NASA/TM—2001-210701

A Comprehensive C++ Controller for a Magnetically Supported Vertical Rotor: Version 1.0

Carlos R. Morrison
Glenn Research Center, Cleveland, Ohio

April 2001

NASA/TM—2001-210701

A Comprehensive C++ Controller for a Magnetically Supported Vertical Rotor: Version 1.0

Carlos R. Morrison
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

April 2001

Trade names or manufacturers' names are used in this report for identification only. This usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

CONTENTS

A COMPREHENSIVE C++ CONTROLLER FOR A MAGNETICALLY SUPPORTED VERTICAL ROTOR: VERSION 1.0

Carlos R. Morrison
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

SUMMARY

This manual describes the new FATMaCC (Five-Axis, Three-Magnetic-Bearing Control Code). The FATMaCC (pronounced "fat mak") is a versatile control code that possesses many desirable features that were not available in previous in-house controllers. The ultimate goal in designing this code was to achieve full rotor levitation and control at a loop time of 50 µs. Using a 1-GHz processor, the code will control a five-axis system in either a decentralized or a more elegant centralized (modal control) mode at a loop time of 56 µs. In addition, it will levitate and control (with only minor modification to the input/output wiring) a two-axis and/or a four-axis system. Stable rotor levitation and control of any of the systems mentioned above are accomplished through appropriate key presses to modify parameters, such as stiffness, damping, and bias. A signal generation block provides 11 excitation signals. An excitation signal is then superimposed on the radial bearing *x*- and *y*-control signals, thus producing a resultant force vector. By modulating the signals on the bearing *x*- and *y*-axes with a cosine and a sine function, respectively, a radial excitation force vector is made to rotate 360° about the bearing geometric center. The rotation of the force vector is achieved manually by using key press or automatically by engaging the "one-per-revolution" feature. Rotor rigid body modes can be excited by using the excitation module. Depending on the polarities of the excitation signal in each radial bearing, the bounce or tilt mode will be excited.

1.0 INTRODUCTION

For the past 14 years, the NASA Glenn Research Center has been actively involved in the development of magnetic bearings. Most of these dynamic suspension systems support a rotor in a two-axis or four-axis configuration. One of these two-axis systems, the Dynamic Spin Rig (DSR), supports a vertical rotor by employing a ball bearing at the upper end and a radial magnetic bearing at the lower end. The DSR is used primarily for vibration testing of turbomachinery blades and components under a spinning condition in a vacuum. The ball bearing imposes limitations, such as frictional heating, on the rotational speeds (less than 18 000 rpm) of the rotor.

By the late 1990's, the previous technologies had set the stage for the development of the Five-Axis, Three-Magnetic-Bearing Dynamic Spin Rig. The motivation for developing this type of bearing system was to achieve higher rotational speeds (25 000 to 60 000 rpm) in the spin rig for use in high-cycle-fatigue research projects pertaining to damping and mistuning for bladed disks.

The Five-Axis, Three-Magnetic-Bearing Dynamic Spin Rig consists of three magnetic bearings: a thrust bearing, a radial upper bearing, and a radial lower bearing. Figure 1 shows the actual shaft or rotor; figure 2, the rotor being held for size comparison; figure 3, the top portion of the rotor where the thrust bearing is affixed; figure 4, the thrust plate and the thrust coils; and figure 5, the upper and lower radial stators.

A control code written in C++ was designed for this magnetic bearing configuration. A 100-MHz processor PC, capable of running the code at a sampled average loop time of 100 µs, can simultaneously control all three magnetic bearings in a centralized (modal control) or decentralized mode. When the code's executable file is launched and all the input parameters are correctly set, the bearings will levitate a vertical, solid, cylindrical shaft. The energized bearings are capable of lifting and shaking a rotor and test article that have a combined weight of 400 lb.

The 23 sections of this manual and appendix A will help the user to correctly set up and run the code. Appendix B lists the source code cited in the manual.

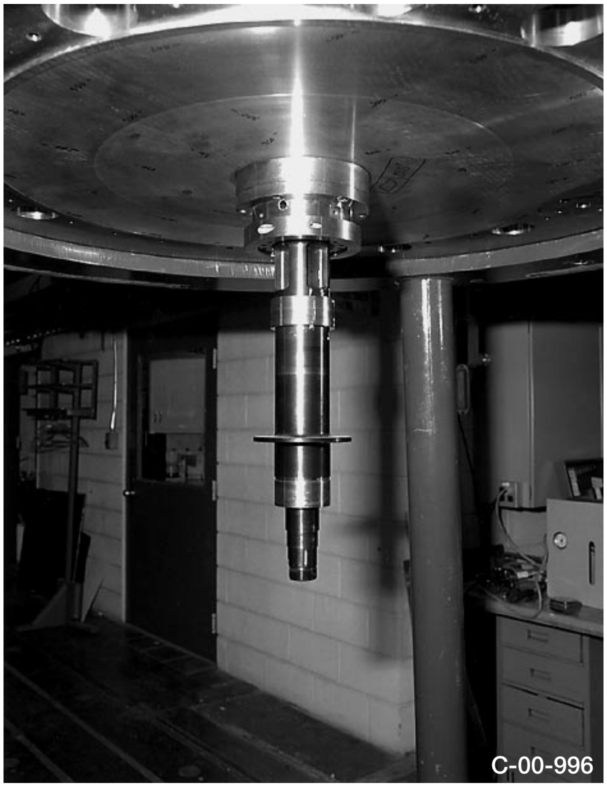


Figure 1.—Rotor without stator assembly.

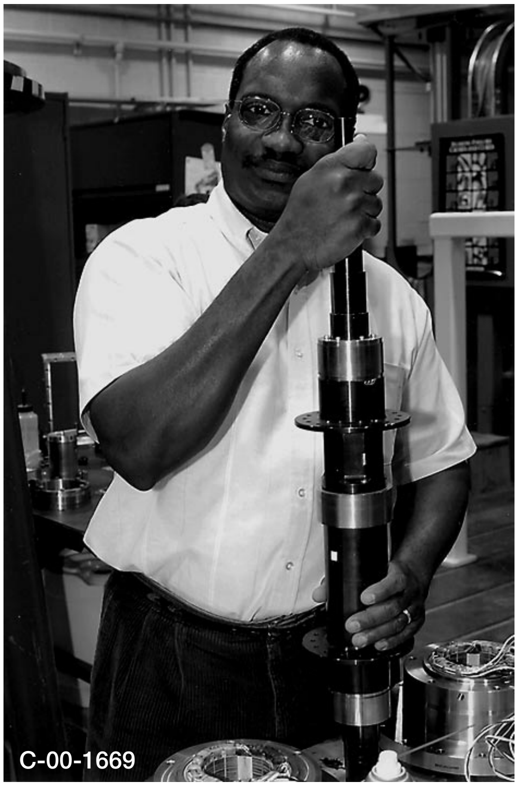


Figure 2.—Rotor juxtaposititton for size comparison.

