHANGE OF SIGN
; LOAD ADDRESS OF PER IN HL
; STR--STORES THE FLOATING-POINT-
; ACCUMULATOR IN THE ADDRESS GIVEN BY HL
008C  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
1410  210613    LXI H, PER       ;LOAD ADDRESS OF PER IN HL
1413  1 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
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
141D  1 210001  +       LXI H,LAMI   ;LOAD ADDRESS OF LAMI IN HL
1420  2 CD6E02  +       CALL LOD    ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
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 CD0702  +       CALL AD      ;AD
142F  1 211013  +       LXI H, HOLD  ;LOAD ADDRESS OF HOLD IN HL
1432  2 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
1435  2 +        RET

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
1410  210613    LXI H, PER       ;LOAD ADDRESS OF PER IN HL
1413  1 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
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
141D  1 210001  +       LXI H,LAMI   ;LOAD ADDRESS OF LAMI IN HL
1420  2 CD6E02  +       CALL LOD    ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
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 CD0702  +       CALL AD      ;AD
142F  1 211013  +       LXI H, HOLD  ;LOAD ADDRESS OF HOLD IN HL
1432  2 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
1435  2 +        RET

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
1410  210613    LXI H, PER       ;LOAD ADDRESS OF PER IN HL
1413  1 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
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
141D  1 210001  +       LXI H,LAMI   ;LOAD ADDRESS OF LAMI IN HL
1420  2 CD6E02  +       CALL LOD    ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
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 CD0702  +       CALL AD      ;AD
142F  1 211013  +       LXI H, HOLD  ;LOAD ADDRESS OF HOLD IN HL
1432  2 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
1435  2 +        RET

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
1410  210613    LXI H, PER       ;LOAD ADDRESS OF PER IN HL
1413  1 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
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
141D  1 210001  +       LXI H,LAMI   ;LOAD ADDRESS OF LAMI IN HL
1420  2 CD6E02  +       CALL LOD    ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
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 CD0702  +       CALL AD      ;AD
142F  1 211013  +       LXI H, HOLD  ;LOAD ADDRESS OF HOLD IN HL
1432  2 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
1435  2 +        RET

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
1410  210613    LXI H, PER       ;LOAD ADDRESS OF PER IN HL
1413  1 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
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
141D  1 210001  +       LXI H,LAMI   ;LOAD ADDRESS OF LAMI IN HL
1420  2 CD6E02  +       CALL LOD    ;LOD--LOADS FLOATING-POINT-ACCUMULATOR
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 CD0702  +       CALL AD      ;AD
142F  1 211013  +       LXI H, HOLD  ;LOAD ADDRESS OF HOLD IN HL
1432  2 CD3E02  +       CALL STR    ;STR--STORES THE FLOATING-POINT-
1435  2 +        RET
1435 1 211801 + LXI H, B11 ;LOAD ADDRESS OF B11 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 CD0702 + 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, B12  
144D 1 210401 + LXI H, LAM2 ;LOAD ADDRESS OF LAM2 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 CD0702 + 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, B12 ;LOAD ADDRESS OF B12 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 CDB402  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, BL3
147D 1 210801  LXI H, LAM3 ; LOAD ADDRESS OF LAM3 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 CDB802  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, BL3 ; LOAD ADDRESS OF BL3 IN HL

