Annual Report
Grant No. NAG-1-349

DIGITAL CONTROL SYSTEM FOR
SPACE STRUCTURE DAMPERS

Submitted to:
National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23665

Attention: Dr. Garnett C. Horner
SDD, MS 230

Submitted by:
J. K. Haviland
Professor

Department of Mechanical and Aerospace Engineering
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF VIRGINIA
CHARLOTTESVILLE, VIRGINIA 22901

Report No. UVA/528224/MAE86/105
September 1985
ABSTRACT

This is the final report of a two-year study of digital control systems for space structural dampers, or more specifically, for proof-mass dampers or actuators. Previously, a proof-mass actuator had been developed, of which twelve had been delivered to NASA, and analog and digital control systems had been developed in prototype form. Under the first year of the present study, a Z80 controller was developed, slaved to a TRS80 microcomputer. During the final year, which is covered in this report, a digital controller was developed using an SDK-51 System Design Kit, which incorporates an 8031 microcontroller. As part of this study, the necessary interfaces were installed in the wire-wrap area of the SDK-51 and a pulse-width modulator was developed to drive the coil of the actuator. Also, control equations were developed, using floating-point arithmetic. The design of the digital control system is emphasized in this report, and it is shown that, provided certain rules are followed, an adequate design can be achieved. It is recommended that the so-called w-plane design method be used, and that the time elapsed before output of the up-dated coil-force signal be kept as small as possible. However, the cycle time for the controller should be watched carefully, because very small values for this time can lead to digital noise.

ACKNOWLEDGEMENT

The gift by the INTEL Corporation of an SDK-51 System Design Kit is gratefully acknowledged. Without this gift, much of the work reported here could not have been attempted.
# TABLE OF CONTENTS

## INTRODUCTION
- Discussion
- Equipment
- Work on the 8051 Series

## SDK-51 DEVELOPMENT BOARD
- Description
  - Comparison of 8051 with Z80
    - Advantages of 8051 Series
    - Advantages of Z80

## DERIVATION OF CONTROL EQUATIONS
- Floating Point Subroutines
- Digital Program by Rectangular Rule
  - Difference Equations for $P_1$-$D$
  - Implementation of Program
  - Plots of Real Damping and Response Amplitude
  - Timing
- Digital Program by $w$-Plane Analysis
  - Design in the $w$-Plane
  - Derivation of the Difference Equations
- System with Minimum Delay
  - Numerical Accuracy
  - Minimum Delay
  - Difference Equations for Minimum Delay Case
  - Plots of Real Damping

## SUMMARY
- Controller Design
- Digital Control Equations
- Floating-Point Calculations
- Pulse Width Modulation
- Word Length
- Future Development

## CONCLUSIONS AND RECOMMENDATIONS

## REFERENCES

## APPENDIX A - EXPERIMENTAL PROGRAM FOR SDK-51 BOARD

## APPENDIX B - SCHEMATICS
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Proof-Mass Actuator Section</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Proof-Mass Actuator with Proximeter</td>
<td>3</td>
</tr>
<tr>
<td>4.</td>
<td>Bode Plots of Synthetic Spring: Analog</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Real Damping, etc: Low Synthetic Stiffness</td>
<td>23</td>
</tr>
<tr>
<td>6.</td>
<td>Norm. of Response, etc: Low Synthetic Stiffness</td>
<td>24</td>
</tr>
<tr>
<td>7.</td>
<td>Real Damping, etc: Low Accelerometer Gain</td>
<td>25</td>
</tr>
<tr>
<td>8.</td>
<td>Norm. of Response, etc: Low Accelerometer Gain</td>
<td>26</td>
</tr>
<tr>
<td>9.</td>
<td>Real Damping, etc: Default Values</td>
<td>27</td>
</tr>
<tr>
<td>10.</td>
<td>Norm. of Response, etc: Default Values</td>
<td>28</td>
</tr>
<tr>
<td>11.</td>
<td>Real Damping, etc: High Accelerometer Gain</td>
<td>29</td>
</tr>
<tr>
<td>12.</td>
<td>Norm. of Response, etc: High Accelerometer Gain</td>
<td>30</td>
</tr>
<tr>
<td>13.</td>
<td>Real Damping, etc: High Synthetic Stiffness</td>
<td>31</td>
</tr>
<tr>
<td>14.</td>
<td>Norm. of Response, etc: High Synthetic Stiffness</td>
<td>32</td>
</tr>
<tr>
<td>15.</td>
<td>Digital Filter with Analog Plant. Effect of Delay</td>
<td>34</td>
</tr>
<tr>
<td>17.</td>
<td>Bode Plot of Synthetic Damper: Digital</td>
<td>38</td>
</tr>
<tr>
<td>18.</td>
<td>Bode Plots of Synthetic Spring: Digital</td>
<td>39</td>
</tr>
<tr>
<td>19.</td>
<td>Digital Filter with Analog Plant: Immediate</td>
<td>45</td>
</tr>
<tr>
<td>20.</td>
<td>Real Damping, etc: (c=10Ns/m, mT₀=0)</td>
<td>48</td>
</tr>
<tr>
<td>21.</td>
<td>Real Damping, etc: (c=10Ns/m, mT₀=4096 musecs)</td>
<td>49</td>
</tr>
<tr>
<td>22.</td>
<td>Real Damping, etc: (c=80Ns/m, mT₀=0)</td>
<td>50</td>
</tr>
<tr>
<td>23.</td>
<td>Real Damping, etc: (c=80Ns/m, mT₀=4096 musecs)</td>
<td>51</td>
</tr>
<tr>
<td>B1.</td>
<td>Sheet 1: Drivers</td>
<td>82</td>
</tr>
<tr>
<td>B2.</td>
<td>Sheet 2: A/D Converter</td>
<td>83</td>
</tr>
<tr>
<td>B3.</td>
<td>Sheet 3: A/D Trigger and Clock</td>
<td>84</td>
</tr>
<tr>
<td>B4.</td>
<td>Sheet 4: Channel Select</td>
<td>85</td>
</tr>
<tr>
<td>B5.</td>
<td>Sheet 5: Analog Input Port (Typical)</td>
<td>86</td>
</tr>
<tr>
<td>B6.</td>
<td>Sheet 6: PWM Board</td>
<td>88</td>
</tr>
</tbody>
</table>
DEFINITIONS

\( a_0, a_1 \) = Coefficients of polynomial

\( a_F, A_F \) = Structural acceleration

\( b_1 \) = Coefficient of polynomial

\( c \) = Design damping (Ns/m)

\( D(s), \text{ etc.} = \text{Transfer function} \)

\( F \) = Coil force

\( g \) = Acceleration of gravity (9.81 m/s\(^2\))

\( G \) = Analog gain

\( G^* \) = Digital gain

\( H(s), \text{ etc.} = \text{Transfer function} \)

\( H_C(s), \text{ etc.} = \text{Complex damping} \)

\( I_n \) = Integer form of \( n \)

\( k \) = Integer time-interval variable

\( k_A, \text{ etc.} = \text{Digital gain terms} \)

\( k_{\text{max}} \) = Maximum synthetic stiffness

\( k_s \) = Synthetic stiffness (N/m)

\( K \) = Analog gains used in calibration of system

\( m \) = Integer time-count for data output

\( M \) = Proof-mass (kg)

\( n \) = Integer cycle-time count for calculation cycle

\( R_C(s) \) = Response amplitude ratio

\( s \) = Laplace variable

\( t \) = Time variable

\( T \) = Calculation cycle time

\( T_0 \) = Basic time interval (256 microseconds)

\( u, U \) = Control state or output variable

\( w \) = Transform variable

\( x, X \) = Input variable

\( z \) = Transform variable

\( Z\{\} \) = z-transform equivalent of a Laplace transform

\( \phi_M \) = Phase margin

\( \gamma \) = Lag to lead frequency ratio

\( \zeta \) = Accelerometer gain parameter

\( \nu \) = \( w \)-Plane frequency

\( \omega \) = \( s \)-Plane frequency

Subscripts:

\( A \) = Accelerometer

\( B \) = Component of damping equation

\( c \) = Relating to damping

\( C \) = Coil
D = Relative proof-mass motion
L = LVDT
P = Proximeter
s = Relating to stiffness
V = Component of synthetic stiffness equation
INTRODUCTION

Discussion: This report covers the second year of a study of space structure damping under NASA Grant No. NAG-1-349, following Proposal No. MAE-NASA-2548-83 (1). Earlier, a general study of possible damper configurations had been reported under NASA Grant No. NAG-1-137-1 (2). Following that work, purchase order No. L46164B had been received from NASA for the design and construction of twelve proof-mass actuators, also referred to as space structure dampers. A sectioned assembly drawing for this design is shown as Figure 1. During these last two years, Mr. Michael Mallette, a doctoral candidate, has worked on the development of control laws under a NASA student fellowship. His dissertation is imminent. Under the present two-year grant, earlier reports (3,4) have covered design of the proof-mass actuator, and development of analog and Z80 controllers. The work reported here covers development of an 8051 series controller exclusively.

Equipment: The work on the 8051 series controllers was aided considerably by the donation of an SDK-51 System Design Kit from the INTEL Corporation. Also, two of the twelve NASA owned proof-mass actuators were obtained on loan, and were modified to take Bentley-Nevada Model 190 proximeter probes. This required two new cases, and tapered sleeves on the proof-masses, so that their position could be determined by proximeters. One of these actuators has been used by Mr. Mallette, this is shown in Figure 2 with an accelerometer which is also on loan from NASA. The other was used in the present study. It has a Sunstrand Model QA-900 accelerometer, a Bentley-Nevada 3106-2800-190 amplifier, and a
Figure 2. PROOF-MASS ACTUATOR WITH PROXIMETER
home-made pulse-width modulator (PWM) attached. A control system was built in the wire-wrap area of the SDK-51 board, as described later.

Work on the 8051 series: Work on the 8051 series controllers, which is literally an 8031, which has no internal program memory, was limited mainly to development of the system described above, and to the requisite SDK-51 programs, including two versions of the PI-D control realization first discussed in Reference 4. Behavior of the system was largely checked by simple observation, relying on Mr. Mallette’s experience for further insight into its behavior. The following report covers a description of the controller hardware which was developed, and of the control program, together with computer predictions of the real damping vs. frequency, and of the relative amplitude of motion of the proof-mass within its case. The long general purpose SDK-51 program which was used is listed in Appendix A.

SDK-51 DEVELOPMENT BOARD

Description

An SDK-51 Development Board was obtained as a gift from the INTEL Corporation. Although it is designed for teaching the 8051 language, a wire-wrap area is provided for user experiments. This area was used to configure a controller for the proof-mass damper. Components in this area include four analog input ports (two populated), an A/D converter, and pulse-width modulated (PWM) outputs. An overall schematic of this system is shown in Figure 3,
FIGURE 3. PROOF MASS ACTUATOR: DIGITAL LOGIC

Coil Power

PWM Board

Driver

RH Coil Pin 3.5

LH Coil Pin 3.4

Pin 3.3

Proximeter

Accelerometer

NC

NC

A/D Converter

Bidirectional Driver

A/D 8-Bits

Pins 1, 0, 1, 2, 3

12 MHz

Port 1

SDK - 51

Port 3

Main Power

12V

5V

GND

RS 232

Printer

Cassette Recorder
and logic diagrams are given in Appendix B.