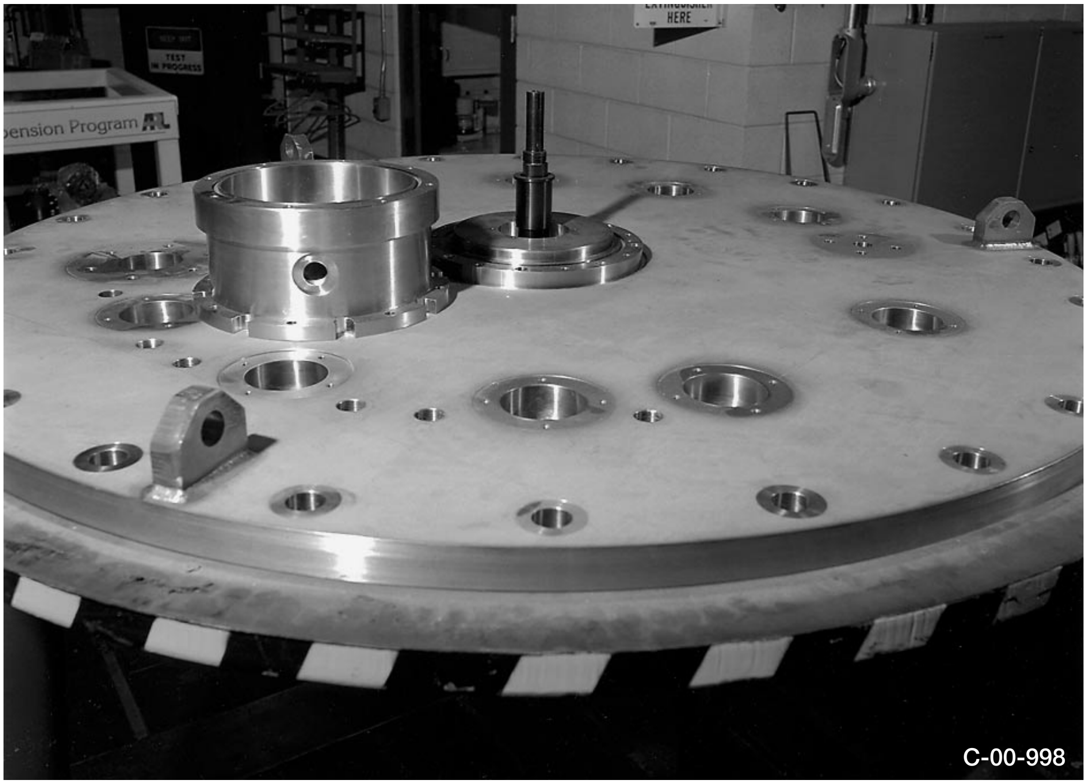


Figure 3.—Top view of rotor without thrust bearing assembly.

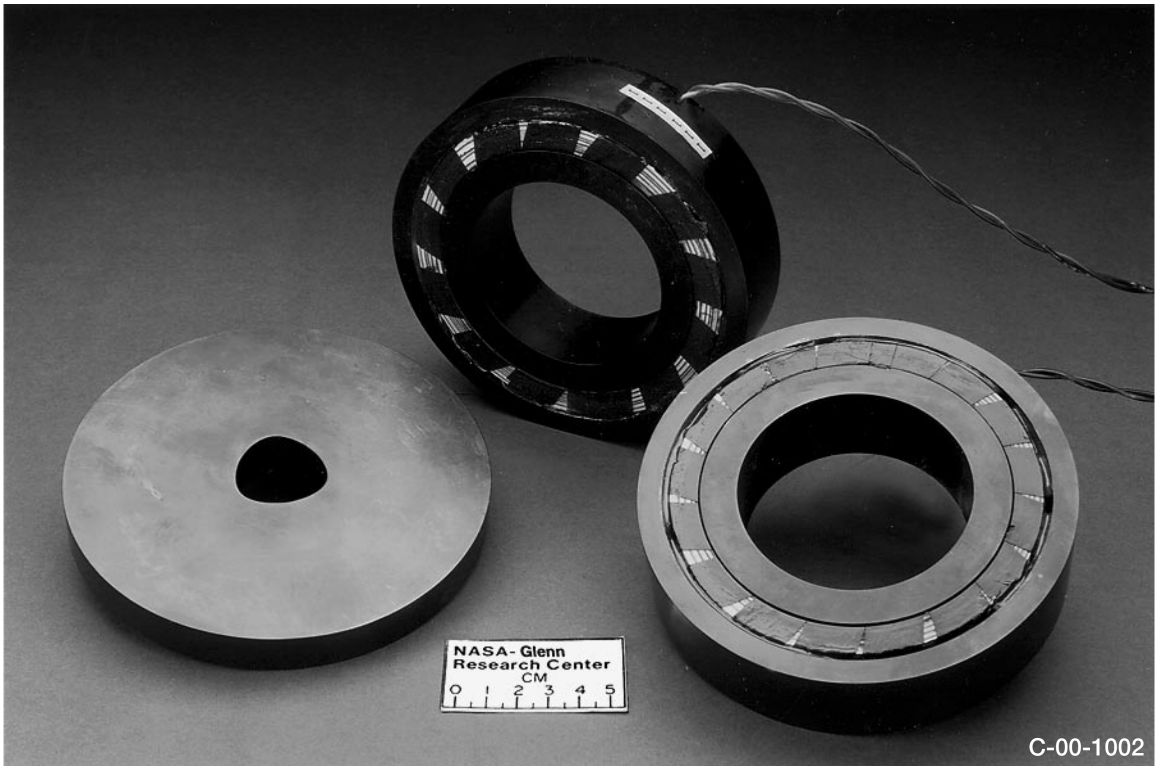


Figure 4.—Thrust plate and thrust coils.

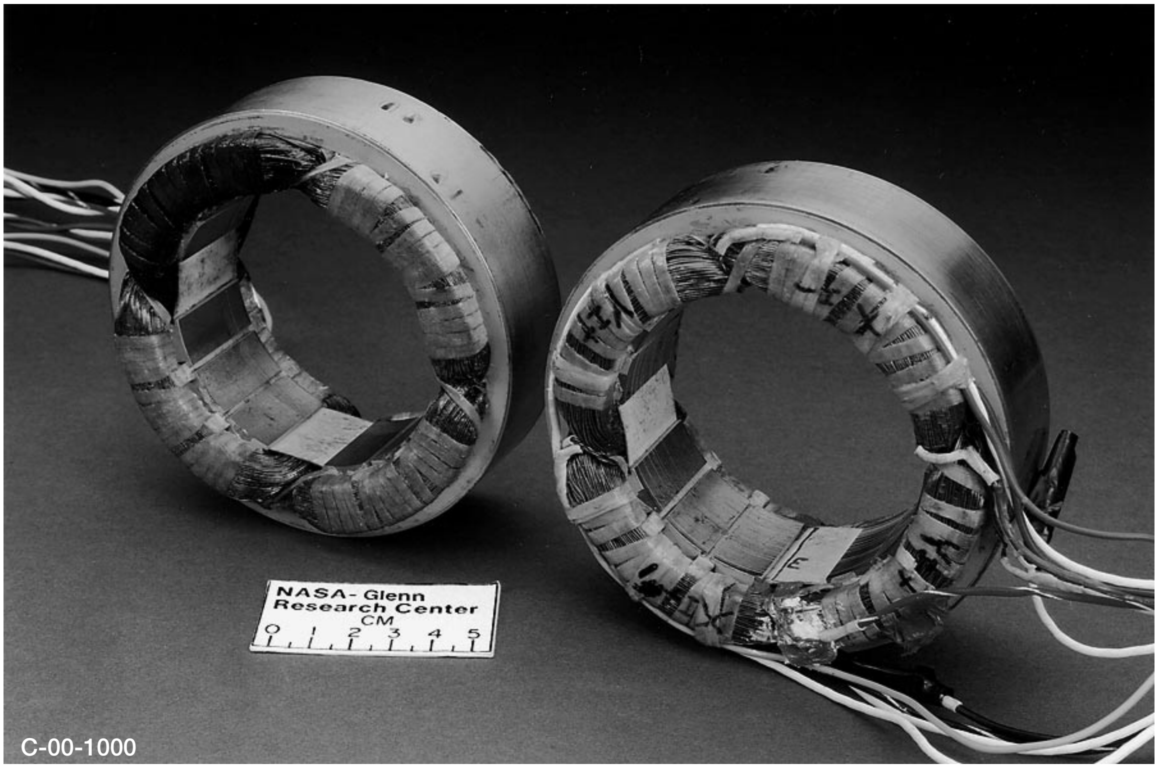


Figure 5.—Upper and lower radial stators.

2.0 MAGNETIC BEARING CONTROL FORCE EQUATIONS

From reference 1, it can be shown that the net controlling force (due to an opposing pair of identical electromagnets) acting on the rotor has the form

$$F = Z\left(\frac{i_1^2}{x_{g1}^2} - \frac{i_2^2}{x_{g2}^2}\right) \tag{1}$$

where

$$Z = \frac{\mu_0 N^2 A}{4} \tag{2}$$

and i_1 and i_2 are the currents in the opposing coils; x_{g1} and x_{g2} are the gap distances between the rotor and each opposing pole face; μ_0 is the permeability of free space; N is the number of coil turns; and A is the pole face area.

The squared terms in equation (1) are undesirable from a control standpoint and are thus eliminated by using a linearizing technique that incorporates a bias current and a control current. By replacing i_1 and i_2 in equation (1) with $(i_b + i_c)$ and $(i_b - i_c)$, respectively, and x_{g1} and x_{g2} with $(x_0 - x)$ and $(x_0 + x)$, respectively, the force equation becomes

$$F = Z\left[\frac{(i_b + i_c)^2}{(x_0 - x)^2} - \frac{(i_b - i_c)^2}{(x_0 + x)^2}\right] \tag{3}$$

where i_b is the bias current, i_c is the control current, x_0 is the nominal gap, and x is the deviation from the nominal value.