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 6
1498 2 CD6E02 + CALL LOD  
1498 1 211013 + LODD--LOADS FLOATING-POINT-ACCUMULATOR  
149E 1 CDB402 + FROM THE ADDRESS GIVEN IN HL  
14AA 2 CD3E02 + LOAD ADDRESS OF HOLD IN HL  
14A1 1 210213 + CALL DIV  
14A4 1 CDD702 + DIVIDE  
14A7 1 210213 + LXI H,RES  
14AA 2 CD3E02 + LOAD ADDRESS OF RES IN HL  
14AF 1 210013 + CALL STR  
14AD 1 210C01 + STR--STORES THE FLOATING-POINT-  
14B0 2 CD6E02 + ACCUMULATOR IN THE ADDRESS GIVEN BY HL  
14BB 1 210613 + PART LAM4, BI4  
14B3 1 210613 + LOAD ADDRESS OF LAM4 IN HL  
14B6 1 CDBC02 + LODE  
14BF 1 211013 + CALL LOD  
14C2 2 CD3E02 + LODD--LOADS FLOATING-POINT-ACCUMULATOR  
14C5 1 212401 + FROM THE ADDRESS GIVEN IN HL  
14C8 2 CD6E02 + LOAD ADDRESS OF RES IN HL  
14CE 1 CDB402 + CALL LOD  
14D1 1 210213 + LODD--LOADS FLOATING-POINT-ACCUMULATOR  
14D4 1 CDD702 + FROM THE ADDRESS GIVEN IN HL  
14D7 1 210213 + CALL DIV  
14D9 1 213801 + DIVIDE  
14DA 1 210103 + LXI H,RES  
14DB 1 211013 + CALL AD  
14DD 1 210213 + ADD  
14EF 1 210103 + LXI H,RES  
14F0 1 210213 + CALL AD  
14F2 1 210213 + ADD  
14F4 1 210213 + LXI H,RES  
14F6 1 210213 + CALL AD  
14F8 1 210213 + ADD  
14FA 1 210213 + LXI H,RES  
14FB 1 210213 + CALL AD  
14FC 1 210213 + ADD  
14FD 1 210213 + LXI H,RES  
14FE 1 210213 + CALL AD  
14FF 1 210213 + ADD
2 + STRIN
14DA 2 CD3E02 + CALL STR ;STR--STORES THE FLOATING-POINT-ACCUMULATOR IN THE ADDRESS GIVEN BY HL
1 + PART LAM5, B15
14DD 1 211001 + LXI H, LAM5 ;LOAD ADDRESS OF LAM5 IN HL
2 + LODE
14E0 2 CD6E02 + CALL LOD ;LOD--LOADS FLOATING-POINT-ACCUMULATOR 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 CD0702 + 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-ACCUMULATOR IN THE ADDRESS GIVEN BY HL
14F5 1 212801 + LXI H, B15 ;LOAD ADDRESS OF B15 IN HL
2 + LODE