As presently configured, 12 pins on two ports of the 8031 are used. These are all eight pins of port 1, and pins 3.3, 4, and 5 of port 3. In addition, pin 3.0 has been programmed temporarily to indicated completion of digital calculations as an oscilloscope signal, but this could easily be discontinued. Pins 1.0, 1, 2, and 3 are connected to a transceiver, and can be used for output, otherwise, port 1 is used to read the A/D. Pin assignments are as follows:

<table>
<thead>
<tr>
<th>Port 1 (Input)</th>
<th>Read A/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1.0 (Output)</td>
<td>Input channel selection</td>
</tr>
<tr>
<td>Pin 1.1</td>
<td>&quot;  &quot;  &quot;  &quot;</td>
</tr>
<tr>
<td>Pin 1.2</td>
<td>Trigger A/D</td>
</tr>
<tr>
<td>Pin 1.3</td>
<td>Enable A/D</td>
</tr>
<tr>
<td>Pin 3.0 (Output)</td>
<td>Temporary oscilloscope signal.</td>
</tr>
<tr>
<td>Pin 3.3</td>
<td>Sets transceiver to output when high.</td>
</tr>
<tr>
<td>Pin 3.4</td>
<td>Sets L.H. end of coil to high voltage.</td>
</tr>
<tr>
<td>Pin 3.4</td>
<td>Sets R.H. end of coil to high voltage.</td>
</tr>
</tbody>
</table>

Four analog inputs were originally designed, but two will not be populated (LVDT and signal generator) until requirements for a slaved 8031 have been determined. The two which have been populated are #0, proximeter, and #1, accelerometer. The four inputs are selected by the outputs of Pins 1.0, and 1, through half of a 74LS139 decoder, and an LF13332 analog switch, with a TL087 high speed operational amplifier to improve output impedance. The selected signal is converted directly to 2's com-
plement eight-bit form using a DAC0800 D/A and a DM2502 successive approximation register, with a LM361 high speed comparator to compare the two signals. Timing comes from the 12MHz crystal on the SDK-51 board, divided by powers of two in a 74LS163, as selected by jumpers. Signals are synchronized by a dual D-flip-flop, in a one-and-one-only configuration. A 75451 driver is used for the PWM output; the actual PWM function is carried out on a separate board attached to the proof-mass damper. This board consists of two pairs of Darlington transistors (NTE261 and NTE262), one pair is attached to each end of the coil, their bases are driven by 2N3904 transistors, which are themselves driven by 4N28 optoelectrical transistors from the PWM signals. With this arrangement, about +1 to -1 Amperes can be produced in the 8.5 Ohm coil. However, an important feature of this arrangement is that there is no coil current when both PWM signals are equal. Thus the coil does not heat up when the proof-mass damper is quiescent.

Comparison of 8051 with Z80:

The work reported here, in conjunction with the work reported for the previous year in Reference 4, affords an opportunity to compare the 8051 with the Z80, in the following ways:

Advantages of 8051 Series:

Multiplication: Only available on the 8051 series.

Division: Only available on the 8051 series, but of dubious value because it only produces the integer part of the quotient.
On-Board Timer: There are two onboard timers on the 8051 series, both with interrupts, whereas the same functions have to be provided by hardware with the Z80 (the 8052 series has an additional timer).

Interrupt Priority: There are two levels of interrupt priority, with a total of five interrupts (two timer, two general, and one serial). Again, this arrangement must be provided by hardware for the Z80.

Internal RAM: Internal RAM is provided on the 8051 series, with one page of byte addresses, plus another page of bit addresses covering part of the same field. One half page is devoted in each case to special function registers. This provides computing power unique to the 8051 series.

Internal UART: An internal UART on the 8051 series makes master-slave arrangements relatively simple. A third timer can be used to provide the needed Baud rate, or very high speed serial data exchange can be obtained using the clock-timer. In the master-slave arrangement, several slaves can be addressed individually.

Advantages of Z80:

16-Bit Arithmetic: Many operations can be carried out with 16 bits, compared to only 8 bits on the 8051 series.

BUSREQ: This feature of the Z80 permits a single slave arrangement in which the memory space of the slave is relatively easy to address. This proved to be a great
advantage in the development of the Z80 system.

**Vectored Interrupts:** The Z80 can receive address vectors for interrupts, which simplifies the selection of different programs when running as a slave.

**IN/OUT:** The separate mapping of in/out memory space was an advantage, because these instructions could be decoded, and could be used to trigger operations such as read A/D. The same functions are obtained on the 8051 series by SETB and CLR instructions to the port pins.

**The Z80 is Used in Small Computers:** The fact that the Z80 is a well-known and popular computer chip was to its advantage in last year’s work because it was relatively simple to use the Radio Shack Model 1 computer as a development system. A comparable system for the 8051 series, although considerably better, costs about ten times as much.

**DERIVATION OF DIGITAL CONTROL EQUATIONS**

**Floating Point Subroutines**

Since the INTEL 8051 series controller can only execute eight bit arithmetic, unlike the Z80 which can handle many sixteen bit operations, an early decision was made to use a sixteen (16) bit floating point format, with a signed seven bit mantissa and exponent, as follows:

```
Bit #  15  14  13  12  11  10  9  8  7  6  5  4  3  2  1  0
    sign of mantissa  sign of exponent
```
Thus +1 becomes .4OX01, and -1 becomes .8OX00, where X stands for exponent, and the decimal point means that the mantissa is fractional. We cannot use E for the exponent, as with decimals, because it is a hexadecimal digit.

The following subroutines are available:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>NEGATE</td>
</tr>
<tr>
<td>SUBTRACT</td>
<td>MOVE IN MEMORY</td>
</tr>
<tr>
<td>MULTIPLY</td>
<td>FIXED TO FLOATING FROM MEM.</td>
</tr>
<tr>
<td>STORE ABOVE RESULTS</td>
<td>FIXED TO FLOATING FROM ACCUM.</td>
</tr>
<tr>
<td>IN MEMORY FLOATING</td>
<td>F FLOATING TO FIXED</td>
</tr>
</tbody>
</table>

These subroutines are identified in the listing supplied in the Appendix. Results of all operations except FLOATING TO FIXED are normalized by shifting ones into positive numbers and zeros into negative numbers, thus:

- .00X00 becomes .7FXF9
- .3FX00 " .7FXFF
- .FFX00 " .80XF9
- .OX00 " .80XFF

Sometimes, the application of an operation and its inverse, such as ADD and SUBTRACT, or NEGATE NEGATE, results in a change in the last bit. Also, if exponent overflow occurs, it is replaced by X7F or X80, as appropriate, but the mantissa is meaningless. Further, the difference of two equal numbers leaves a zero mantissa, which is then normalized as a positive number. Thus the
final result has a mantissa of .7F, while the exponent is reduced by 7.

The program in Appendix A includes a floating point calculator simulation program, similar to the reverse Polish system on the Hewlett-Packard calculators. The program includes all of the subroutines listed, with the exception of FIXED TO FLOATING, which is covered by the NORMALIZE operation. In addition, numbers can be entered into the display on the SDK 51 board, and the command ENTER can then enter them into a three tier stack, while READ can bring them back into the display. Operations on two numbers involve the SDK 51 display and the first number on the stack, the result is displayed, and the stack is moved down by one. The calculator program was written to permit development of the floating point subroutines, and to make it easier to calculate parameters to be used in experimental programs. It includes provisions for inserting floating-point parameters into data memory for use in the control programs.

Often it is necessary to find the floating point equivalent of a decimal number for insertion into the controller program. The following procedure was found to be useful:

(a) Express in form $M \times 2^E$

(b) Convert to form $.MH \times 2^{(EH+7)}$

(c) Write in form $.MHX(EH+7)$ for entry onto board.

(d) Write in form $MH,(EH+7)$ for entering into memory.

Note: $M,E$ are decimal integers, with $M$ between 63 and 127,
while MH and EH are hexadecimal equivalents to 2 places.

Example: Convert 0.0287

(a) \(0.0287 = 117 \times 2^{-12}\)

(b) \(= 0.75 \times 2^{-12+7} = 0.75 \times 2^{FBH}\)

(c) \(= 0.75XFB\)

(d) \(= 75H, FBH\)

To find the decimal equivalent, this process is reversed:

(a) Express in form \(MH \times 2^{EH-7}\)

(b) Convert to form \(M \times 2^{E-7}\)

(c) Evaluate

Example: Convert .75XFB

(a) \(.75XFB = 75H \times 2^{FBH-7} = 75H \times 2^{F4H}\)

(b) \(= (7 \times 16 + 5) \times 2^{-12} = 117 \times 2^{-12}\)

(c) \(= 117/4096 = 0.0286\)

**Digital Program by Rectangular Rule**

The program shown in Appendix A is based on the rectangular rule of integration, but such refinements as zero-order-hold and computational delay have been omitted. The program corresponds very closely to the P1-D program of Reference 1, except that the
16-bit arithmetic of the Z80 has been replaced by the floating point arithmetic described in the preceding paragraph, and the divide by powers-of-two operations have been replaced by full multiplications.

The system to be investigated is shown in block diagram form in Figure 4. Some changes in notation have been made relative to Reference 4, mainly the replacement of number subscripts to avoid confusion with state-space notation, and a redefinition of $H_A$.

From Figure 4:

$$F(s) = H_A(s)A_F(s) - H_P(s)X_D(s)$$

while, from the dynamics of the proof mass

$$F(s) = MA_F(s) + Ms^2X_D(s)$$

The signal generator input, $x_S$, has not been included in these equations. We can now develop two functions which are of considerable importance in the evaluation of damper performance:

$$H_C(s) = sF(s)/A_F(s)$$

$$= s\{(H_A(s) + H_P(s)/s^2)/(1 + H_P(s)/Ms^2)\}$$

$$R_C(s) = s^2X_D(s)/A_F(s)$$

$$= -(1 - H_A(s)/M)/(1 + H_P(s)/Ms^2)$$

where $H_C$ is the complex damping, whose real part must be positive at any frequency at which energy is to be absorbed, and $R_C$ is the ratio of the proof-mass amplitude to that of the structure. For example, if its norm is 2, then the proof mass will just hit the
FIGURE 4. PROOF-MASS ACTUATOR CONTROLS: ANALOG
stops of a one inch stroke damper when the structural double-amplitude reaches a half inch.

It has been found that satisfactory values for \( \text{Re}\{H_C\} \) and \( \text{Norm}(H_C) \) can be obtained if the following rules are followed:

(1) There is a positive input to the A/D (this may mean a negative voltage, because most A/D’s invert) when there is an acceleration directed from the structure to the damper, i.e., a positive acceleration.

(2) There is a positive input to the A/D when the proof-mass is against the structure, i.e., a negative displacement.

(3) A +/- 1g accelerometer range exactly covers the full input range to the A/D. (referred to as +/- 1 here, rather than to a range of voltages).

(4) The full range of proof-mass travel exactly covers the full input range to the A/D.

(5) The force exerted on the proof mass, when the accelerometer is attached, exactly balances its weight component.

(6) The synthetic spring stiffness, \( k_s \), is a fraction of the maximum available value, \( k_{\text{max}} \), chosen to give suitable centering behavior.

(7) At high frequency, \( H_A \) should approach the real value, \( c \), of the required design damping.

(8) At high frequency, \( H_p \) should approach zero.

(9) The open loop gain, \( H_p/M_s^2 \), of the synthetic spring circuit should have an adequate phase margin.

Rules 1 and 2 ensure the correct polarity, and permit a simple evaluation of the damper using the DEMO modes described in the
Appendix. When this polarity is correct, the damper exhibits simple spring behavior or a tendency to remain centered when the damper assembly is tilted, according to which DEMO program is selected.

Rules 3 and 4 permit calibration of the system by one of the following methods:

(a) Direct monitoring of the A/D inputs with a voltmeter.
(b) Use of the DISPLAY subroutine described in the Appendix which displays the input in 2's complement hexadecimal form on the SDK-51 board.
(c) Use of the appropriate DEMO program together with monitoring of the output to the coil.

Applying these rules, we have:

\[ K_A = \frac{1}{1g} = \frac{1}{9.8} = 0.1020 \text{ s}^2/\text{m} \]

\[ K_P = \frac{40 \text{ in} \cdot \text{s}/\text{m}}{1/2 \text{ inch amplitude}} = 80 \text{ m}^{-1} \]

Rules 3 and 7 are satisfied if \( H_A \) has the form:

\[ H_A(s) = \frac{2M\zeta}{1+s/\omega_A} \]

with

\[ \zeta = \frac{c}{2M\omega_A} \]

while rule 5 is satisfied when \( \zeta = 1/2 \).