After making the appropriate algebraic manipulation and taking the requisite partial derivatives, the force, current, and position are shown to have the linear relationship

$$F_n = K_x x + K_i i \tag{4}$$

where K_x is the position stiffness and K_i the current stiffness. For proportional-derivative (PD) feedback control when an excitation signal is used, i is replaced by $-(K_p x + K_d \dot{x}) + i_{ex}$ where K_p and K_d are the proportional control gain and derivative control gain, respectively, and i_{ex} is the excitation current variable. Equation (4) thus becomes

$$F_{ex} = m_{eq}\ddot{x} + K_i K_d \dot{x} + \left(K_i K_p - K_x\right)x \tag{5}$$

where m_{eq} is the rigid rotor equivalent mass and $F_{ex} = K_i i_{ex}$. Further algebraic simplification produces an expression of the form

$$F_{ex} = m_{eq}\ddot{x} + c_{eq}\dot{x} + k_{eq}x \tag{6}$$

The control force equations used in the code have a form similar to this expression, and the offset and the bias current parameters make it possible for an operator to adjust the position and current stiffness, respectively, of the bearings.

3.0 MODAL CONTROL THEORY

Most methods of multimagnetic bearing control rely on independently levitating each end of the rotor. However, modal control is more sophisticated and elegant because it is accomplished by coupling the sensor signals

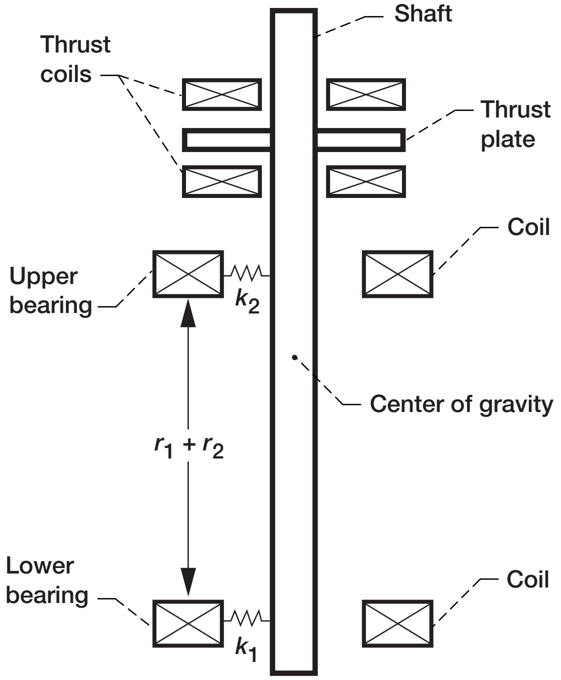


Figure 6.—Five-axis ensemble (not to scale).

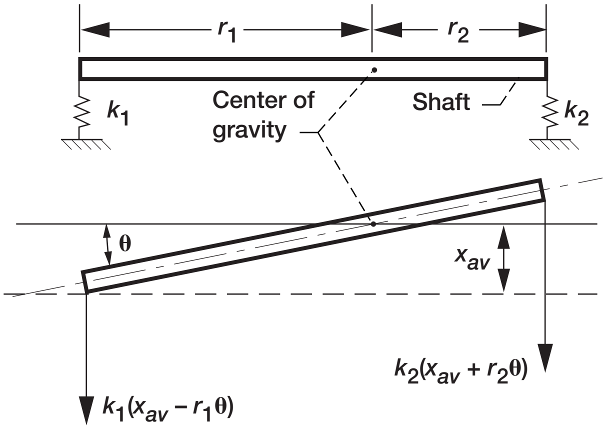


Figure 7.—Free-body diagram for modal control mathematical derivation.

extant at the upper and lower bearings and then using that information to control each bearing. In other words, the rigid rotor motion information (as opposed to the independent motions at the bearings) is used to control the radial bearings.

The rotor and bearings are depicted schematically in figure 6 where the magnetic restoring forces are represented by springs. For the vertically oriented axis, gravity does not affect the radial degrees of freedom (see fig. 7 for the free-body diagram of the shaft motion). The motion of the center of mass (c.m.) (ref. 2) in the *x*,*z*-plane is thus given by Newton's second law as

$$m\ddot{x} = -k_1\left(x_{av} - r_1\theta\right) - k_2\left(x_{av} + r_2\theta\right) - c_1\dot{x}_1 - c_2\dot{x}_2 \tag{7}$$

$$m\ddot{x} = -\left(k_1 + k_2\right)x_{av} - \left(k_2r_2 - k_1r_1\right)\theta - \left(c_1 + c_2\right)\dot{x}_{av} - \left(c_2r_2 - c_1r_1\right)\theta \tag{8}$$

where, for the lower bearing, $k_1 = k_{eq1}$ and for the upper bearing, $k_2 = k_{eq2}$; x_{av} is the average displacement of the center of gravity; r_1 and r_2 are the distances from the ends of the shaft to the center of gravity; θ is the tilt angle; c_1 and c_2 are damping constants, where $c_1 = c_{\mathrm{eq}1}$ and $c_2 = c_{\mathrm{eq}2}$.

The equations relating to shaft centerline tilt displacement in the *x*,*z*-plane are

$$I_G\ddot{\theta} = k_1\left(x_{av} - r_1\theta\right)r_1 - k_2\left(x_{av} + r_2\theta\right)r_2 \tag{9}$$

$$I_G\ddot{\theta} = \left(k_1r_1 - k_2r_2\right)x_{av} - \left(k_2r_2^2 + k_1r_1^2\right)\theta \tag{10}$$

where I_G is the moment of inertia about the center of gravity.

From equations (8) and (10), it is seen that the centralized force equations have the form

$$\text{Force (center of mass translation)} = -\left(k_1 + k_2\right)x_{av} - \left(c_1 + c_2\right)\dot{x}_{av} \tag{11}$$

$$\text{Force (rotation)} = -\left(k_2r_2^2 + k_1r_1^2\right)\theta - \left(c_2r_2^2 + c_1r_1^2\right)\dot{\theta} \tag{12}$$

Hence, the total centralized force is given by

$$\text{Force (total)} = \text{force (center of mass translation)} + \text{force (rotation)} \tag{13}$$

Similar equations apply in the *y*,*z*-plane. Equation (13) was used in the code (source code lines 1887–1891; 1907–1911; and 1915–1925).

4.0 INITIAL COMPUTER HARDWARE REQUIREMENTS

This code was designed to run in the pure DOS mode on any Pentium-class PC having a processor speed of 100 MHz or higher. Robust control at all operating speeds requires a loop time of 100 µs or less. Higher processor speeds, in most instances, trend towards a shorter loop time. A shorter loop time can provide more stable control of the rig at higher rotor speeds. Figure 8 shows the Datel A/D input and Metrabyte D/A output boards as they appear in the back of the central processing unit. The ribbon cables are attached to the output boards and the coaxial cables are connected to the input boards. These boards should be installed in ISA expansion slots (source code lines 86–122 for the input board initial setup and lines 126–158 for the output board initial setup). The channels of the output boards are as indicated in source code lines 136–141 and 153–158, and the channels of the input boards are specified in lines 670–682. There should be 8 input (fig. 8) and 12 output channels (fig. 9). Eleven of the twelve output channels (the zero channel on the upper bearing output box is not used) are actually employed in this rig. The monitor should be an SVGA or better for the best text display. Figure 10 shows the operations center of the five-axis rig.