8080 MACRO ASSEMBLER, VER 2.4
ERRORS = 0 PAGE 7

14F8 2 CD6E02 + CALL LOD ;LOD--LOADS FLOATING-POINT-ACCUMULATOR FROM THE ADDRESS GIVEN IN HL
14F8 1 211013 + LXI H, HOLD ;LOAD ADDRESS OF HOLD IN HL
14FE 1 CD8402 + CALL DIV ;DIVIDE
1501 1 210213 + LXI H, RES ;LOAD ADDRESS OF RES IN HL
1504 1 CD0702 + 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 + PART LAM6, BI6
1 +
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 CD0702 + 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 + PART BI6 IN HL
1525 1 212C01 + LXI H, BI6 ;LOAD ADDRESS OF BI6 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 CD8402 + CALL DIV ;DIVIDE
1531 1 210213 + LXI H, RES ;LOAD ADDRESS OF RES IN HL
1534 1 CD0702 + 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 + PART PNLF IN 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 COB402 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
0153 06FO MVI B,360Q ; RHO+9
0155 BB CMP B ; THIS VALUE IS TESTED TO DETERMINE IF A
0156 CA7101 JZ CHK1 ; BLANK OR THE NUMBER 025, WHICH MEANS AN
0159 3E00 OTPTO: MVI A,0H ; EXPONENT Follows
015B 1E01 MVI E, 1 ; LOAD B WITH THE VALUE 360 (CODE FOR BLANK)
015D D304 OTNUM: OUT 4 ; COMPARE B WITH THE ACCUMULATOR, TEST FOR
015F CDD201 CALL CHECK ; BLANK IN RHO+9
0162 D305 OUT 5 ; JUMP TO CHK1 IF THE RESULT IS ZERO
0164 CA7101 JZ CHK1 ; OUTPUT ACCUMULATOR TO DEVICE 4
0166 C000 CALL SUBROUTINE CHECK ; 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 OTPT0 ;JUMP TO OTPT0
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  CABA01  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
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,RH0+2 ;LOAD H L REGISTER PAIR WITH RH0+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,RHO+3600 ;LOAD B WITH 360Q (CODE FOR SPACE)
01BE  B8  CMP B ;COMPARE B WITH ACCUMULATOR
01BF  CACA01  J2 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
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
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   07   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  C9   RET  ; RETURN
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
<th>Beta Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0118 75</td>
<td></td>
<td>BI1: DB 165Q</td>
<td>( \beta_1 = 0.000247 )</td>
</tr>
<tr>
<td>0119 01</td>
<td></td>
<td>DB 161Q</td>
<td></td>
</tr>
<tr>
<td>011A 7F</td>
<td></td>
<td>DB 177Q</td>
<td></td>
</tr>
<tr>
<td>011B C8</td>
<td></td>
<td>DB 310Q</td>
<td></td>
</tr>
<tr>
<td>011C 77</td>
<td></td>
<td>BI2: DB 167Q</td>
<td>( \beta_2 = 0.001385 )</td>
</tr>
<tr>
<td>011D 35</td>
<td></td>
<td>DB 165Q</td>
<td></td>
</tr>
<tr>
<td>011E 88</td>
<td></td>
<td>DB 210Q</td>
<td></td>
</tr>
<tr>
<td>011F E4</td>
<td></td>
<td>DB 344Q</td>
<td></td>
</tr>
<tr>
<td>0120 77</td>
<td></td>
<td>BI3: DB 167Q</td>
<td>( \beta_3 = 0.001222 )</td>
</tr>
<tr>
<td>0121 20</td>
<td></td>
<td>DB 165Q</td>
<td></td>
</tr>
<tr>
<td>0122 87</td>
<td></td>
<td>DB 203Q</td>
<td></td>
</tr>
<tr>
<td>0123 83</td>
<td></td>
<td>DB 203Q</td>
<td></td>
</tr>
<tr>
<td>0124 78</td>
<td></td>
<td>BI4: DB 170Q</td>
<td>( \beta_4 = 0.002645 )</td>
</tr>
<tr>
<td>0125 20</td>
<td></td>
<td>DB 170Q</td>
<td></td>
</tr>
<tr>
<td>0126 57</td>
<td></td>
<td>DB 170Q</td>
<td></td>
</tr>
<tr>
<td>0127 8C</td>
<td></td>
<td>DB 170Q</td>
<td></td>
</tr>
<tr>
<td>0128 76</td>
<td></td>
<td>BI5: DB 166Q</td>
<td>( \beta_5 = 0.000832 )</td>
</tr>
<tr>
<td>0129 5A</td>
<td></td>
<td>DB 166Q</td>
<td></td>
</tr>
<tr>
<td>012A 1A</td>
<td></td>
<td>DB 166Q</td>
<td></td>
</tr>
<tr>
<td>012B 92</td>
<td></td>
<td>DB 222Q</td>
<td></td>
</tr>
<tr>
<td>012C 74</td>
<td></td>
<td>BI6: DB 164Q</td>
<td>( \beta_6 = 0.000169 )</td>
</tr>
<tr>
<td>012D 31</td>
<td></td>
<td>DB 164Q</td>
<td></td>
</tr>
<tr>
<td>012E 35</td>
<td></td>
<td>DB 164Q</td>
<td></td>
</tr>
<tr>
<td>012F 97</td>
<td></td>
<td>DB 227Q</td>
<td></td>
</tr>
<tr>
<td>0130 91</td>
<td></td>
<td>CONV: DB 221Q</td>
<td>( \text{CONVERSION FACTOR FROM ABSOLUTE} )</td>
</tr>
<tr>
<td>0131 43</td>
<td></td>
<td>DB 103Q</td>
<td>( \text{REACTIVITY TO PCM} )</td>
</tr>
<tr>
<td>0132 50</td>
<td></td>
<td>DB 120Q</td>
<td></td>
</tr>
<tr>
<td>0133 00</td>
<td></td>
<td>DB 000Q</td>
<td></td>
</tr>
<tr>
<td>0134 73</td>
<td></td>
<td>PNLF: DB 163Q</td>
<td>( \text{PROMP NEUTRON LIFETIME} = 0.0001 )</td>
</tr>
<tr>
<td>0135 51</td>
<td></td>
<td>DB 121Q</td>
<td></td>
</tr>
<tr>
<td>0136 87</td>
<td></td>
<td>DB 267Q</td>
<td></td>
</tr>
<tr>
<td>0137 16</td>
<td></td>
<td>DB 026Q</td>
<td></td>
</tr>
</tbody>
</table>
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 BYTE 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  FLXI:  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