Rules 4, 6 and 8 are satisfied if \( H_P(s) \) has the form:

\[ H_P(s) = k_s \frac{(1+s/\omega_V)}{(1+s/\omega_P)} \]
so that the open-loop transfer function is:

\[ \frac{H_p(s)}{M s^2} = \frac{(\omega_N^2/s^2)(1+s/\omega_V)}{(1+s/\omega_P)} \]

where:

\[ \omega_N^2 = k_s/M \]

and:

\[ k_s < k_{\text{max}} = K_P K_C \]

From several measurements on the present damper design, when the current is adjusted to range from -1 to +1 Amps. over the full range of digital input:

\[ K_C = 1.92 \text{ N} \]

thus:

\[ k_{\text{max}} = K_P K_C \]

\[ = (80 \text{ m}^{-1})(1.93 \text{ N}) = 155 \text{ N/m} \]

Rule 9 is satisfied if suitable values are picked for the two break frequencies in \( H_p(s) \). Using the Bode plot of Figure 5, and designing for a phase margin of \( \phi_M \):

\[ \gamma = \frac{\omega_P}{\omega_V} \]

\[ = 1/(\tan(45 - \phi_M/2))^2 \]

where, from the geometry of the figure 5:

\[ \omega_V = \omega_N/\gamma^{1/4} \]
FIGURE 5. BODE PLOTS OF SYNTHETIC SPRING: ANALOG
\[ \omega_c = \omega_N \gamma^{1/4} \]
\[ \omega_p = \omega_N \gamma^{3/4} \]

**Difference Equations for P1-D: From Figure 4, the difference equations must provide the two filters:**

\[ H_A(s) / K_A K_C = (2\xi M / K_A K_C) / (1 + s / \omega_A) \]
\[ = G_A / (s + \omega_A) \]
\[ H_p(s) / K_p K_C = (k_s / k_{max}) (1 + s / \omega_V) / (1 + s / \omega_p) \]
\[ = (sG_V + G_p) / (s + \omega_p) \]

The digital equations for the realizations of these filters are derived using the rectangular rule as follows:

\[ x_p(k) = x_p(k) \text{ or } x_L(k) \]
\[ u_p(k) = (1 - \omega_p T) u_p(k-1) - G_p T x_p(k) \]
\[ - G_V \{ x_p(k) - x_p(k-1) \} \]
\[ u_A(k) = (1 - \omega_A T) u_A(k-1) + G_A T x_A(k) \]
\[ u(k) = u_p(k) + u_A(k) + x_S(k) \]

It may be noted that the second and third equations could be written as the two equations:

\[ u_V(k) = u_V(k-1) - \omega_p T u_p(k-1) - G_p T x_p(k) \]
\[ u_p(k) = u_V(k) - G_V x_p(k) \]

where \( u_V \) is essentially a state variable. Note that these
equations include the input $x_S$ from the signal generator, and the alternative position signal $x_L$ from the LVDT.

**Implementation of Program:** Appendix A describes a program with two modes of input. They are:

Program P: This program has default parameters, as shown below in parenthesis. New parameters can be entered, and the program can be restarted as Program Q. Values for these parameters are determined as follows:

\[
\begin{align*}
I_n &= \text{Integer value of } n \text{ used to count cycles.} \\
&= .10X00 = 16 \\
T &= \text{Time interval, musecs.} \\
&= 256n \\
&= .43XF9 = 4096 \text{ musecs} \\
\omega_A &= \text{Accelerometer break frequency, rads/sec.} \\
&= c/2\pi M \\
&=.48X06 = 36 \text{ rads/sec.} \\
\omega_P &= \text{Proximeter (or LVDT) break frequency, rads/sec.} \\
&= \gamma^{3/4} \omega_N \\
&=.5EX07 = 94.4 \text{ rads/sec} \\
G_A &= \text{Accelerometer gain.} \\
&= c/K_A K_C
\end{align*}
\]
\[
G_P = \text{Proximeter gain.}
\]

\[
G_P = k_s \frac{\omega_P}{k_{\text{max}}}
\]

\[
= \gamma^{3/4} M_{\omega_N}^{3/4} / K_P K_C
\]

\[
(= .5E05 = 23.6)
\]

\[
G_V = \text{Proximeter feedforward gain.}
\]

\[
G_V = G_P / \omega_V
\]

\[
= \gamma^{1/4} G_P / \omega_N
\]

\[
(= .40X03 = 4.0)
\]

The above equations assume that the design damping, \( c \) Ns/m, and the required synthetic spring frequency, \( \omega_N \) rads/sec., are known. Also, \( n \) must be chosen so that the program has time to complete a cycle of calculations. As for the default parameters, values for \( K_A, K_P, \) and \( K_C \) are assumed as discussed earlier, the proof mass \( M \) is 0.278 kg., \( \zeta \) is 1/2, and \( \gamma \) is 16, corresponding to a phase margin \( \phi_M \) of 62 degrees. Default values for \( n, c, \) and \( \omega_N \) are the same as for Program T discussed below.

Program T: In this program, default values are included for the following parameters, and the remainder are calculated from them. They can be entered, and the program can be restarted as Program U:

\[
n = \text{Integer value for } n \text{ in floating-point format.}
\]
\( c = \text{Design damping, Ns/m.} \)

\( (= .50 \times 10^4 = 10 \text{ Ns/m}) \)

\( \omega_N = \text{Synthetic spring natural frequency, rads/sec.} \)

\( = \sqrt{k_s/M}, \text{where } k_s = \text{design stiffness, N/m.} \)

\( (= .5 \times 10^4 = 11.8 \text{ rads/sec, i.e., } k_s = 38.7 \text{ N/m}) \)

**Plots of Real Damping and Response Amplitude:** Plots of the real damping, \( \text{Re}\{H_C\} \), and the amplitude of the response ratio, \( \text{Norm}\{R_C\} \), are supplied as Figures 6 to 15 for five values of the design damping, \( c \), three values of the design stiffness, \( k_s \), and three values of \( \zeta \). Note that the real damping goes negative at low frequencies when \( \zeta > 1/2 \). Otherwise, the damping is positive over the range of frequencies shown, and is asymptotic to the design damping, \( c \). Although the design stiffness, \( k_s \), was varied over a 16:1 range, it had relatively little effect on the damping curves. Previous investigations, using much lower values for the phase margin, have shown resonance peaks in both curves. Unfortunately, due to the choice of program for the Z80, adequate phase margins could not be used, however, the problem of resonance peaks has been solved since the introduction of floating-point arithmetic.

**Timing:** The P and T programs described in the Appendix use the #0 and #1 timer interrupt modes available on the 8051 series. The #0
Proof Mass (m) = 0.278 kg
Zeta (ζ) = 0.5
Gamma (γ) = 16
Synthetic Stiffness (k) = 9.7 N/m
= k_{max}/16
Design Damping (c) = 5, 10, 20, 40, 80 N.s/m

Figure 6. REAL DAMPING FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: LOW SYNTHETIC STIFFNESS
Figure 7. NORM OF RESPONSE RATIO 
FOR PROOF-MASS ACTUATOR 
WITH ANALOG CONTROLLER:
LOW SYNTHETIC STIFFNESS 

Proof Mass (m) = .278 kg
Zeta (ζ) = .5
Gamma (γ) = 16
Synthetic Stiffness (k) = 9.7 N/m
Design Damping (c) = 5, 10, 20, 40, 80 N.s/m 

\[ k = \frac{k_{\text{max}}}{16} \]
Figure 8. REAL DAMPING FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: LOW ACCELEROMETER GAIN

Proof Mass \( m \) = 0.278 kg
Zeta \( \zeta \) = 0.25
Gamma \( \gamma \) = 16
Synthetic Stiffness \( k \) = 38.7 N/m
Design Damping \( c \) = 5, 10, 20, 40, 80 N.s/m
**Figure 9. NORM OF RESPONSE RATIO FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: LOW ACCELEROMETER GAIN**

Proof Mass \((m)\) = 0.278 KG kg
Zeta \((\zeta)\) = 0.25
Gamma \((\gamma)\) = 16
Synthetic Stiffness \((k)\) = 38.7 N/m
\[ k_{\text{max}}/4 \]
Design Damping \((c)\) = 5, 10, 20, 40, 80 N.s/m

<table>
<thead>
<tr>
<th>FREQ. RAD/SEC.</th>
<th>0.00</th>
<th>20.00</th>
<th>40.00</th>
<th>60.00</th>
<th>80.00</th>
<th>100.00</th>
<th>120.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM OF RESP. RATIO ZETA = 0.25</td>
<td>0.60</td>
<td>0.80</td>
<td>1.00</td>
<td>1.20</td>
<td>1.40</td>
<td>1.60</td>
<td>1.80</td>
</tr>
</tbody>
</table>

\(c = 5\) N.s/m
\(c = 10\)
\(c = 20\)
\(c = 40\)
\(c = 80\)
Figure 10. REAL DAMPING FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: DEFAULT VALUES

- Proof Mass (m) = 0.278 kg
- Gamma (\(\gamma\)) = 16
- Synthetic Stiffness (k) = 38.7 N/m
- Design Damping (c) = 5, 10, 20, 40, 80 N.s/m
- Zeta (\(\zeta\)) = \(\frac{k_{max}}{4}\)
Figure 11. NORM OF RESPONSE RATIO FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: DEFAULT VALUES

Proof Mass \((m)\) = 0.278 kg
Zeta \((\zeta)\) = 0.5
Gamma \((\gamma)\) = 16
Synthetic Stiffness \((k)\) = 38.7 N/m
Design Damping \((c)\) = 5, 10, 20, 40, 80 N.s/m

\(k_{\text{max}}/4\)
Proof Mass \( m \) = .278 kg
Zeta \( \zeta \) = 1.0
Gamma \( \gamma \) = 16
Synthetic Stiffness \( k \) = 38.7 N/m
Design Damping \( c \) = 5, 10, 20, 40, 80 N.s/m

\[ c = \frac{k_{\text{max}}}{4} \]

**Figure 12.** REAL DAMPING FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: HIGH ACCELEROMETER GAIN
Figure 13. NORM OF RESPONSE RATIO FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER:
HIGH ACCELEROMETER GAIN

Proof Mass (m) = 0.278 kg
Zeta (ζ) = 1.0
Gamma (γ) = 16
Synthetic Stiffness (k) = 38.7 N/m
   = k_{max}/4
Design Damping (c) = 5, 10, 20, 40, 80 N.s/m
Proof Mass \( (m) \) = 0.278 kg
Zeta \( (\zeta) \) = 0.5
Gamma \( (\gamma) \) = 16
Synthetic Stiffness \( (k) \) = 155 N/m

Design Damping \( (c) \) = 5, 10, 20, 40, 80 N.s/m

Figure 14. REAL DAMPING FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: HIGH SYNTHETIC STIFFNESS
Figure 15. NORM OF RESPONSE RATIO FOR PROOF-MASS ACTUATOR WITH ANALOG CONTROLLER: HIGH SYNTHETIC STIFFNESS

Proof Mass \((m) = 0.278\) kg
Zeta \((\zeta) = 0.5\)
Gamma \((\gamma) = 16\)
Synthetic Stiffness \((k) = 155\) N/m
Design Damping \((c) = 5, 10, 20, 40, 80\) N.s/m
interrupt is encountered every 256 muscs, and is used to reset the pulse-width-modulator (PWM), while the #1 interrupt is used to set the pulse width of the PWM. The counter n is used to set the value for T, which is equal to 256n muscs. Typically, n has been 16, resulting in a value for T of 4096 muscs. Lower values, such as 3072 muscs, have been used, but, according to Reference 5, an excessively small value for T can cause problems with round-off noise, even if the digital calculations are completed in time. Both the P and T programs update their output at the end of their cycle, so that there is a full-cycle time-delay of T.