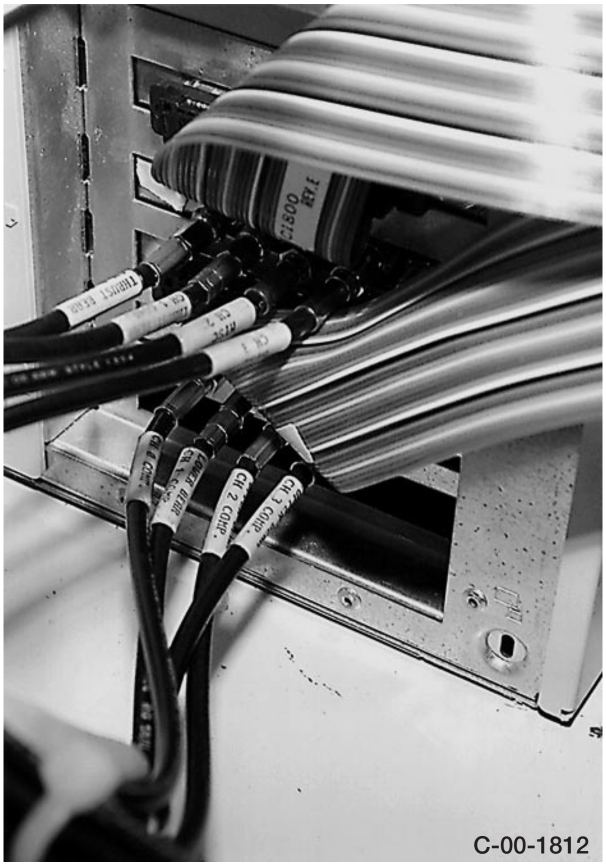


Figure 8.—Input and output board configuration in central processing unit.

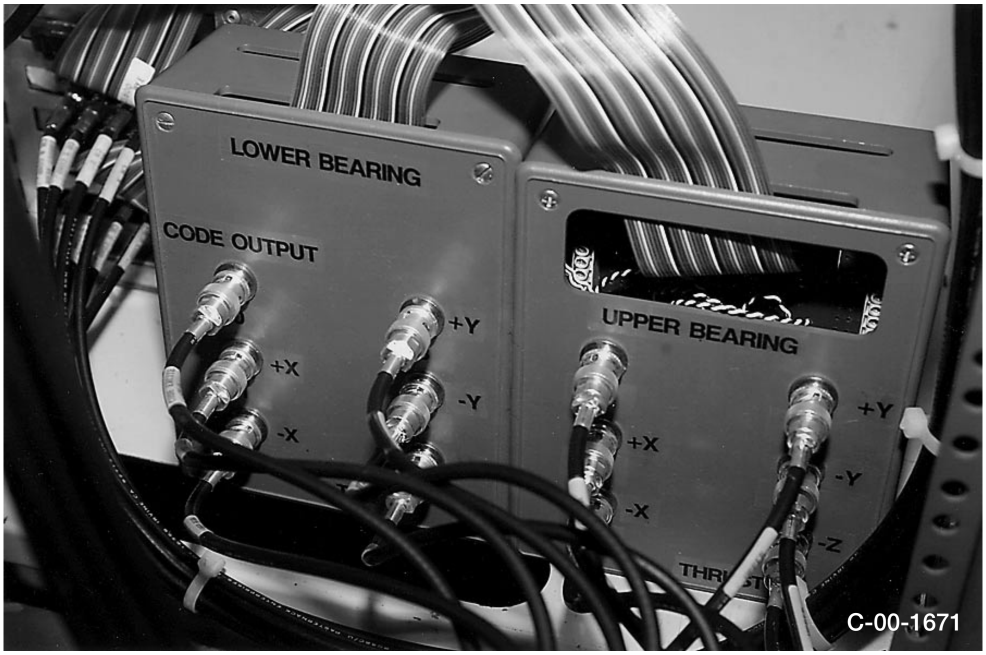


Figure 9.—Twelve-channel output box from central processing unit.

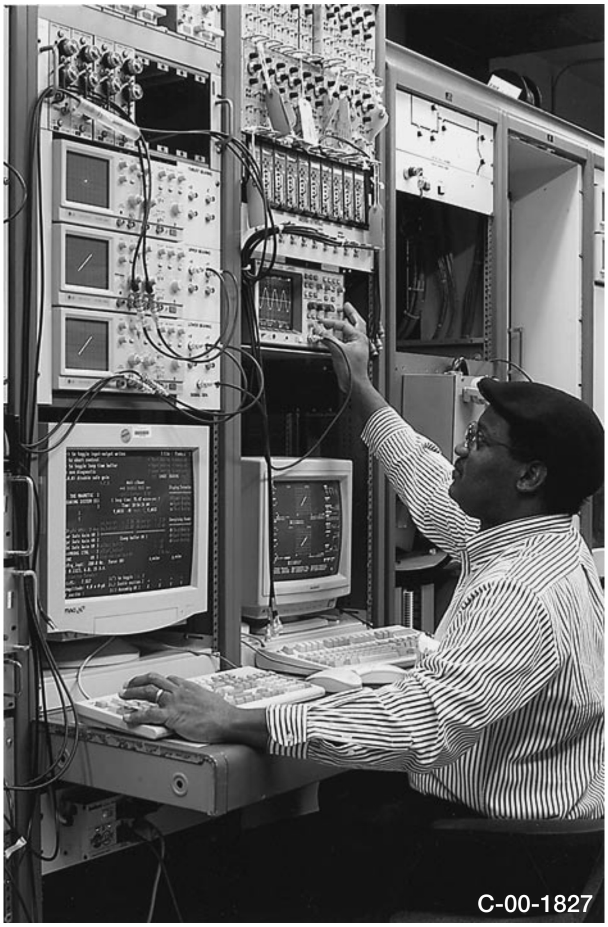


Figure 10.—Operations center for Five-Axis, Three-Magnetic-Bearing Dynamic Spin Rig.

5.0 INITIAL SCREEN DISPLAY PARAMETER

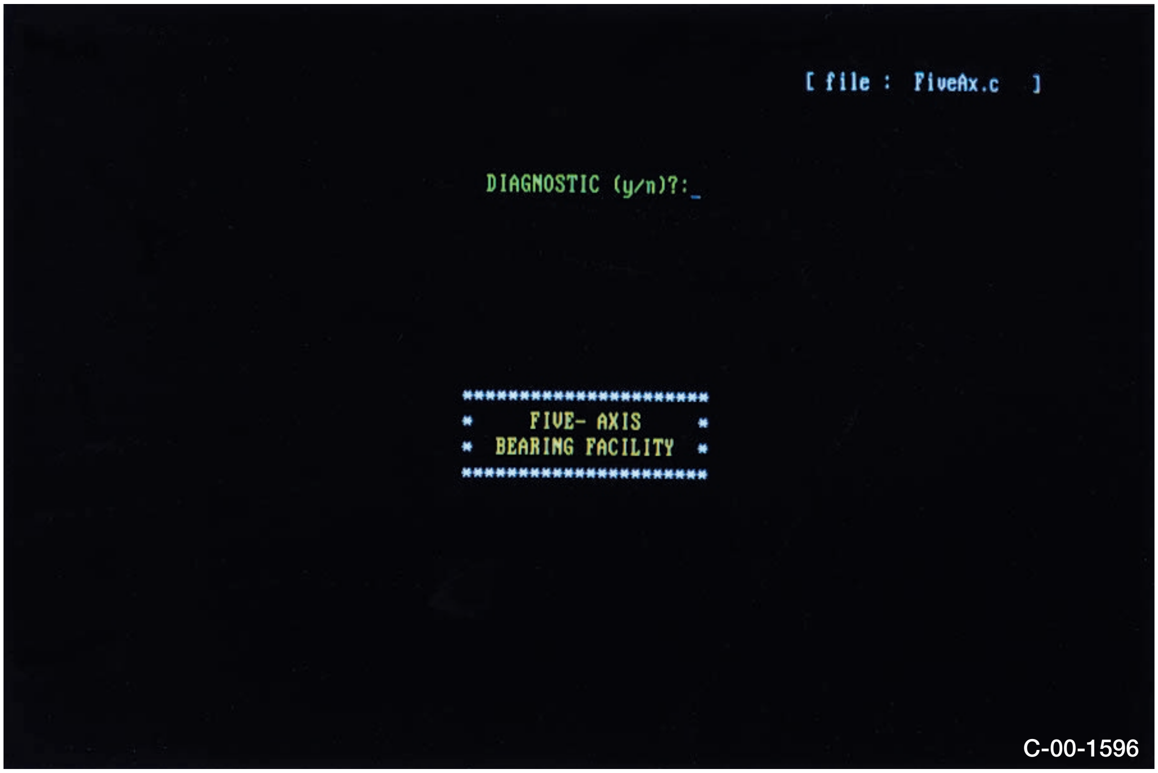
When the file “FiveAx.exe” is launched, “DIAGNOSTIC (y/n)?:” appears on the screen along with a logo of the test facility (fig. 11). If **y** is selected, the screen changes to the diagnostic mode (fig. 12). If **n** is selected, the screen changes to the nondiagnostic mode (fig. 13). The diagnostic mode allows one to make critical adjustments to the rig parameters before and/or during levitation. After setting these parameters, the nondiagnostic display may be toggled. The values of the parameters are preserved on transitioning to the nondiagnostic mode and the screen will be minimally congested. As a rule, *always toggle the diagnostic mode first*. If the nondiagnostic mode is initially toggled, the default values of critical parameters may not be appropriate for a stable levitation of the rotor.