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0007</td>
<td>ABS</td>
<td>0250</td>
<td>AD</td>
<td>02D7</td>
<td>B</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BI1</td>
<td>0118</td>
<td>BI2</td>
<td>011C</td>
<td>BI3</td>
<td>0120</td>
<td>BI4</td>
<td>0124</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BI5</td>
<td>0128</td>
<td>BI6</td>
<td>012C</td>
<td>BLANK</td>
<td>0145</td>
<td>C</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALC</td>
<td>1400</td>
<td>CHECK</td>
<td>01DA</td>
<td>CHK1</td>
<td>0171</td>
<td>CHS</td>
<td>0240</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHSGN</td>
<td>00AD</td>
<td>CMPAR</td>
<td>0043</td>
<td>CONV</td>
<td>0130</td>
<td>D</td>
<td>0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>01E7</td>
<td>DIV</td>
<td>0284</td>
<td>DMANT</td>
<td>130A</td>
<td>E</td>
<td>0003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCH</td>
<td>0072</td>
<td>FLG</td>
<td>00A1</td>
<td>FLT</td>
<td>04FF</td>
<td>FLX1</td>
<td>1321</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLX2</td>
<td>1325</td>
<td>H</td>
<td>0004</td>
<td>HMEM</td>
<td>0091</td>
<td>HOLD</td>
<td>1310</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INP</td>
<td>054A</td>
<td>INPUT</td>
<td>0028</td>
<td>INT</td>
<td>022F</td>
<td>INTRO</td>
<td>0024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0005</td>
<td>LAM1</td>
<td>0100</td>
<td>LAM2</td>
<td>0104</td>
<td>LAM3</td>
<td>0108</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAM4</td>
<td>010C</td>
<td>LAM5</td>
<td>0110</td>
<td>LAM6</td>
<td>0114</td>
<td>LOD</td>
<td>026E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LODE</td>
<td>0366</td>
<td>M</td>
<td>0006</td>
<td>MEM</td>
<td>1301</td>
<td>MEM1</td>
<td>1300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINUS</td>
<td>0144</td>
<td>MUL</td>
<td>028C</td>
<td>NEGAT</td>
<td>00AA</td>
<td>NTH10</td>
<td>0140</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONE</td>
<td>0138</td>
<td>OTNUM</td>
<td>015D</td>
<td>OTPT0</td>
<td>0159</td>
<td>OTPT2</td>
<td>0180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTPT3</td>
<td>01A7</td>
<td>OTPT4</td>
<td>019E</td>
<td>OU</td>
<td>060C</td>
<td>OUTPT</td>
<td>0150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PART</td>
<td>037D</td>
<td>PER</td>
<td>1306</td>
<td>PNLF</td>
<td>0134</td>
<td>PSW</td>
<td>0006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES</td>
<td>1302</td>
<td>RHO</td>
<td>1314</td>
<td>RWMEM</td>
<td>0146</td>
<td>SB</td>
<td>0204</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGST</td>
<td>1329</td>
<td>SIGN</td>
<td>0189</td>
<td>SP</td>
<td>0006</td>
<td>START</td>
<td>060E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STR</td>
<td>023E</td>
<td>STRIN</td>
<td>034E</td>
<td>TH100</td>
<td>013C</td>
<td>TRACK</td>
<td>0087</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td>009C</td>
<td>WOUT</td>
<td>01E1</td>
<td>ZERO</td>
<td>0102</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

295
The vita has been removed from the scanned document
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.