Digital Program by w-Plane Analysis

The control equations described in the preceding paragraphs, and contained in the program described in Appendix A, do not allow for a zero-order-hold, or for the time delay T which is inherent in the method of calculation. Typically, an analog plant driven by a digital filter can be represented by Figure 16a, if there is no time delay, and by Figure 16b if there is a time delay, following methods described in the literature, such as for example, in References 5 or 6. The zero-order hold has a z-transform equal to (z-1)/z, thus the open loop transfer functions are:

for no delay:

\[ U(z)/X(z) = H(z)((z-1)/z)Z\{G(s)/s}\]

and for a delay of T:

\[ U(z)/X(z) = H(z)((z-1)/z^2)Z\{G(s)/s}\]
Figure 16: Digital Filter with Analog Plant: Effect of Delay

(a) No Delay

PLANT \( z \cdot \frac{g(s)}{s} \)

\( \frac{z^0 \cdot h}{1 - z^{-1}} \)

FILTER \( H(z) \)

\( X(z) \)

(b) With Delay

PLANT \( z \cdot \frac{g(s)}{s} \)

\( \frac{z^0 \cdot h}{1 - z^{-1}} \)

DELAY \( \frac{-1}{z} \)

FILTER \( H(z) \)

\( X(z) \)
where $Z$ represents the $z$-transform equivalent of a Laplace transform. A block diagram of the damper, corresponding to Figure 4, but incorporating the concepts shown in Figure 16, is shown in Figure 17. The input is represented as an acceleration $a_p$, so that the $z$-transform derived below represents $F(z)/A_p(z)$, which can be readily converted to the form $H_C$. First, two equations are derived from Figure 16:

\[
F(z) = \{H_A(z)A_p(z) - H_p(z)X_D(z)\}/z
\]

\[
X_D(z) = \{F(z)/M-A_p(z)\}((z-1)/z)Z\{1/s^3\}
\]

so that the overall transfer-function can be written as:

\[
F(z)/A_p(z) = \{D_A(z) + MD_p(z)\}/\{1 + D_p(z)\}
\]

Note that:

\[
Z\{1/s^3\} = T^2z(z+1)/2(z-1)^3
\]

then:

\[
D_A(z) = H_A(z)/z
\]

\[
D_p(z) = H_p(z)((z-1)/z^2)Z\{1/Ms^3\}
\]

\[
= H_p(z)(T^2/2M)(z+1)/z(z-1)^2
\]

The $w$-transform maps the $z$-plane into a space which more nearly resembles the $s$-plane. In fact, as $s$ moves along the imaginary axis from zero to the Nyquist frequency, as represented by $s=j\omega$, $w$ moves along the real axis from zero to infinity, as represented by $w=j\nu$. The substitution for $z$ is:
FIGURE 17. PROOF-MASS ACTUATOR CONTROLS: DIGITAL
\[ z = \frac{1 + wT/2}{1 - wT/2} \]

while \( \nu \) is given by:

\[ \nu = (2/T) \tan(\omega T/2) \]

the inverse being given by:

\[ \omega = \frac{2}{T} \arctan(\nu T/2) \]

so that the \( D(w) \) transfer functions become:

\[ D_A(w) = H_A(w)\frac{1 - wT/2}{1 + wT/2} \]

\[ D_p(w) = H_p(w)\frac{1 - wT/2}{Mw^2(1 + wT/2)} \]

**Design in the \( w \)-Plane:** The rules for designing in the \( w \)-plane are almost identical to those for designing in the \( s \)-plane. The familiar Bode plots can be made, the only difficulty being that the \( w \)-transfer functions are often not of minimum phase form. This means that the phase cannot be inferred from the Bode plot alone, but this is not a problem of any significance. The Bode plots for \( D_A(w) \) and \( D_p(w) \) are shown in Figures 18 and 19. In the following discussion, \( T=4096 \) musecs, so that the Nyquist frequency is \( \pi/T = 767 \) rads/sec in the \( s \)-plane. Proceeding with the design in the \( w \)-plane, using almost identical methods to those used in the \( s \)-plane, but taking \( \xi = 1/2 \), we find that:

\[ H_A(w) = M/\{1+w/\nu_A\} \]

\[ H_p(w) = k_s\{1+w/\nu_V\}/(1+w/\nu_p) \]

Using the previous default values of \( n (=16) \), \( c (=10 \text{ Ns/m}) \),
FIGURE 18. BODE PLOT OF SYNTHETIC DAMPER: DIGITAL
FIGURE 19. BODE PlOTS OF SYNTHETIC SPRING: DIGITAL
and $k_s$ (=38.7 N/m), and taking $\gamma$=16, we find values for $\nu_A$, $\nu_P$, and $\nu_V$ which are numerically equal to the corresponding $\omega$ values found for the analog design case. Thus the actual $\omega$ values have decreased according to the transformation law given above.

The apparent -1 break at $\nu=2/T=488.3$ rads/sec in the Bode plot of $D_p(w)$ is misleading, because it is a multiple phase break and introduces additional phase lags of 90 degrees in the case of $D_A$ and 135 degrees in the case of $D_p$. For $D_A$ to have the high frequency performance characteristic of a damper, it should lag 90 degrees. However, calculations show that the lag is 180 degrees, so that real damping is zero, at $\nu=525$ rads/sec, corresponding to a true frequency of 401 rads/sec. Again, although the value for the gain at the design crossing frequency $\nu_C$ (= 23.6 rads/sec.) of the open-loop transfer function $D_p$ is calculated to be 0.9996, the phase margin is found to be 8.4 degrees less than the design value of 62 degrees, because of the triple phase break at $T/2$.

Derivation of the Difference Equations: To transform back to the z-plane, we apply the transformation:

$$w = (2/T)(z-1)/(z+1)$$

From Figure 16, the difference equations must provide two filters which, on transformation to the z-plane, become:

$$H_A(z)/K_AK_C = G_A^*(z+1)/(z-k_A)$$

$$H_p(z)/K_pK_C = G_p^*(z-k_V)/(z-k_p)$$

Values for the new terms are as follows, with numerical
default values in parenthesis:

\[ G_A^* = \frac{cT/2K_AK_C}{1+\nu_AT/2} \]

\[ = 0.194/2 \]

\[ k_A = \frac{1-\nu_AT/2}{1+\nu_AT/2} \]

\[ = 0.863 \]

\[ G_P^* = \frac{k\nu_p/k_{\text{max}}\nu_V}{1+\nuVT/2}(1+\nu_pT/2) \]

\[ = 3.39 \]

\[ k_V = \frac{1-\nuVT/2}{1+\nuVT/2} \]

\[ = 0.976 \]

\[ k_P = \frac{1-\nu_PT/2}{1+\nu_PT/2} \]

\[ = 0.676 \]

The difference equations derived from the above are:

\[ u_p(k) = k_p u_p(k-1) - G_P^* (1-k_V)\{x_p(k)+x_p(k-1)\}/2 \]

\[ - G_P^* (1+k_V)\{x_p(k)-x_p(k-1)\}/2 \]

\[ u_A(k) = k_A u_A(k-1) + G_A^*\{x_A(k)+x_A(k-1)\} \]

These can be compared with the equations obtained by the rectangular rule from the analog design:

\[ u_p(k) = (1-\omega_PT)u_p(k-1) - G_PT x_p(k) \]

\[ -G_V\{x_p(k)-x_p(k-1)\} \]
It will be noted that the w-plane design method directly implies use of the trapezoidal rule. It is easier to compare the two approaches if the default values are substituted for the coefficients. For the w-plane design, we get:

\[ u_A(k) = (1-\omega_A T)u_A(k-1) + G_A T x_A(k) \]

\[ u_p(k) = 0.676u_p(k-1) - 0.0814\{x_p(k)+x_p(k-1)\}/2 \]
\[ - 3.35\{x_p(k)-x_p(k-1)\} \]
\[ u_A(k) = 0.863u_A(k-1) + 0.194\{x_A(k)+x_A(k-1)\}/2 \]

which can be compared with the results of the rectangular rule design:

\[ u_p(k) = 0.613u_p(k-1) - 0.0967x_p(k) - 4.0\{x_p(k)-x_p(k-1)\} \]
\[ u_A(k) = 0.853u_A(k-1) + 0.208x_A(k) \]

The worst difference between the coefficients used in the two sets of equations is about 20 percent, so that, evidently, there is no serious loss of performance with the rectangular rule. The difference equations for the w-plane design can be put into more useable form, and the damping can be calculated readily from its w-transform. However, we shall look into another point first.

**System with Minimum Delay**

**Numerical Accuracy:** Consider first, that only the accelerometer circuit is active. Then a +1 input, representing 9.8 m/s² if the channel is calibrated, should result in a force on the proof-mass of \( Mg = 2.72 \) N, or an output from the computer of \( Mg/K_c = 1.42 \).
which is out of the range of the system. As a check on numerical accuracy, let \( u_A(k-1) \) equal 1.42, and let \( x_A(k) \) equal 1.0, then:

\[
u_A(k) = (0.863)(1.42) + (0.194)
= 1.42
= u_A(k-1)
\]

However, the output is quantized to only 256 values, so that the maximum input of +1 is equivalent to \((0.194)(256) = 49\) quantized values. In other words, there are only 49 possible values for the coil force in the static case, and one third of them are out of range. Looking at the synthetic spring from the same approach, an input of -1, representing the proof mass against the structure, should result in an output of 0.25, representing one quarter of \( k_{\text{max}} \). Taking \( u_V(k-1) \) equal to 0.25, and \( x_p(k) \) equal to -1, we get:

\[
u_p(k) = (0.676)(0.25) - (0.0814)(-1)
= 0.25
= u_p(k-1)
\]

In this case, however, the input is equivalent to \((0.0814)(256) = 20\) values, so that the restoring force is limited to 20 quantized values. It is somewhat surprising that the synthetic spring appears to be smooth to the touch, however, it might prove impossible to obtain a very small spring value, equal to a few percent of \( k_{\text{max}} \). This quantization effect would be reduced if \( T \) were increased, but the phase margin might also be reduced at the same time.
Minimum Delay: Figure 20 shows an analog plant driven by a
digital filter in which the time delay is kept to a minimum by
timing the output to occur immediately after the calculations are
completed. The basic period $T_0$ is assumed to be 256 musecs, but
calculations are repeated every $T (=nT_0)$ musecs, while output
occurs at $mT_0$, with $m<n$. We now have the open-loop transfer func-
tion:

$$U(z^n)/X(z^n) = H(z^n)((z^n-1)/z^{n+m})Z_n\{G(s)/s\}$$

The delay and zero-order hold blocks of Figure 17 can be
modified accordingly, so that the overall transfer function
becomes:

$$F(z^n)/A_p(z^n) = \{D_A(z^n)+MD_p(z^n)\}/\{1+D_p(z^n)\}$$

where:

$$D_A(z^n) = H_A/z^m$$

$$D_p(z^n) = H_p(z^n)((z^n-1)/z^{n+m})Z_n\{1/Ms^3\}$$

$$= H_p(z^n)(T^2/2M)(z^n+1)/z^m(z^n-1)^2$$

The $w$-transform is now:

$$z^n = (1+wT/2)/(1-wT/2)$$

with its inverse:

$$w = (2/T)(z^n-1)/(z^n+1)$$

also:

$$\omega = (2/T)\arctan(\nu T/2)$$
FIGURE 20. DIGITAL FILTER WITH ANALOG PLANT: IMMEDIATE OUTPUT
and:

\[ v = (2/T)\tan(\omega T/2) \]

The \( D(w) \) transfer functions now become:

\[
D_A(w) = H_A(w) \left( \frac{1-wT/2}{1+wT/2} \right)^{m/n}
\]

\[
D_p(w) = H_p(w) \left( \frac{1+m/n}{Mw^2(1+wT/2)} \right)^{m/n}
\]

Thus, apart from a change in output timing, and possible redesign for improved phase margin, the difference equations are essentially unchanged when the output is speeded up. However, there should be an improvement in the real damping as \( m/n \) is decreased, which would partially offset the effect of increasing \( n \) to obtain longer cycle times.

**Difference Equations for Minimum Delay Case:** Assuming that the \( H_A(w) \) and \( H_p(w) \) filters are essentially the same as before, we find that on applying the inverse \( w \)-transform we have \( H_A(z^n) \) and \( H_p(z^n) \). However, in obtaining the difference equations from these, we obtain expressions for \( u_V(nkT_0) = u_V(kT) \), etc., so that the final equations are the same as before. The form in which the equations were left is not the most convenient, but note that the first order transfer function:

\[ u(z)/x(z) = a_0(1+z^{-1}a_1)/(1+z^{-1}b_1) \]

can either be written as:

\[ u(k) = -b_1u(k-1) + a_0x(k) + a_0a_1x(k-1) \]

or as the pair of equations:
\[ u_1(k) = -b_1 u_1(k-1) - (a_0/b_1)x(k) \]
\[ u(k) = (a_1-b_1)u_1(k) + (a_0a_1/b_1)x(k) \]

where the additional variable, \( u_1 \) is essentially a state variable. Using this representation, the complete set of equations can be written as:

\[ x_p(k) = x_p(k) \text{ or } x_L(k) \]
\[ u_V(k) = k_F u_V(k-1) - (G^*_p/k_p)x_p(k) \]
\[ u_p(k) = (k_p-k_V)u_V(k) - (G^*_p k_V/k_p)x_p(k) \]
\[ u_B(k) = k_A u_B(k-1) + (G^*_A/k_A)x_A(k) \]
\[ u_A(k) = (1+k_A) u_B(k) - (G^*_A/k_A)x_A(k) \]
\[ u(k) = u_p(k) + u_A(k) + x_S(k) \]

where \( u_V, u_B \) are the corresponding state variables.