6.0 BEARING ENERGIZING PARAMETERS

If the diagnostic mode is initially selected, the status indicators for the thrust, upper, and lower bearings show that they are not energized (fig. 12). The on/off toggle letters **H**, **I**, **J** (listed below the heading “Energizing Parmtr”) are also blinking. The blinking letters are an aid to quickly identifying the appropriate bearing toggle letter. Energize the bearings, beginning with the thrust bearing, and then energize the upper and then the lower bearing using the on/off toggle letters **H**, **I**, and **J**. The status indicators of the bearings change to red, and the on/off toggle letters no longer blink (fig. 14). The rotor should be in levitation at this point, provided that the gains are correct (see sec. 9.0).

7.0 LOOP BUFFER TOGGLE

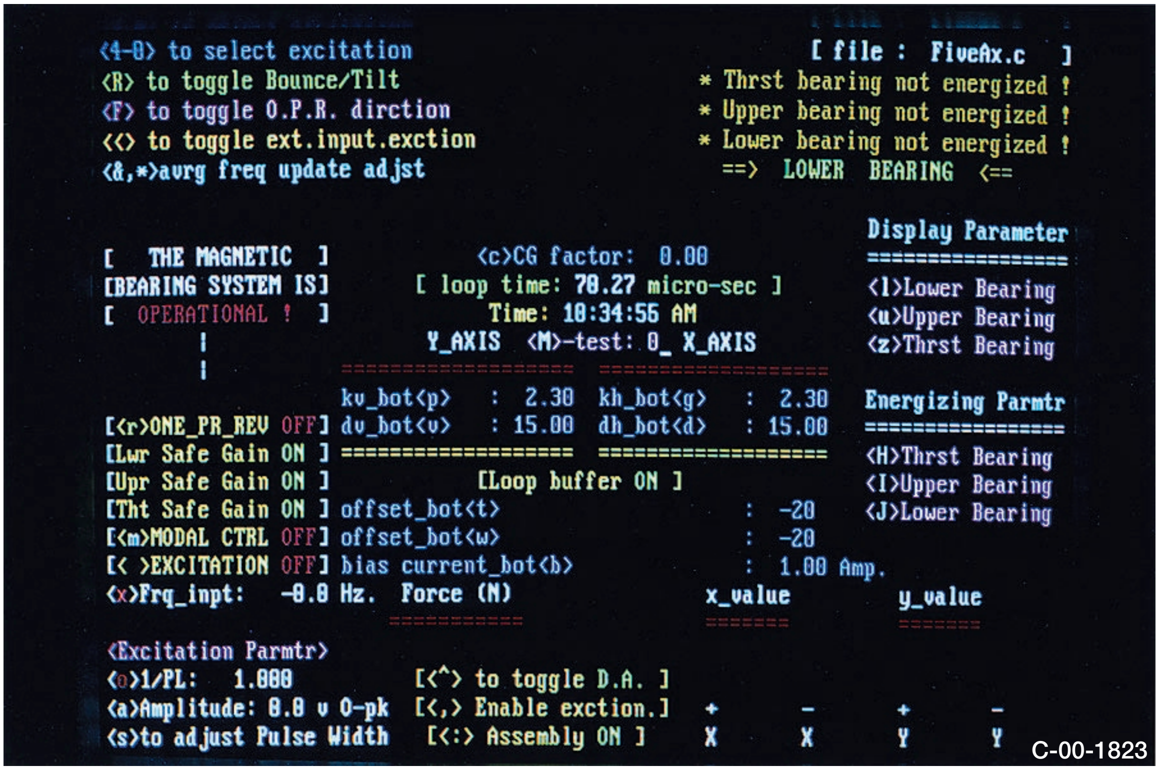
The “Loop buffer” is a series of dummy mathematical statements (source code lines 1513–1518; 1864–1869; and 2663–2666) that automatically activate when one or two of the bearings are deactivated. Its sole purpose is to maintain the loop time of the code, irrespective of the state of the energizing parameters. If loop buffering were not done, the controlling characteristics of the code would change as each bearing is toggled on or off. The code



```
                                [ file :  FiveAx.c   ]


                DIAGNOSTIC (y/n)?:_





               **********************
               *    FIVE- AXIS      *
               *  BEARING FACILITY  *
               **********************
```

C-00-1596

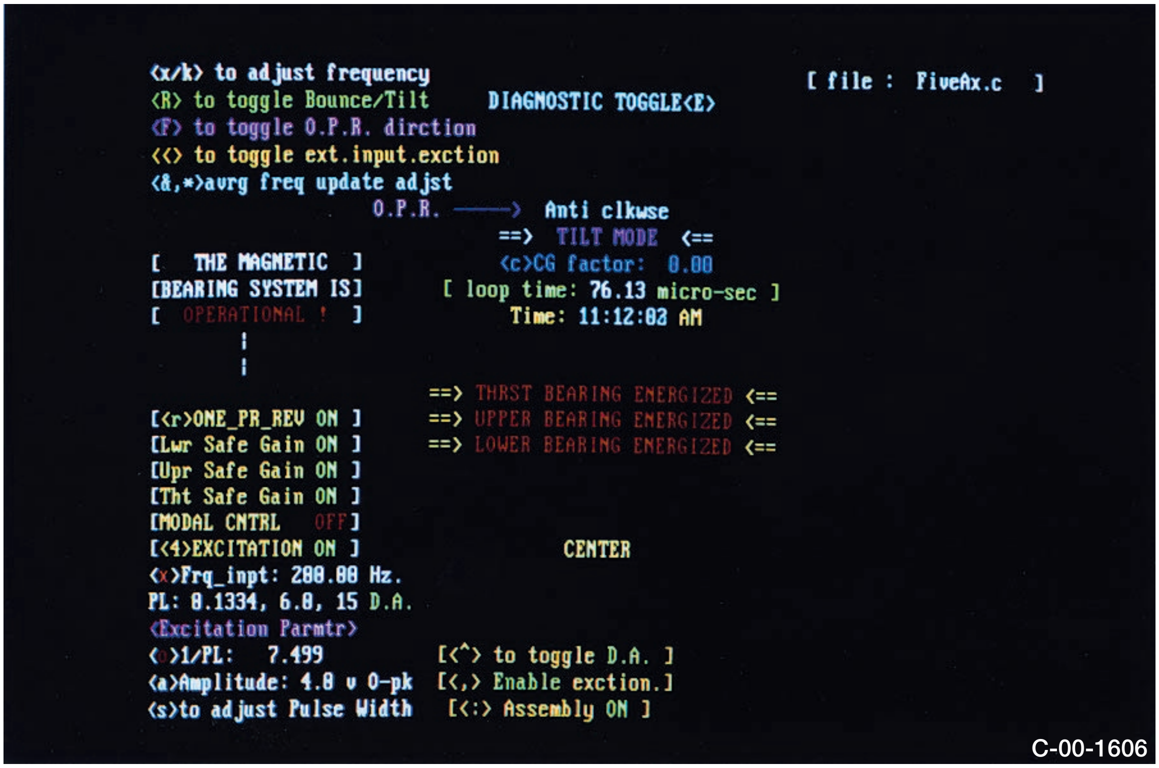
Figure 11.—Initial screen display.



```
<4-0> to select excitation                           [ file :  FiveAx.c   ]
<R> to toggle Bounce/Tilt                        * Thrst bearing not energized !
<F> to toggle O.P.R. dirction                    * Upper bearing not energized !
<<> to toggle ext.input.exction                  * Lower bearing not energized !
<&,*>avrg freq update adjst                        ==>  LOWER  BEARING  <==

                                                               Display Parameter
[   THE MAGNETIC   ]         <c>CG factor:  0.00               =================
[BEARING SYSTEM IS]      [ loop time: 70.27 micro-sec ]        <l>Lower Bearing
[  OPERATIONAL !   ]            Time: 10:34:55 AM              <u>Upper Bearing
        |                  Y_AXIS  <M>-test: 0_ X_AXIS         <z>Thrst Bearing
        |            ====================  ====================
                     kv_bot<p>    :  2.30  kh_bot<g>    :  2.30  Energizing Parmtr
[<r>ONE_PR_REV OFF] dv_bot<v>    : 15.00  dh_bot<d>    : 15.00  =================
[Lwr Safe Gain ON ] ====================  ====================  <H>Thrst Bearing
[Upr Safe Gain ON ]          [Loop buffer ON ]                  <I>Upper Bearing
[Tht Safe Gain ON ] offset_bot<t>                      :  -20   <J>Lower Bearing
[<m>MODAL CTRL OFF] offset_bot<w>                      :  -20
[< >EXCITATION OFF] bias current_bot<b>                :  1.00 Amp.
<x>Frq_inpt:   -0.0 Hz.  Force (N)            x_value        y_value
                        ===========           =======        =======
<Excitation Parmtr>
<o>1/PL:    1.000        [<^> to toggle D.A. ]
<a>Amplitude: 0.0 v 0-pk [<,> Enable exction.]  +       -       +       -
<s>to adjust Pulse Width  [<:> Assembly ON ]    X       X       Y       Y
```

C-00-1823

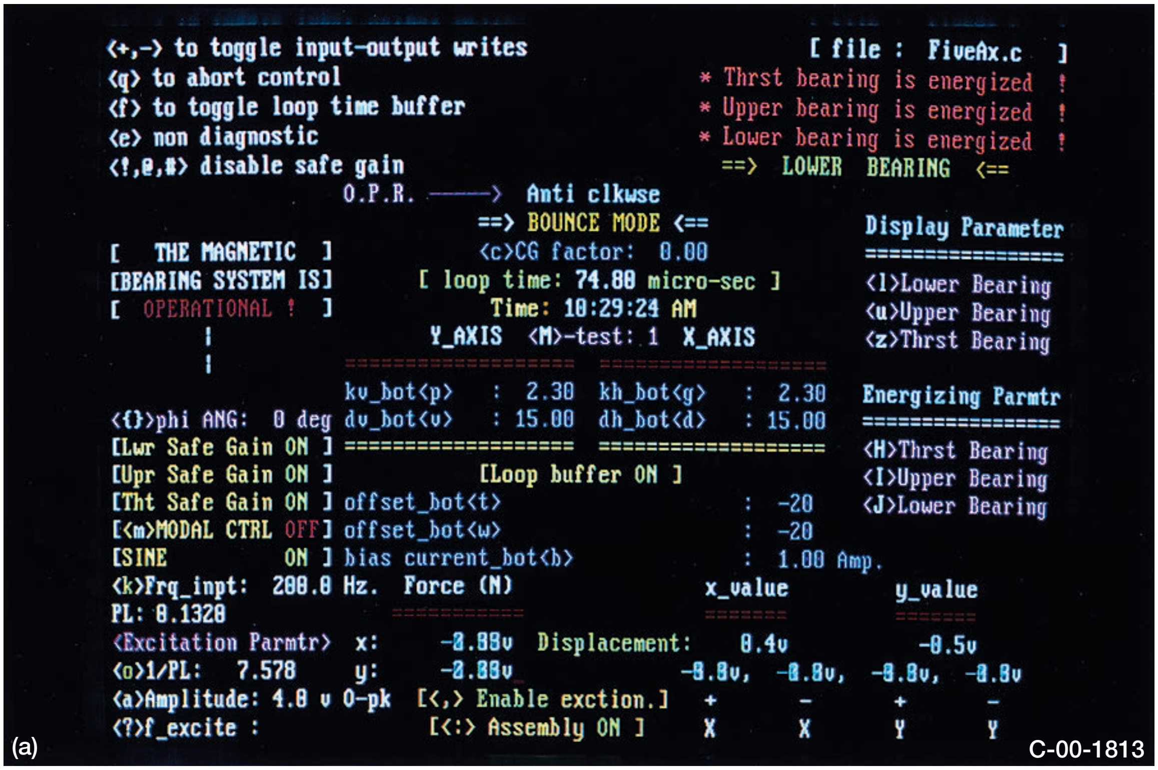
Figure 12.—Initial diagnostic mode screen display.



```
<x/k> to adjust frequency                                  [ file :  FiveAx.c   ]
<R> to toggle Bounce/Tilt      DIAGNOSTIC TOGGLE<E>
<F> to toggle O.P.R. dirction
<<> to toggle ext.input.exction
<&,*>avrg freq update adjst
                  O.P.R. ———> Anti clkwse
                             ==>  TILT MODE  <==
[   THE MAGNETIC   ]          <c>CG factor:  0.00
[BEARING SYSTEM IS]      [ loop time: 76.13 micro-sec ]
[  OPERATIONAL !   ]            Time: 11:12:02 AM
        |
        |
                         ==> THRST BEARING ENERGIZED <==
[<r>ONE_PR_REV ON ]      ==> UPPER BEARING ENERGIZED <==
[Lwr Safe Gain ON ]      ==> LOWER BEARING ENERGIZED <==
[Upr Safe Gain ON ]
[Tht Safe Gain ON ]
[MODAL CNTRL   OFF]
[<4>EXCITATION ON ]                   CENTER
<x>Frq_inpt: 200.00 Hz.
PL: 0.1334, 6.0, 15 D.A.
<Excitation Parmtr>
< >1/PL:    7.499          [<^> to toggle D.A. ]
<a>Amplitude: 4.0 v 0-pk   [<,> Enable exction.]
<s>to adjust Pulse Width    [<:> Assembly ON ]
```