**Plots of Real Damping:** The real damping can be calculated as:

\[ H_c = \text{REAL}\{j\omega F(z)/A_F(z)\} \]

This is shown in Figures 21 to 24 for four cases each. One represents the analog approximation obtained by taking \( T=0 \) and is identical to the results shown in Figure 9 for the same parameters, while the remaining three cases are for \( T=4096, 8192, \) and 16,384 microseconds. The other parameters which are varied are the output time delay, which is 0 and 4096 microseconds, \((m=0,16)\), and the design damping, which is 10 and 80 Ns/m. The cases where \( T \) and the time delay are both 4096 microseconds
Real Damping for Proof-Mass Actuator with Digital Controller

Proof-Mass \( m \) = .278kg
Synthetic Stiffness = 38.7 N/m
Design Damping \( c \) = 10 Ns/m
Computation Delay \( mT_0 \) = 0
Real Damping for Proof-Mass Actuator with Digital Controller

Proof Mass (m) = 0.278 kg
Synthetic Stiffness = 38.7 N/m
Design Damping (c) = 10 Ns/m
Computation Delay (mT₀) = 4096 μs
Real Damping for Proof-Mass Actuator with Digital Controller

Proof Mass \( (m) = 0.278 \text{kg} \)
Synthetic Stiffness \( (k) = 38.7 \text{ N/m} \)
Design Damping \( (c) = 80 \text{ Ns/m} \)
Computation Delay \( (mT_o) = 0 \)
Real Damping for Proof-Mass Actuator with Digital Controller

Proof-Mass (m) = 0.278 kg
Synthetic Stiffness (k) = 38.7 N/m
Design Damping (c) = 80 Ns/m
Computation Delay (mTo) = 4096 μs
corresponds to the P- and T-programs listed in Appendix A.

As might be expected, better agreement with the analog approximation is shown when the time delay is 0. Otherwise, agreement is best when T is a minimum. However, at low frequencies, the higher values for T show increased damping, presumably because of greater phase lags. It must be emphasized that two of the timing cases, where the time delay is zero or equal to T, have accurate solutions. The remaining cases introduce additional approximations of uncertain validity.

SUMMARY

Controller Design: The third in a series of controllers for the UVA Proof-Mass Actuator has been designed, built in prototype form, and demonstrated. The present design uses an INTEL 8031 microcontroller mounted in an SDK-51 System Design Kit. Previously, an analog controller had been breadboarded, and a Z80 controller had been developed as a slave to a TRS80 computer. References 7 and 8 are essential for working with the SDK-51, and Reference 9 is of great help.

Digital Control Equations: A procedure for developing digital control equations has been developed, which meets specific requirements:

* A given design damping value.
* Insensitivity to steady acceleration, including gravity.
* A given design centering stiffness.
* A specified phase margin.

Equations based on rectangular integration have been demonstrated. Improved equations, based on w-transform theory, have been developed, which show small changes from the demonstrated values. Finally, real damping vs. frequency has been calculated for both sets of equations, and results of these calculations have been presented in this report.

**Floating-Point Calculations:** The demonstrated equations used floating-point subroutines which were developed for the 8051 series microcontrollers.

**Pulse Width Modulation:** A pulse-width modulator (PWM) was developed for the proof-mass actuator. This draws no current and therefore develops no heat when the actuator is in a quiescent state.

**Word Length:** It is recognized that four factors determine the accuracy of the control program, they are:

* Possible loss of accuracy due to limited word length in input and output.
* Possible loss of significance due to overflow or underflow during internal calculations.
* Digital noise due to inadequate word length.
* Long computational time due to arithmetic complexity.

Experience with the Z80 and the current 8051 series control programs gave no indications of problems due to input or output word length. For example, when programmed as a pure spring, the
proof-mass appears to behave smoothly, without any apparent 'stair-step' feel when operated manually. However, with the 16-bit Z80 system, there were definite indications of internal number overflow. Possibly, these could have been corrected by shifting to the middle 8 bits for input and output. However, the 8051 series is not well adapted to 16-bit arithmetic, and this is why the floating-point approach was tried. Several other schemes could have been used, overall, one might consider any of the following:

* Signed 7-bit arithmetic (8-bit total).
* Signed 15-bit arithmetic (16-bit total).
* Signed 15-bit arithmetic with shift (16-bit total).
* Signed 7-bit mantissa and exponent (16-bit total).
* Signed 11-bit mantissa and signed 3-bit exponent (16-bit total).
* Signed 15-bit mantissa and signed 7-bit exponent (24-bit total).

Since the 8031 chip was used, requiring two ports dedicated to memory access, the SDK-51 system was limited to an 8-bit A/D. Also, but for different reasons, the PWM was limited to 8 effective bits. Since no advantage was seen in going to more bits in either case, the extra hardware which would have been required did not have to be used.

**Future Development:** This report concludes work under the NASA grant, so that any future work will be carried out on internal funds. However, the development of a slave-master system is of
particular interest, because it will make it possible to change the gains on individual controllers, as might be required in operation. Presently available development systems make this a difficult task, because only a single 8051 can be simulated at any one time. Specifically, the proposed development would include the following:

* Installing a slave 8031 in the wire-wrap area of the SDK-51.
* Installing 2K of RAM so that it can be programmed from the SDK-51, but can be used to run programs on the slave.
* Provision for installation of a 2K EPROM in the RAM slot.
* Provision for programming the EPROM in place.
* Interconnection of the serial lines on the two 8031's.
* Use of the four high address bits on the slave 8031 to control A/D and other board functions.

This system would be used to develop slave controller programs on EPROM which would be used in building separate controller boards. The EPROM programming capability would also be used to develop additional library programs for the SDK-51.

CONCLUSIONS AND RECOMMENDATIONS

* The 8051 series microcontrollers are capable of controlling the proof-mass actuator.
* Eight-bit input and output appears adequate, however, with the availability of the additional ports on the 8751, A/D's and D/A's with more bits pose no problem and would require
little extra time.
* Although the floating-point arithmetic gave good results, other arithmetic schemes might require less computing time. The question requires more investigation than was given in the present work.
* The parallel realization design procedure described in this report worked well and appears to be adequate.
* The recommended design procedure requires a fair amount of calculation, especially if the phase margin is to be optimal. For best results, it might be advisable to write a computer program to determine parameters for the difference equations.

REFERENCES


4. Haviland, J.K., "Digital Control System for Space Structural
Dampers," University of Virginia, Department of Mechanical and Aerospace Engineering, Report No. UVA/528224/MAE85/102, July 1984.


APPENDIX A

EXPERIMENTAL PROGRAM FOR SDK-51 BOARD

This program was written to assist in the overall development of the wire-wrapped controller added to INTEL's SDK-51 development board. It is loaded from a cassette tape, titled ABC9, and responds to the keyboard command 'GO FROM 0', by executing a program called DEMO1. While executing this or any other program, it continuously polls the keyboard, and responds to any inputs with ASCII values of 20H to 5FH by a subroutine call to the appropriate location in a table. If it encounters RET, it simply returns to the current program, but, if it encounters JMP addr., it jumps to a new program. The following is a list of keyboard entries which cause jumps to new programs, the number in parenthesis is the address of the program:

C=COIL (0568H): The program waits for two hex characters in 2's complement form, which is output to the coil. Used to measure coil force output.

D=DISPLAY (01A8H): Displays four hex bytes, in 2's complement form, indicating readings of the four analog input ports. Used for calibration of analog inputs.

E=ENTER (0454H): Enters floating-point contents of 06,07 into first stack location, moves two stack contents up, and loses contents of third stack location.

F=FIX (04E0H): Fixed-point equivalent of floating-point
number in 06,07 is stored in 06 (and displayed).

N=NEGATE (0448H): Floating-point contents of 06,07 are negated and replaced in 06,07 (and displayed).

P=P1-D Program (0300H): The P-Program, as described in the test, is run.

Q=continue P1-D Program (0308H): The P-Program is restarted with current parameters (i.e., default values are not read).

R=READ (0470H): First floating-point number on stack is read into 06,07 (and displayed). Remainder of stack is moved down, and third stack location is left unchanged.

T=T version of P1-D Program (0330H): The T-Program, as described in the text, is run.

U=continue T version (0338H): The T-Program is restarted with current parameters (i.e., default values are not read).

X=EXponent (0424H): The program waits for two hex characters, representing the exponent, and enters them into 07 (and displays them). This must follow the mantissa entry, which writes over the current exponent.

Z=NORMALIZE (043CH): The floating-point contents of 06,07 are normalized and replaced in 06,07 (and displayed).

Space Bar, Shift 0,1 (04DOHO: Parameters are entered
from floating point numbers in 06,07, to be followed by U to restart T-Program, according to following table:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space Bar</td>
<td>n</td>
</tr>
<tr>
<td>Shift 1</td>
<td>c</td>
</tr>
<tr>
<td>Shift 2</td>
<td>( \omega_N )</td>
</tr>
</tbody>
</table>

Shift 2 to Shift 9 (04DOH): Parameters are entered from floating-point numbers in 06,07, to be followed by Q to restart P-Program, according to the following table:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift 3</td>
<td>( I_n )</td>
</tr>
<tr>
<td>Shift 4</td>
<td>T</td>
</tr>
<tr>
<td>Shift 5</td>
<td>( \omega_A )</td>
</tr>
<tr>
<td>Shift 6</td>
<td>( \omega_P )</td>
</tr>
<tr>
<td>Shift 7</td>
<td>( G_A )</td>
</tr>
<tr>
<td>Shift 8</td>
<td>( G_P )</td>
</tr>
<tr>
<td>Shift 9</td>
<td>( G_V )</td>
</tr>
</tbody>
</table>

'\*'=MULTIPLY (0490H): Floating-point contents of first stack position are multiplied by contents of 06,07, and replaced in 06,07 (and displayed). Stack contents are moved down, so that both multiplier and multiplicand are lost.

'+'=ADD (04AOH): Floating-point contents of 06,07 are added to contents of first stack position, and replaced in 06,07 (and displayed). Stack contents are moved down, so that both addends are lost.

'\-'=SUBTRACT (04BOH): Floating-point contents of 06,07 are subtracted from contents of first stack position,
and replaced in 06,07 (and displayed). Stack contents are moved down, so that subtractor and subtrahend are lost.

'.'=MANTISSA (0418H): The program waits for two hex characters, and enters them in both 06 and 07. The contents of 06 will represent the mantissa, but the exponent should follow to be placed in 07.

0 to 3 (0186H): DEMO0 to DEMO3 are run, according to the following table:

0 .......... DEMO0 places Channel #0 input at output.
1 .......... DEMO1 places Channel #1 input at output.
2 .......... DEMO2 places Channel #2 input at output.
3 .......... DEMO3 places Channel #3 input at output.

From the point-of-view of proof-mass controller development, the most important items are the two versions of the P1-D controller, referred to as as the P- and T- Programs. These use floating-point subroutines, and make use of the timer interrupt feature of the 8031. A key to internal data memory and a listing of program ABC9 follows.
### Key to Internal Data Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00,01</td>
<td>R0 and R1 pointers</td>
</tr>
<tr>
<td>02</td>
<td>R2 is used for display</td>
</tr>
<tr>
<td>03</td>
<td>R3 cycle counter</td>
</tr>
<tr>
<td>04</td>
<td>R4 exponent</td>
</tr>
<tr>
<td>05</td>
<td>R5 shift counter</td>
</tr>
<tr>
<td>06,07</td>
<td>R6, R7 floating point results</td>
</tr>
<tr>
<td>20</td>
<td>Bit 00 = sign</td>
</tr>
<tr>
<td></td>
<td>Bit 01 = flag</td>
</tr>
<tr>
<td>22</td>
<td>Channel counter</td>
</tr>
<tr>
<td>23</td>
<td>Display counter</td>
</tr>
<tr>
<td>24</td>
<td>Key input</td>
</tr>
<tr>
<td>25</td>
<td>n = # cycles</td>
</tr>
<tr>
<td>26</td>
<td>Present output, 2’s complement hex</td>
</tr>
<tr>
<td>27</td>
<td>Output during next calculation cycle</td>
</tr>
<tr>
<td>2A,2B</td>
<td>n (default value)</td>
</tr>
<tr>
<td>2C,2D</td>
<td>c (default value)</td>
</tr>
<tr>
<td>2E,2F</td>
<td>( \omega_N ) (default value)</td>
</tr>
<tr>
<td>30,31</td>
<td>Calculator stack #1</td>
</tr>
<tr>
<td>32,33</td>
<td>Calculator stack #2</td>
</tr>
<tr>
<td>34,35</td>
<td>Calculator stack #3</td>
</tr>
<tr>
<td>36</td>
<td>( I_n ) (default value)</td>
</tr>
<tr>
<td>38,39</td>
<td>T (default value)</td>
</tr>
<tr>
<td>3A,3B</td>
<td>( \omega_A ) (default value)</td>
</tr>
<tr>
<td>3C,3D</td>
<td>( \omega_P ) (default value)</td>
</tr>
<tr>
<td>3E,3F</td>
<td>( G_A ) (default value)</td>
</tr>
</tbody>
</table>
40,41 & \( G_p \) (default value) \\
42,43 & \( G_v \) (default value) \\
44,45 & \( \omega_A T \) \\
46,47 & \( \omega_P T \) \\
48,49 & \( G_A T \) \\
4A,4B & \( G_P T \) \\
50,51 & \( -x_p \) \\
52,53 & \( x_A \) \\
54,55 & \( -x_L \) \\
56,57 & \( x_S \) \\
58,59 & \( u_V \) \\
5A,5B & \( u_P \) \\
5C,5D & \( u_A \)
APPENDIX A  PROGRAM ABC9

0000=AJMP 0180
0002=NOP
0003=LJMP E003
0006=NOP
0007=NOP
0008=NOP
0009=NOP
000A=NOP
000B=NOP
000C=NOP
000D=NOP
000E=NOP
000F=NOP
0010=NOP
0011=NOP
0012=NOP
0013=NOP
0014=NOP
0015=NOP
0016=NOP
0017=NOP
0018=NOP
0019=NOP
001A=NOP
001B=NOP
001C=NOP
001D=NOP
001E=NOP
001F=NOP
0020=NOP
0021=NOP
0022=NOP
0023=NOP
0024=NOP
0025=NOP
0026=NOP
0027=PUSH DO
0028=PUSH EO
0029=MV A,26
002A=SETB C
002B=RLC A
002C=JC 0032
002D=CPL A
002E=MV 8C,A
002F=MV B5,C
0030=CPL C
0031=MV B4,C
0032=SETB 8E
0033=DJNZ R3,0047
0034=MV R3,25
0035=MV A,27
0036=MV 26,A
0037=SETB 01
0038=SETB 80
0039=FOP E0
003A=FOE D0
003B=RET

RESET - Jump to 0EMO 9

INTERRUPT 1 - Required for SDK-51

TIMER 0 INTERRUPT

Turn TIMER 1 off
Invert R.H. Voltage on Coil
Clear Flag
Call Subroutine
Return from Interrupt

TIMER 1 INTERRUPT

Turn Timer 1 off
Invert R.H. Voltage on coil
Return from Interrupt

TIMER 0 SUBROUTINE

Save PSW
Save A
Output to A
Set Carry
Rotate output left
Skip next instruction if negative
Complement output
Set TIMER 1
RH Voltage high if output negative
Invert sign
L.H. voltage low if output negative
Start TIMER 1
Skip 5 instructions if R3 not zero
Reset
Update
output
Set flag
Set Oscilloscope Signal
Retrieve A
Retrieve RSW
Return
SUBROUTINE to READ A/D

Low digit
Next digit
Enable A/D
Trigger
A/D
Wait
Set Port 3 to read
Set Transceiver to read
Read A/D into A
Disable A/D
Set transceiver to write
Set exponent to zero
Return

SUBROUTINE FOR INITIAL SETUP

Set TCON = 0
Set TMOD. TIMER 0 = Mode 3, TIMER 1 = Mode 2
Set PCON = 0
Set SCON = 0
Set IE. Enable both timer interrupts
Set IP. Timer 1 has priority
Return

SUBROUTINE TO POLL KEYBOARD

Clear carry
Look for keyboard entry
Jump to return on no entry
Read ASC II input
Set bit #7 to zero
Save key input
CALL INTERPRET
Return

SUBROUTINE TO INTERPRET KEYSTROKES

Remove 4 low bits
Test for 20H to 2FH
Jump if successful
Test for 30H to 3FH
Jump if successful
Test for 40H to 4FH
Jump if successful
Test for 50H to 5FH
Get original entry
NOP
Skip two high bits
Multiply by 2
Set DATA POINTER to start of table
Make it a subroutine call
Return from subroutine
Stop TIMER 0
L.H. voltage to zero
R.H. voltage to zero
Jump to table
00DA=LCAL E00F
00DD=MOV R2,*2E
00DF=LCAL E006
00E2=MOV R2,*06
00E4=LCAL E015
00E7=MOV R2,*58
00E9=LCAL E006
00EC=MOV R2,*07
00EE=LCAL E015
00F1=MOV R2,*20
00F3=LCAL E006
00F6=MOV R2,*24
00FB=LCAL E006
00FB=ACAL 0094
00FD=SJMP 00FB

SUBROUTINE TO DISPLAY & WAIT
Clear display
Output
period
Output
mantissa
Output
Cap. X
Output
exponent
Output
space
Output
keystroke
Look for new
keystroke
TABLE = KEYSTROKES 40H to 57H

C = Coil Force
D = Display four analog inputs
E = Enter onto stack
F = Fixed Decimal

N = Negate
P = PI-D Program
Q = Continue P
R = Read stack
T = Alternate PI-D Program
U = Continue T
<table>
<thead>
<tr>
<th>Decimal</th>
<th>Assembly Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0130</td>
<td>AJMP 0424</td>
<td>X = Exponent</td>
</tr>
<tr>
<td>0132</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0133</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0134</td>
<td>AJMP 043C</td>
<td>Z = Normalize</td>
</tr>
<tr>
<td>0136</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0137</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0138</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0139</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013A</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013B</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013C</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013D</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013E</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>013F</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0140</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0141</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0142</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0143</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0144</td>
<td>AJMP 04C0</td>
<td>Shift 0 to 2</td>
</tr>
<tr>
<td>0146</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0147</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0148</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0149</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014A</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014B</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014C</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014D</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014E</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>014F</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0150</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0151</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0152</td>
<td>AJMP 04D0</td>
<td>Shift 3 to 9</td>
</tr>
<tr>
<td>0154</td>
<td>AJMP 0490</td>
<td>'x' = Multiply</td>
</tr>
<tr>
<td>0156</td>
<td>AJMP 04A0</td>
<td>'+' = Add</td>
</tr>
<tr>
<td>0158</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0159</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>015A</td>
<td>AJMP 04B0</td>
<td>'-' = Subtract</td>
</tr>
<tr>
<td>015C</td>
<td>AJMP 0418</td>
<td>'l' = Mantissor</td>
</tr>
<tr>
<td>015E</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>015F</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>0160</td>
<td>AJMP 0186</td>
<td>DEMO 1 - Proximeter test</td>
</tr>
<tr>
<td>0162</td>
<td>AJMP 0186</td>
<td>DEMO 2 - Accelerometer test</td>
</tr>
<tr>
<td>0164</td>
<td>AJMP 0186</td>
<td>DEMO 3 - LVDT test</td>
</tr>
<tr>
<td>0166</td>
<td>AJMP 0186</td>
<td>DEMO 4 - Signal generator test</td>
</tr>
<tr>
<td>0168</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>017F</td>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>
DEMO 0-3 PROGRAMS

0180=LCAL E00C
0183=MOV 24 r #30
0186=MOV 81 r #60
0189=MOV 25 r #01
018C=ACAL 007B
018E=SETB 8C
0190=CLR 01
0192=MOV A r 24
0194=MOV 22 r A
0196=ACAL 0054
0198=MOV 27 r A
019A=CLR B0
019C=ACAL 0094
019E=JB 01 r 0190
01A1=SJMP 019E

01A8=MOV 81 r #60
01AB=NOP
01AC=NOP
01AD=NOP
01AE=ACAL 007B
01B0=MOV A8 r #E1
01B3=MOV B8 r #00
01B6=SETB 8C
01B8=ACAL 0100
01BA=ACAL 0094
01BC=ACAL 007B
01C0=JNB BD r 01E1
01C3=CLR BD
01C5=INC 23
01C7=MOV A r 23
01C9=JNZ 01E1
01CB=LCAL E00F
01CE=MOV R0 r #22
01D0=MOV @R0 r #00
01D2=ACAL 0054
01D4=MOV R2 r A
01D5=LCAL E015
01D8=MOV R2 r #2C
01DA=LCAL E006
01DD=INC @R0
01DE=CJNE @R0 r #04 r 01D2
01E1=RET

DISPLAY 4 INPUTS

0180=LCAL E00C
0183=MOV 24 r #30
0186=MOV 81 r #60
0189=MOV 25 r #01
018C=ACAL 007B
018E=SETB 8C
0190=CLR 01
0192=MOV A r 24
0194=MOV 22 r A
0196=ACAL 0054
0198=MOV 27 r A
019A=CLR B0
019C=ACAL 0094
019E=JB 01 r 0190
01A1=SJMP 019E

01A8=MOV 81 r #60
01AB=NOP
01AC=NOP
01AD=NOP
01AE=ACAL 007B
01B0=MOV A8 r #E1
01B3=MOV B8 r #00
01B6=SETB 8C
01B8=ACAL 0100
01BA=ACAL 0094
01BC=ACAL 007B
01C0=JNB BD r 01E1
01C3=CLR BD
01C5=INC 23
01C7=MOV A r 23
01C9=JNZ 01E1
01CB=LCAL E00F
01CE=MOV R0 r #22
01D0=MOV @R0 r #00
01D2=ACAL 0054
01D4=MOV R2 r A
01D5=LCAL E015
01D8=MOV R2 r #2C
01DA=LCAL E006
01DD=INC @R0
01DE=CJNE @R0 r #04 r 01D2
01E1=RET

SUBROUTINE FOR INPUT

0180=LCAL E00C
0183=MOV 24 r #30
0186=MOV 81 r #60
0189=MOV 25 r #01
018C=ACAL 007B
018E=SETB 8C
0190=CLR 01
0192=MOV A r 24
0194=MOV 22 r A
0196=ACAL 0054
0198=MOV 27 r A
019A=CLR B0
019C=ACAL 0094
019E=JB 01 r 0190
01A1=SJMP 019E

01A8=MOV 81 r #60
01AB=NOP
01AC=NOP
01AD=NOP
01AE=ACAL 007B
01B0=MOV A8 r #E1
01B3=MOV B8 r #00
01B6=SETB 8C
01B8=ACAL 0100
01BA=ACAL 0094
01BC=ACAL 007B
01C0=JNB BD r 01E1
01C3=CLR BD
01C5=INC 23
01C7=MOV A r 23
01C9=JNZ 01E1
01CB=LCAL E00F
01CE=MOV R0 r #22
01D0=MOV @R0 r #00
01D2=ACAL 0054
01D4=MOV R2 r A
01D5=LCAL E015
01D8=MOV R2 r #2C
01DA=LCAL E006
01DD=INC @R0
01DE=CJNE @R0 r #04 r 01D2
01E1=RET

0180=LCAL E00C
0183=MOV 24 r #30
0186=MOV 81 r #60
0189=MOV 25 r #01
018C=ACAL 007B
018E=SETB 8C
0190=CLR 01
0192=MOV A r 24
0194=MOV 22 r A
0196=ACAL 0054
0198=MOV 27 r A
019A=CLR B0
019C=ACAL 0094
019E=JB 01 r 0190
01A1=SJMP 019E

01A8=MOV 81 r #60
01AB=NOP
01AC=NOP
01AD=NOP
01AE=ACAL 007B
01B0=MOV A8 r #E1
01B3=MOV B8 r #00
01B6=SETB 8C
01B8=ACAL 0100
01BA=ACAL 0094
01BC=ACAL 007B
01C0=JNB BD r 01E1
01C3=CLR BD
01C5=INC 23
01C7=MOV A r 23
01C9=JNZ 01E1
01CB=LCAL E00F
01CE=MOV R0 r #22
01D0=MOV @R0 r #00
01D2=ACAL 0054
01D4=MOV R2 r A
01D5=LCAL E015
01D8=MOV R2 r #2C
01DA=LCAL E006
01DD=INC @R0
01DE=CJNE @R0 r #04 r 01D2
01E1=RET

DEMO 0-3 PROGRAMS

Read keyboard
Set input channel to #0
Stack pointer = 60
Counter input = 0
CALL INITIAL SETUP
Start TIMER 0
Clear flag
Keystroke (0 to 3)
into Channel #
Call READ A/D
A/D input to 27
Clear oscilloscope signal
CALL POLL KEYBOARD
Jump if flag high
Wait for interrupt

DISPLAY 4 INPUTS

Stack pointer = 60

SUBROUTINE FOR INPUT

Return if TIMER 0 flag low
Clear TIMER 0 overflow flag
Increment display counter
Return on
nonzero display counter
Clear display
Set channel #0
to zero
CALL READ A/D
Display
reading
Output
comma
Increment channel #
Continue if channel # not 5
Return
SUBROUTINE TO CORRECT MANTISSA OVERFLOW
Return on no OVERFLOW
Rotate right
Jump if exponent not maximum
Return
Increment exponent
Return