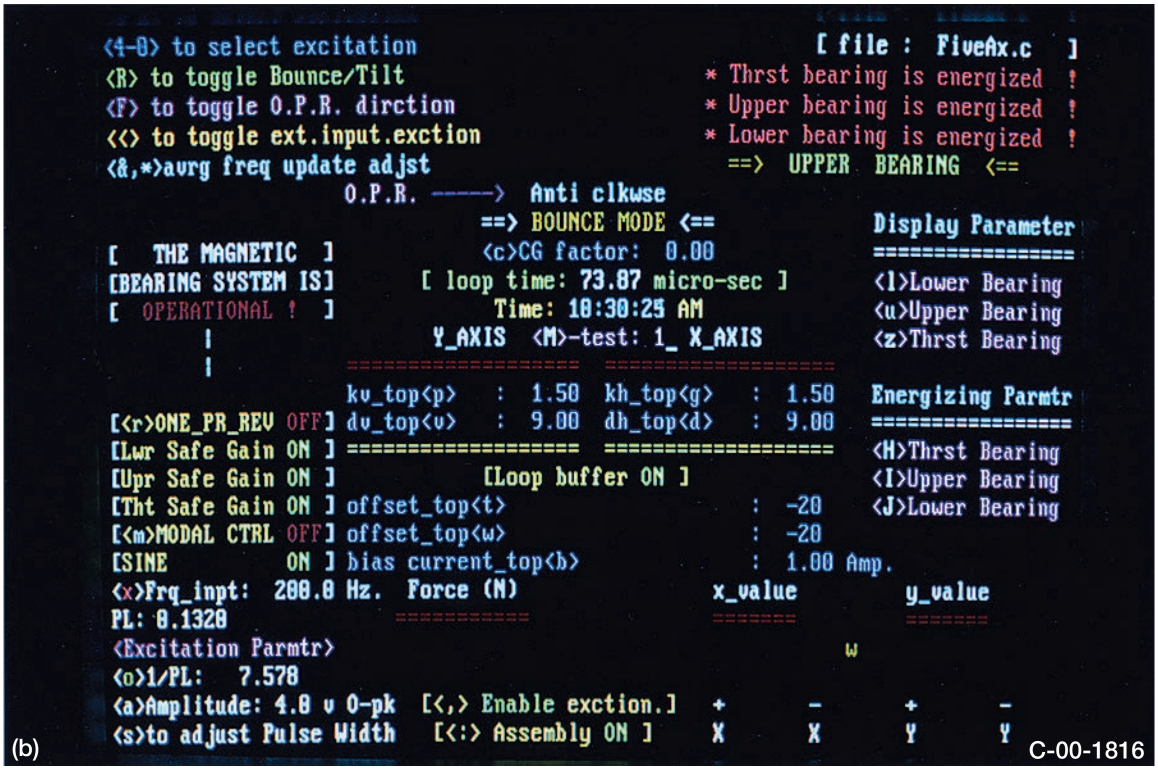
C-00-1606

Figure 13.—Nondiagnostic screen display.



```
<+,-> to toggle input-output writes                     [ file :  FiveAx.c   ]
<q> to abort control                                * Thrst bearing is energized  !
<f> to toggle loop time buffer                      * Upper bearing is energized  !
<e> non diagnostic                                  * Lower bearing is energized  !
<!,@,#> disable safe gain                             ==>  LOWER  BEARING  <==
                             O.P.R. ----->  Anti clkwse
                                    ==> BOUNCE MODE <==                Display Parameter
[   THE MAGNETIC   ]                <c>CG factor:  0.00                =================
[BEARING SYSTEM IS]             [ loop time: 74.80 micro-sec ]         <l>Lower Bearing
[  OPERATIONAL !   ]                 Time: 10:29:24 AM                 <u>Upper Bearing
        |                        Y_AXIS  <M>-test: 1  X_AXIS           <z>Thrst Bearing
        |                    ====================  ===================
                             kv_bot<p>    :  2.30  kh_bot<g>    :  2.30  Energizing Parmtr
<{}>phi ANG:  0 deg dv_bot<v>    : 15.00  dh_bot<d>    : 15.00  =================
[Lwr Safe Gain ON ] ====================  ===================  <H>Thrst Bearing
[Upr Safe Gain ON ]                [Loop buffer ON ]                  <I>Upper Bearing
[Tht Safe Gain ON ] offset_bot<t>                          :  -20     <J>Lower Bearing
[<m>MODAL CTRL OFF] offset_bot<w>                          :  -20
[SINE          ON ] bias current_bot<b>                    :  1.00 Amp.
<k>Frq_inpt:  200.0 Hz.  Force (N)                 x_value          y_value
PL: 0.1320                ===========              =======          =======
<Excitation Parmtr>  x:      -0.00v  Displacement:    0.4v             -0.5v
<o>1/PL:    7.578    y:      -0.00v             -0.0v,  -0.0v,  -0.0v,  -0.0v
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]   +       -       +       -
<?>f_excite :              [<:> Assembly ON ]     X       X       Y       Y
```

(a) C-00-1813



```
<4-0> to select excitation                               [ file :  FiveAx.c   ]
<R> to toggle Bounce/Tilt                            * Thrst bearing is energized  !
<F> to toggle O.P.R. dirction                        * Upper bearing is energized  !
<<> to toggle ext.input.exction                      * Lower bearing is energized  !
<&,*>avrg freq update adjst                            ==>  UPPER  BEARING  <==
                             O.P.R. ----->  Anti clkwse
                                    ==> BOUNCE MODE <==                Display Parameter
[   THE MAGNETIC   ]                <c>CG factor:  0.00                =================
[BEARING SYSTEM IS]             [ loop time: 73.07 micro-sec ]         <l>Lower Bearing
[  OPERATIONAL !   ]                 Time: 10:30:25 AM                 <u>Upper Bearing
        |                        Y_AXIS  <M>-test: 1_ X_AXIS           <z>Thrst Bearing
        |                    ====================  ===================
                             kv_top<p>    :  1.50  kh_top<g>    :  1.50  Energizing Parmtr
[<r>ONE_PR_REV OFF] dv_top<v>    :  9.00  dh_top<d>    :  9.00  =================
[Lwr Safe Gain ON ] ====================  ===================  <H>Thrst Bearing
[Upr Safe Gain ON ]                [Loop buffer ON ]                  <I>Upper Bearing
[Tht Safe Gain ON ] offset_top<t>                          :  -20     <J>Lower Bearing
[<m>MODAL CTRL OFF] offset_top<w>                          :  -20
[SINE          ON ] bias current_top<b>                    :  1.00 Amp.
<x>Frq_inpt:  200.0 Hz.  Force (N)                 x_value          y_value
PL: 0.1320                ===========              =======          =======
<Excitation Parmtr>                                           w
<o>1/PL:    7.578
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]   +       -       +       -
<s>to adjust Pulse Width   [<:> Assembly ON ]     X       X       Y       Y
```

(b) C-00-1816

Figure 14.—Diagnostic mode screen displays for upper and lower bearings. (a) Lower bearing. (b) Upper bearing.

executes successively faster as each bearing in turn is de-energized. The variation in the controlling characteristic is undesirable if diagnostic tests are to be performed during the levitation of one or two bearings. The changes in the control characteristic are due, in large part, to the action of the derivative terms present in the force equations (source code lines 1181–1186 and 1325–1327; 1333–1338 and 1477–1479; 1532–1537 and 1676–1678; 1684–1689 and 1828–1830; 2481–2487 and 2626–2628). Note that the loop buffer defaults ON.

8.0 ASSEMBLY TOGGLE

The goal in designing this code was to achieve full rotor levitation and control with a minimum loop time of 50 µs. The loop time of 68 µs was attained on a 533-MHz PC and was further reduced to 65 µs by coding the input/output statements of the boards in assembly language. The actual percentage improvement from using assembly vis-à-vis C++, however, will depend on the type of processor employed in running the code. One tends to see progressively less benefit as the processor speed increases. The fastest Pentium-class machines (1 GHz and higher, where the minimum loop time observed was 56 µs) showed marginal to no improvement with the code running in the assembly mode. The greatest percentage improvement was achieved with a 486 machine on which a 13-µs loop time reduction was observed using assembly statements, albeit, the minimum loop time was more than 400 µs. It should be noted that the assembly mode is the default state of the code. Press the **Shift** and **:** keys to toggle the assembly mode; see display “[<:>Assembly ON].”

9.0 STIFFNESS AND DAMPING GAIN ADJUSTMENT

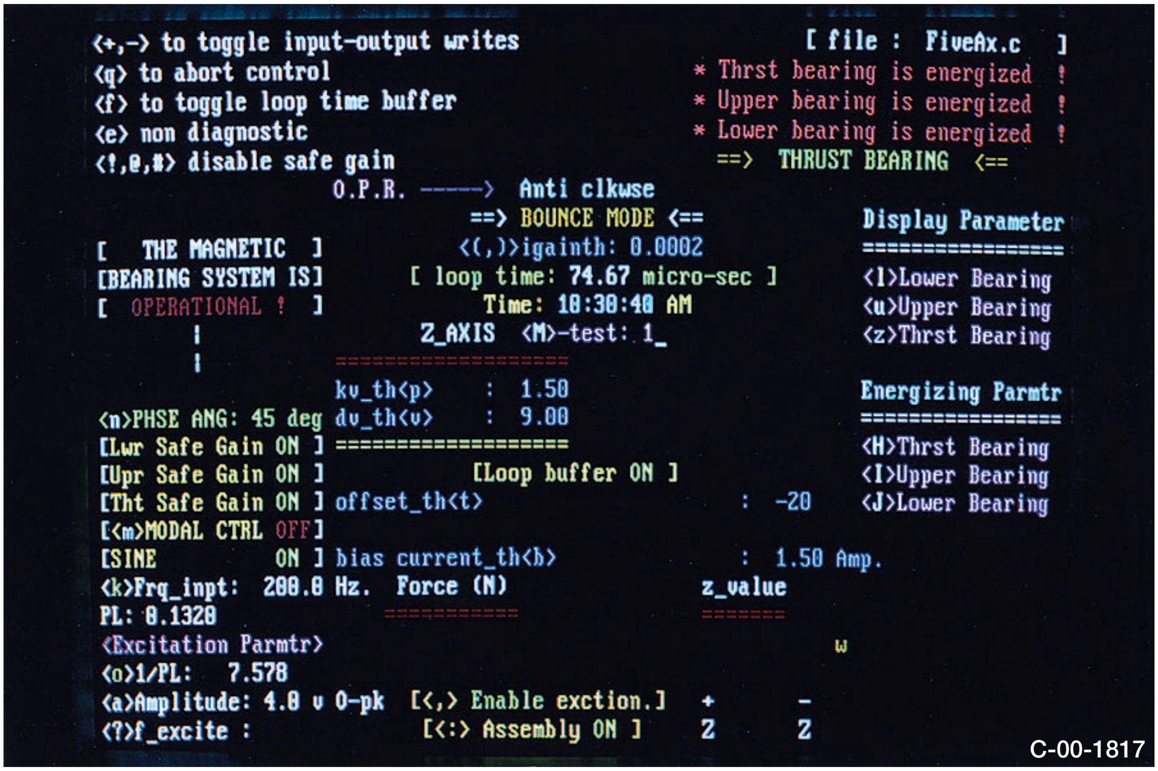
The default values for the stiffness (proportional control gain) and damping (derivative control gain) may not be appropriate for stable levitation (source code lines 1185, 1336, 1536, 1688, and 2486). Hence, these values may have to be adjusted until the rotor position, as observed on the oscilloscopes and/or on the spectrum analyzer, is within the safe zone area and is well damped. Note that the lower bearing parameters are initially displayed (fig. 12). Press the **p** and **g** keys to increase the stiffness values along the *y*- and *x*-axes respectively, and press the **v** and **d** keys to increase the damping values along the *y*- and *x*-axes, respectively. Decrease the stiffness/damping values by depressing the **Shift** key while simultaneously pressing said keys. If necessary, select the upper bearing display by pressing the **u** key and repeat the procedure just described. Press the **z** key to display the thrust bearing parameters. Make any necessary adjustment to the thrust bearing parameter values. The menu for selecting each bearing parameter display is listed under the header “Display Parameter.” Each bearing display toggle letter blinks after its selection.