SUBROUTINE TO NEGATE A, 04
Complement A
Add Unity
Correct overflow
Return

SUBROUTINE TO NORMALIZE MANTISSA
Jump if negative
Return if normalized
Set carry if positive
to enter 1's
Return if normalized
Clear carry if negative to enter 0's
Rotate left through carry
Jump if exponent not minimum
Return
Decrement exponent
Continue

SUBROUTINE TO SHIFT MANTISSA TO RIGHT
Increment shift counter
Decrement shift counter, jump if nonzero
Return
Set carry = sign bit
Rotate, right
Continue

SUBROUTINE MOVE A; 04 TO 06, 07
A to 06
04 to 07
Return
SUBROUTINE @RO + @R1 + 06, 07
Increment RO to exponent address
Increment RI to exponent address
Exponent = #0
Clear carry
Subtract exponent #0
Complement carry
Skip next instruction if no overflow
Rotate right
Skip eight instructions if negative
Set exponent difference in shift counter
Store exponent #0 in 04
Decrement RI to Mantissa address
Mantissa #0 to A
CALL SHIFT MANTISSA
Decrement RO to Mantissa address
Add Mantissa #0
Jump to exit
Complement to get exponent difference
Add 1 to get 2's complement
Set exponent difference in shift counter
Store exponent #1 in 04
Decrement RO to Mantissa address
Mantissa #0 to A
CALL SHIFT MANTISSA
Decrement RI to Mantissa address
Add Mantissa #1
(Exit). CALL CORRECT MANTISSA
CALL NORMALIZE MANTISSA
CALL MOVE A, 04 to 06, 07
Return
Mantissa #1 to A
Increment RI to exponent address
Exponent #1 to 04
Decrement RI to Mantissa address
CALL NEGATE A, 04
CALL NORMALIZE MANTISSA
Store Mantissa
Increment RI to exponent address
Store exponent
Restore RI
Return
Increment RI to exponent address
CALL NEGATE A, 04
CALL NORMALIZE MANTISSA
Store Mantissa
Increment RI to exponent address
Store exponent
Restore RI
Return
Increment RI to exponent address
CALL NEGATE A, 04
CALL NORMALIZE MANTISSA
Store Mantissa
Increment RI to exponent address
Store exponent
Restore RI
Return
Increment RI to exponent address
CALL NEGATE A, 04
CALL NORMALIZE MANTISSA
Store Mantissa
Increment RI to exponent address
Store exponent
Restore RI
Return
Increment RI to exponent address
CALL NEGATE A, 04
CALL NORMALIZE MANTISSA
Store Mantissa
Increment RI to exponent address
Store exponent
Restore RI
Return
SUBROUTINE FLOAT @RO to @R1

Clear A
Zero to 04
Mantissa #0 to A
CALL NORMALIZE MANTISSA (ENTRY)*
A to Mantissa #1
Increment R1 to exponent
04 to exponent #1
Restore R1
Return

*ENTRY FOR FLOAT A, 04 to @R1

SUBROUTINE FIX @RO to @R1

Increment R0 to exponent
Exponent #0 to A
Decrement R0 to Mantissa
Skip 5 instructions if negative
Skip 6 instructions if zero
Mantissa #0 to A
Keep sign of Mantissa
CALL NORMALIZE MANTISSA
Jump to exit
Complement negative exponent
2's complement
Set shift counter
Mantissa #0 to A
CALL SHIFT MANTISSA
(Exit) A to Mantissa #1
Return

SUBROUTINE STORE '06, 07 in @R1
06 to Mantissa #1
Increment R1 to exponent
07 to exponent #1
Restore R1
Return
0300=MOV B1, #60
0303=MOV DPTR, #0AFO
0306=ACAL 0356
0308=ACAL 0362
030A=ACAL 0380
030C=ACAL 03A0
030E=ACAL 0318
0310=MOV R0, #56
0312=MOV R1, #5A
0314=ACAL 0240
0316=MOV R0, #06
0318=MOV R1, #5C
031A=ACAL 0240
031C=MOV R0, #06
031E=MOV R1, #27
0320=ACAL 02DC
0322=CLR B0
0324=ACAL 0094
0326=JB 01, 030A
0329=JMP 0326

P-PROGRAM
Set stack pointer to 60
Set data pointer to TABLE 1
CALL READ P PARAMETERS
CALL MULTIPLY; BY T
CALL READ INPUTS
CALL CALCULATE Ua
CALL CALCULATE Up
SET R0 to Xs
SET R1 to Up
CALL @RO + @R1 to 06, 07
SET R0 to 06
SET R1 to Ua
CALL @RO + @R1 to 06, 07
SET R0 to 06
SET R1 to OUTPUT
CALL FIX @RO to @R1
CLEAR oscilloscope signal
CALL POLL KEYBOARD
LOOP if flag high
Wait for interrupt

0330=MOV B1, #60
0333=MOV DPTR, #05F8
0336=ACAL 0340
0338=ACAL 0500
033A=AJMP 0308

T-PROGRAM
Set stack pointer to 60
Set data pointer to TABLE 2
CALL READ T PARAMETERS
CALL CALCULATE PARAMETERS
Jump to P-Program

0340=ACAL 007B
0342=MOV R1, #2A
0344=MOVX A, @DPTR
0345=MOV @R1, A
0346=INC DPTR
0347=INC R1
0348=CJNE R1, #30, 0344
034B=RET

SUBROUTINE READ T PARAMETERS
CALL INITIAL SETUP
Set R1 to n
TABLE 2 to A
Store A
Increment data pointer
Increment R1
LOOP until R1 = 30
Return

0356=ACAL 007B
0358=MOV R1, #36
035A=MOVX A, @DPTR
035B=MOV @R1, A
035C=INC DPTR
035D=INC R1
035E=CJNE R1, #44, 035A
0361=RET

SUBROUTINE READ P PARAMETERS
CALL INITIAL SETUP
Set R1 to In
TABLE 1 to A
Store A
Increment data pointer
Increment R1
LOOP until R1 = 44
Return
SUBROUTINE SHIFT @ R0 + @R1

0288=MOV A,@R0
Mantissa #0 to A
0289=MOV @R1,A
A to Mantissa #1
028A=INC R0
Increment R0 to exponent address
028B=INC R1
Increment R1 to exponent address
028C=MOV A,@R0
Exponent #0 to A
028D=MOV @R1,A
A to exponent #1
028E=DEC R0
Restore R0
028F=DEC R1
Restore R1
0290=RET
Return

SUBROUTINE @R0 * @R1 + 06, 07

0294=CLR 00
Clear sign flag
0296=INC R0
Increment R0 to exponent address
0297=INC R1
Increment R1 to exponent address
0298=MOV A,@R0
Exponent #0 to A
0299=ADD A,@R1
Add exponent #1
029A=JNB D2,02A5
Skip 4 instructions if no overflow
029D=JNC 02A3
Skip 2 instructions if positive
029F=MOV A,#80
Set exponent to 80H
02A1=JMP 02A5
Skip next instruction
02A3=MOV A,#7F
Set exponent to 7FH
02A5=MOV R4,A
Exponent to 04
02A6=DEC R0
Decrement R0 to Mantissa address
02A7=DEC R1
Decrement R1 to Mantissa address
02A9=MOV A,@R0
Mantissa #0 to A
02A9=JNB E7,02B0
Skip two instructions if positive
02AC=CPL 00
Complement sign bit
02AE=ACAL 0204
CALL NEGATE A, 04
02B0=NOP
NOP
02B1=MOV F0,A
Mantissa #0 to B
02B3=MOV A,@R1
Mantissa #1 to A
02B4=JNB E7,02BB
Skip two instructions if positive
02B7=CPL 00
Complement sign bit
02BA=ACAL 0204
CALL NEGATE A, 04
02BB=CLR C
Clear carry
02BC=RLC A
Rotate left
02BD=MUL AB
Multiply A * B
02BE=MOV A,F0
Product to A
02C0=NOP
NOP's
02C1=NOP
to
02C2=NOP
be
02C3=NOP
removed
02C4=NOP
-
02C5=NOP
-
02C6=JNB 00,02CB
Skip two instructions if sign positive
02C9=ACAL 0204
NEGATE A, 04
02CB=ACAL 0210
CALL NORMALIZE MANTISSA
02CD=ACAL 0234
CALL MOVE A, 04 to 06, 07
02CF=RET
Return
SUBROUTINE MULTIPLY BY T

Set RO to wa
Set R1 to wa
Save R1
Set R1 to T
CALL @RO * @R1 to 06, 07
Retrieve R1
CALL STORE 06, 07 in @R1
Increment RO twice
Increment R1 twice
Loop until R1 = 4C
STORE In
in 25
Start TIMER 0
Return

SUBROUTINE READ INPUTS

Clear flag
Set channel # to 0
Set R1 to -Xp
CALL READ A/D
CALL FLOAT A, 04 to @R1
Increment channel #
Increment R1 twice
Loop until R1 = 58
Set R0 to -Xp
Set R1 to -X1
CALL @R0 + @R1 to 06, 07
CALL STORE 06, 07 in @R1
Return

SUBROUTINE CALCULATE Uv

Set RO to wpt
Set R1 to Up
wpTup to 06, 07
wpTup - Uv to 06, 07
wpTup - Uv to Uv
wpTup - Uv to Uv
R0 set to Gpt
R1 set to Xp
-XpGpT to 06, 07
-XpGpT to 06, 07
-XpGpT to 06, 07
Updated Uv to 06, 07
Updated Uv to Uv
Updated Uv to Uv
Updated Uv to Uv
Set RO to Gv
Set R1 to -Xp
-XpGv to 06, 07
-XpGv to 06, 07
-XpGv to 06, 07
Uv - XpGv to 06, 07
Uv - XpGv to 06, 07
Uv - XpGv to 06, 07
Set R1 to Up
Set R1 to Up
Set R1 to Up
Set R1 to Up
Set R1 to Up
Return
SUBROUTINE CALCULATE \( U_a \)

- Set \( R_0 \) to \( \omega T \)
- Set \( R_1 \) to \( U_a \)
- Set \( R_0 \) to \( 06, 07 \)
- Set \( R_1 \) to \( U_a \)
- \( \omega Ua - Ua \) to \( 06, 07 \)
- \( \omega Ua - Ua \) to \( V_a \)
- Set \( R_0 \) to \( G_T \)
- Set \( R_1 \) to \( X_a \)
- \( G_T X_a \) to \( 06, 07 \)
- \( G_T X_a \) to \( U_a \)
- Updated \( U_a \) to \( 06, 07 \)
- Updated \( U_a \) to \( U_a \)
- Return

SUBROUTINE READ HEX BYTE

- CALL READ KEY
- Store in \( R_2 \)
- CALL CONVERT TO HEX
- Place in top 4 bits
- Store in \( R_7 \)
- CALL READ KEY
- Store in \( R_2 \)
- CALL CONVERT TO HEX
- Add to \( R_7 \)
- Store in \( R_7 \)
- Return

PROGRAM 'I' TO READ MANTISSA

- Set stack pointer to 60
- CALL READ HEX BYTE
- Store in \( 06 \)
- Jump to DISPLAY AND WAIT

PROGRAM 'X' TO READ EXPONENT

- Set stack pointer to 60
- CALL READ HEX BYTE
- Jump to DISPLAY AND WAIT

SUBROUTINE TO NORMALIZE

- \( 06 \) to \( A \)
- \( 07 \) to \( 04 \)
- CALL TO NORMALIZE \( A, 04 \)
- \( A \) to \( 06 \)
- \( 04 \) to \( 07 \)
- Return

SUBROUTINE TO NORMALIZE DISPLAY

- Set stack pointer to 60
- CALL NORMALIZE
- Jump to DISPLAY AND WAIT
I 0448=MOV 81,#60
I 044B=MOV R1,#06
I 044D=ACAL 026C
I 044F=AJMP 000A

PROGRAM 'N' to NEGATE DISPLAY
Set stack pointer to 60
Set R1 to 06
CALL NEGATE @R1
Jump to DISPLAY and WAIT

I 0454=MOV 81,#60
I 0457=MOV R1,#35
I 0459=MOV R0,#33
I 045B=MOV A,@R0
I 045C=MOV @R1,A
I 045D=DEC R0
I 045E=DEC R1
I 045F= CJNE R0,#2F,045B
I 0462=MOV @R1,07
I 0464=DEC R1
I 0465=MOV @R1,06
I 0467=AJMP 000A

PROGRAM 'E' to ENTER STACK
Set stack pointer to 60
Set R1 to STACK 3
Set R2 to STACK 2
Old stack to A
A to new stack
Decrement R0
Decrement R1
Loop until R0 = 2FH
Decrement R1
07 to stack. 1 exponent
06 to Stack 1 'Mantissa
Jume to DISPLAY and WAIT

I 0470=MOV 81,#60
I 0473=MOV R0,#30
I 0475=MOV R1,#06
I 0477=ACAL 0288
I 0479=ACAL 0480
I 047B=AJMP 000A

PROGRAM 'R' to READ STACK
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 to @R1
CALL SHIFT STACK
Jump to DISPLAY and WAIT

I 0480=MOV R0,#32
I 0482=MOV R1,#30
I 0484=MOV A,@R0
I 0485=MOV @R1,A
I 0486=INC R0
I 0487=INC R1
I 0488= CJNE R0,#36,0484
I 048B=RET

SUBROUTINE TO SHIFT STACK
Set R0 to STACK 2
Set R1 to STACK 1
Old stack to A
A to new stack
Increment R0
Increment R1
Loop until R0 = 36
Return

I 0490=MOV 81,#60
I 0493=MOV R0,#30
I 0495=MOV R1,#06
I 0497=ACAL 0274
I 0499=ACAL 0480
I 049B=AJMP 000A

PROGRAM 'M' to MULTIPLY
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 * @R1 to 06, 07
CALL SHIFT STACK
Jump to DISPLAY and WAIT

"PROGRAM IN" to NEGATE DISPLAY
Set stack pointer to 60
Set R1 to 06
CALL NEGATE @R1
Jump to DISPLAY and WAIT

"PROGRAM 'E' to ENTER STACK
Set stack pointer to 60
Set R1 to STACK 3
Set R2 to STACK 2
Old stack to A
A to new stack
Decrement R0
Decrement R1
Loop until R0 = 2FH
Decrement R1
07 to stack. 1 exponent
06 to Stack 1 'Mantissa
Jume to DISPLAY and WAIT

"PROGRAM 'R' to READ STACK
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 to @R1
CALL SHIFT STACK
Jump to DISPLAY and WAIT

"SUBROUTINE TO SHIFT STACK
Set R0 to STACK 2
Set R1 to STACK 1
Old stack to A
A to new stack
Increment R0
Increment R1
Loop until R0 = 36
Return

"PROGRAM 'M' to MULTIPLY
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 * @R1 to 06, 07
CALL SHIFT STACK
Jump to DISPLAY and WAIT

"PROGRAM IN" to NEGATE DISPLAY
Set stack pointer to 60
Set R1 to 06
CALL NEGATE @R1
Jump to DISPLAY and WAIT

"PROGRAM 'E' to ENTER STACK
Set stack pointer to 60
Set R1 to STACK 3
Set R2 to STACK 2
Old stack to A
A to new stack
Decrement R0
Decrement R1
Loop until R0 = 2FH
Decrement R1
07 to stack. 1 exponent
06 to Stack 1 'Mantissa
Jume to DISPLAY and WAIT

"PROGRAM 'R' to READ STACK
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 to @R1
CALL SHIFT STACK
Jump to DISPLAY and WAIT

"SUBROUTINE TO SHIFT STACK
Set R0 to STACK 2
Set R1 to STACK 1
Old stack to A
A to new stack
Increment R0
Increment R1
Loop until R0 = 36
Return

"PROGRAM 'M' to MULTIPLY
Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @R0 * @R1 to 06, 07
CALL SHIFT STACK
Jump to DISPLAY and WAIT
PROGRAM ' + ' to ADD

04A0 = MOV B1, $60
04A3 = MOV RO, $30
04A5 = MOV R1, $06
04A7 = ACAL 0240
04A9 = ACAL 0480
04AB = AJMP 00DA

Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @RO + @R1 to 06, 07
CALL SHIFT STACK
Jump to DISPLAY and WAIT

PROGRAM ' - ' to SUBTRACT

04B0 = MOV B1, $60
04B3 = MOV RO, $30
04B5 = MOV R1, $06
04B7 = ACAL 02CC
04B9 = ACAL 0480
04BB = AJMP 00DA

Set stack pointer to 60
Set R0 to STACK 1
Set R1 to 06
CALL @RO - @R1 to 06, 07
CALL SHIFT STACK
Jump to DISPLAY and WAIT

PROGRAM S.B., SHIFTS 1, 1. INPUTS

04C0 = MOV B1, $60
04C3 = MOV A, $24
04C5 = CLR C
04C6 = RLC A
04C7 = ADD A, $EA
04C9 = MOV R1, A
04CA = ACAL 02F4
04CC = AJMP 00DA

Set stack pointer to 60
Key input to A
Clear carry
Rotate A left
Subtract 6
Result to R1
CALL 06, 07 to @R1
Jump to DISPLAY and WAIT

PROGRAM SHIFTS 1-9. INPUTS

04D0 = MOV B1, $60
04D3 = MOV A, $24
04D5 = CLR C
04D6 = RLC A
04D7 = ADD A, $F0
04D9 = MOV R1, A
04DA = ACAL 02F4
04DC = AJMP 00DA

Set stack pointer to 60
Key input to A
Clear carry
Rotate A left
Subtract 16
Result to R1
CALL 06, 07 to @R1
Jump to DISPLAY and WAIT

PROGRAM ' F ' to FIX DISPLAY

04E0 = MOV B1, $60
04E3 = MOV RO, $06
04E5 = MOV R1, $06
04E7 = ACAL 02DC
04E9 = AJMP 00DA

Set stack pointer to 60
Set R0 to 06
Set R1 to 06
CALL FIX @RO to @R1
Jump to DISPLAY and WAIT
### TABLE 1. PARAMETERS FOR PROGRAM P

<table>
<thead>
<tr>
<th>In</th>
<th>T</th>
<th>wa</th>
<th>wp</th>
<th>Ga</th>
<th>Gp</th>
<th>Gv</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### SUBROUTINE - CALCULATE T PARAMETERS

```assembly
0500=MOV DPTR,#05EB
0503=MOV R0,#06
0505=ACAL 0558
0507=MOV R1,#2A
0509=ACAL 0294
050B=MOV R1,#36
050D=ACAL 02DC
050F=ACAL 0558
0511=MOV R1,#2A
0513=ACAL 0294
0515=MOV R1,#38
0517=ACAL 02F4
0519=ACAL 0558
051B=MOV R1,#2C
051D=ACAL 0294
051F=MOV R1,#3A
0521=ACAL 02F4
0523=ACAL 0558
0525=MOV R1,#2E
0527=ACAL 0294
0529=MOV R1,#3C
052B=ACAL 02F4
052D=ACAL 0558
052F=MOV R1,#2C
0531=ACAL 0294
0533=MOV R1,#3E
0535=ACAL 02F4
0537=ACAL 0558
0539=MOV R1,#2E
053B=ACAL 0294
053D=MOV R1,#2E
053F=ACAL 0294
0541=MOV R1,#42
0543=ACAL 02F4
0545=ACAL 0558
0547=ACAL 0294
0549=MOV R1,#2E
054B=ACAL 0294
054D=MOV R1,#40
054F=ACAL 02F4
0551=RET
```

SET data pointer to TABLE 2
SET R0 to 06
CALL GET DATA
SET R1 to n
CALL @RO + @R1 to 06, 07
SET R1 to In
CALL FIX @RO to @R1
CALL GET DATA
SET R1 to w
0525 = MOV R1, #2E
CALL GET DATA
Set R1 to wn
CALL @RO * @R1 to 06, 07
0529 = MOV R1, #3C
CALL @RO to @R1
Set R1 to wp
052B = ACAL 02F4
CALL 06, 07 to @R1
052D = ACAL 0558
CALL GET DATA
Set R1 to C
0531 = ACAL 0294
CALL @RO * @R1 to 06, 07
0533 = MOV R1, #3E
SET R1 to Ga
0535 = ACAL 02F4
CALL 06, 07 to @R1
0537 = ACAL 0558
CALL GET DATA
Set R1 to wn
0539 = MOV R1, #2E
CALL @RO * @R1 to 06, 07
053B = ACAL 0294
Set R1 to wn
053D = MOV R1, #2E
CALL @RO * @R1 to 06, 07
053F = ACAL 0294
Set R1 to Gv
0541 = MOV R1, #42
CALL 06, 07 to @R1
0543 = ACAL 02F4
CALL GET DATA
0545 = ACAL 0558
CALL @RO * @R1 to 06, 07
0547 = ACAL 0294
Set R1 to wn
0549 = MOV R1, #2E
CALL @RO * @R1 to 06, 07
054B = ACAL 0294
Set R1 to Gp
054D = MOV R1, #40
CALL 06, 07 to @R1
054F = ACAL 02F4
Return
0558=MOVX A, @DPT
0559=MOV 06, A
055B=INC DPTR
055C=MOVX A, @DPT
055D=MOV 07, A
055F=INC DPTR
0560=RET

SUBROUTINE TO GET DATA
Move from TABLE to A
A to 06 (Mantissa)
Increment data pointer
Move from TABLE to A
A to 07 (exponent)
Increment data pointer
Return

0568=MOV 81, $60
056B=MOV 25, $01
056E=ACAL 007B
0570=ACAL 0400
0572=SETB 8C
0574=CLR 01
0576=MOV A, 07
0578=MOV 27, A
057A=CLR B0
057C=ACAL 0094
057E=JB 01, 0574
0581=SJMP 057E

PROGRAM 'C' FOR COIL FORCE
SET stack pointer to 60
SET 1 cycle
CALL SETUP
CALL READ HEX BYTE
Start TIMER 0
Clear flag
07 (HEX BYTE) to A
A to output
Clear oscilloscope signal
CALL POLL KEYBOARD
Loop if flag high
Wait for interrupt

TABLE 2 FOR T-PROGRAM
2-7, 256x10^-6
3.60, 8
5.08, .0287
.5, —
.05, c
.04, 00, 00
.04, 00, 00
Six logic diagrams for the digital controller follow. Each has a sheet number, used when referring to connections between different sheets. Power and ground connections to standard DIPs are not shown, nor are despiking capacitors. The first five sheets refer to circuits on the wire-wrap area of the SDK-51 board, while sheet 6 refers to the PWM board.

Sheet 1, Figure B1: This shows the transceiver which was used to permit sharing of some pins between input and output functions. Also, the drivers for the PWM board are shown. These were needed because the signals from the 8031 where inadequate to drive the LED’s in the opto-transistors.

Sheet 2, Figure B2: This shows the A/D converter, which is made up from a DAC0800 8-bit D/A converter, and a DM2502 successive approximation register. A high speed LM361 comparator is used to produce a TTL signal to the DM2502, which requires ten cycles to convergence. Presently, a 3 MHz clock is being used, so that convergence time is 3.33 microseconds. It is only achieving about 7-bit accuracy, but it is hoped that this will be improved with further adjustment.

Sheet 3, Figure B3: This shows the clock used to drive the A/D converter. It is derived from the 12 MHz crystal on the SDK-51 board, and provides four options, ranging from 6 MHz to 0.75 MHz, selectable with a jumper. Also, there is a one-and-one-only circuit to synchronize the start of the A/D with the clock.

Sheet 4, Figure B4: This shows the analog switch and demultiplexer circuit used to select the analog channel which is to be read by the A/D converter. A high speed operational amplifier is necessary to provide adequate output impedance combined with the switching speed required.

Sheet 5, Figure B5: This shows a typical analog amplifier circuit, of which two are presently populated on the wire-wrap area. The circuit is provided with three jumpers to provide flexibility in selecting gain ranges and input offsets, such as are experienced with the proximeter. Diode protection prevents accidental damage to the circuits, should the input voltage become excessive.

Sheet 6, Figure B6: This shows the pulse-width-modulation (PWM) board, which is mounted on the proof-mass actuator. Each end of the coil can be switched independently, so that, with suitable digital program logic, high currents can be avoided with the actuator in a quiescent state. Although there is a 15V supply, the circuit only provides an 8V differential, which provides about one Ampere of current in either direction. Planned improvements
FIGURE B2
Sheet 2: A/D Converter
FIGURE B3.
Sheet 3: A/D Trigger and Clock

[Diagram of A/D Trigger and Clock]
Sheet 5: Analog Input Port (Typical)
Figure B6
Sheet 6: PWM Board

[Diagram of PWM Board with various components like resistors, capacitors, and transistors labeled with values such as 2.2k, 450Ω, 68Ω, 1k, 8.5Ω, NTE 261, NTE 262, and 4N28.]

Connector to Sheet 1
in this circuit should make at least one and a half Amperes possible. Opto-transistors permit electrical isolation of this board, with its own power supply, from the SDK-51 board.