10.0 OFFSET ADJUSTMENT

The equilibrium position of the rotor is adjusted by varying the offset parameters “offset_bot<t>” and “offset_bot<w>” (fig. 14(a)); “offset_top<t>” and “offset_top<w>” (fig. 14(b)); and “offset_th<t>” (fig. 15). If the lower bearing parameters are initially displayed, press the **t** and **w** keys to increase the offset values along the bearing *x*- and *y*-axes, respectively. Decrease the offset values of the bearing by depressing the **Shift** key while simultaneously pressing said keys. Repeat this procedure for the upper and thrust bearings. There is no “offset_th<w>” parameter for the thrust bearing as it has only one axis of motion (i.e., its direction is along the ±*z*, or axial, axis). Pressing these keys will incrementally move the rotor along the *x*-, *y*-, and *z*-axes. Adjust the position of the rotor until it is in the center of each bearing (as observed on the oscilloscopes in fig. 16).

11.0 BIAS CURRENT ADJUSTMENT

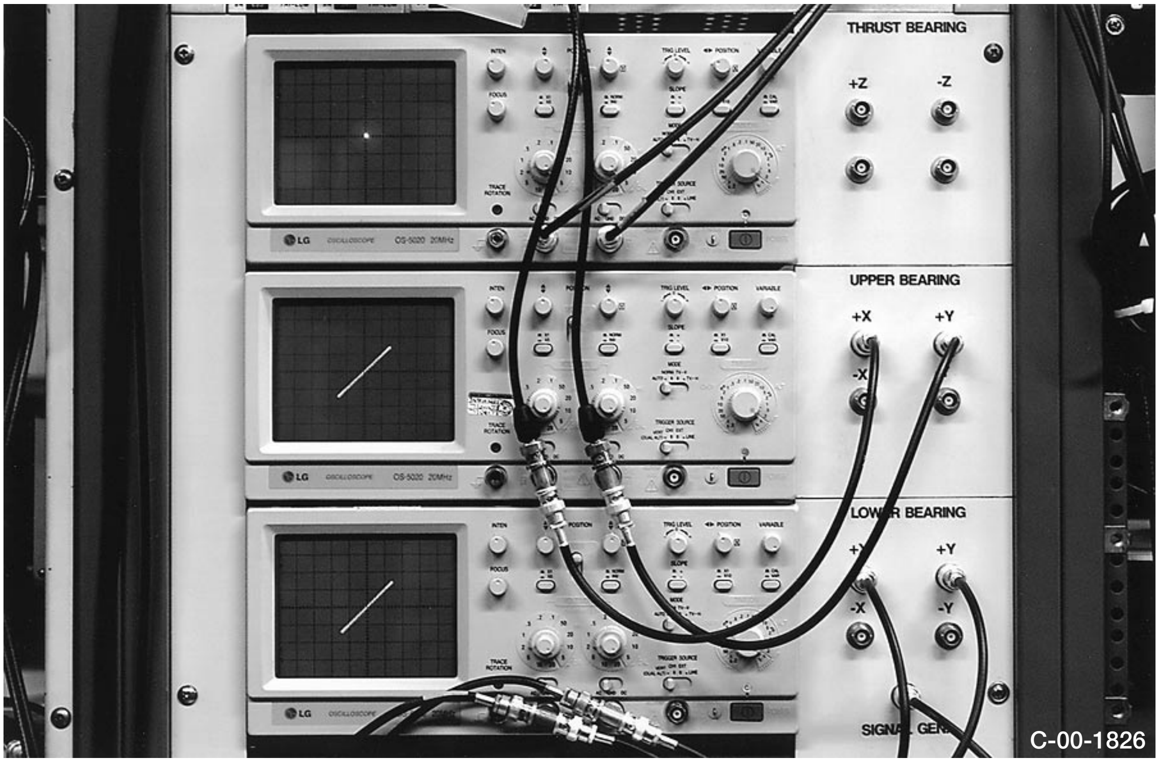
For the Five-Axis, Three-Magnetic-Bearing DSR, the bias current should be kept at its default value of 1.0 A for the lower and upper bearings (figs. 14(a) and (b)) and at 1.5 A for the thrust bearing (fig. 15). If needed, press the **b** key to increase the bias current value. Decrease the bias current by depressing the **Shift** key while simultaneously pressing the **b** key (source code lines 1187, 1188, 1538, 1539, 2488, and 2489).



```
<+,-> to toggle input-output writes                          [ file :  FiveAx.c   ]
<q> to abort control                                       * Thrst bearing is energized  !
<f> to toggle loop time buffer                             * Upper bearing is energized  !
<e> non diagnostic                                         * Lower bearing is energized  !
<!,@,#> disable safe gain                                    ==>  THRUST BEARING  <==
                  O.P.R. -----> Anti clkwse
                           ==> BOUNCE MODE <==                 Display Parameter
[   THE MAGNETIC   ]        <(,)>igainth: 0.0002               =================
[BEARING SYSTEM IS]      [ loop time: 74.67 micro-sec ]        <l>Lower Bearing
[  OPERATIONAL !   ]            Time: 10:30:40 AM              <u>Upper Bearing
        !                   Z_AXIS  <M>-test: 1_               <z>Thrst Bearing
        !         ===================
                  kv_th<p>     :  1.50                         Energizing Parmtr
<n>PHSE ANG: 45 deg dv_th<v>   :  9.00                         =================
[Lwr Safe Gain ON ] ===================                        <H>Thrst Bearing
[Upr Safe Gain ON ]            [Loop buffer ON ]               <I>Upper Bearing
[Tht Safe Gain ON ] offset_th<t>                    :  -20     <J>Lower Bearing
[<m>MODAL CTRL OFF]
[SINE         ON ] bias current_th<b>               :  1.50 Amp.
<k>Frq_inpt:  200.0 Hz.  Force (N)              z_value
PL: 0.1320               ===========            =======
<Excitation Parmtr>                                           w
<o>1/PL:   7.578
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]   +      -
<?>f_excite :              [<:> Assembly ON ]     Z      Z
```

C-00-1817

Figure 15.—Diagnostic mode screen display for thrust bearing.



C-00-1826

Figure 16.—Position display screen for thrust, upper, and lower bearings.

12.0 INTEGRAL GAIN

When the thrust bearing display is toggled (fig. 15), the parameter "<c>CG factor:" display (figs. 14(a) and (b) under the "BOUNCE MODE") is replaced with the "<(,)>igainth:" parameter. This parameter enables adjustment of the integral gain term present in the thrust bearing force equation (source code lines 2482 and 2486). If there is an axial offset of the rotor (i.e., ±*z* about the zero probe position), the integral gain term has the effect of automatically restoring the thrust plate to its zero probe or equilibrium position. A higher value of the integral gain will result in a quicker restoration to the equilibrium position. Press the **(** key to decrease the igainth value or press the **)** key to increase the igainth value.

13.0 CENTER OF GRAVITY ADJUSTMENT OPTION

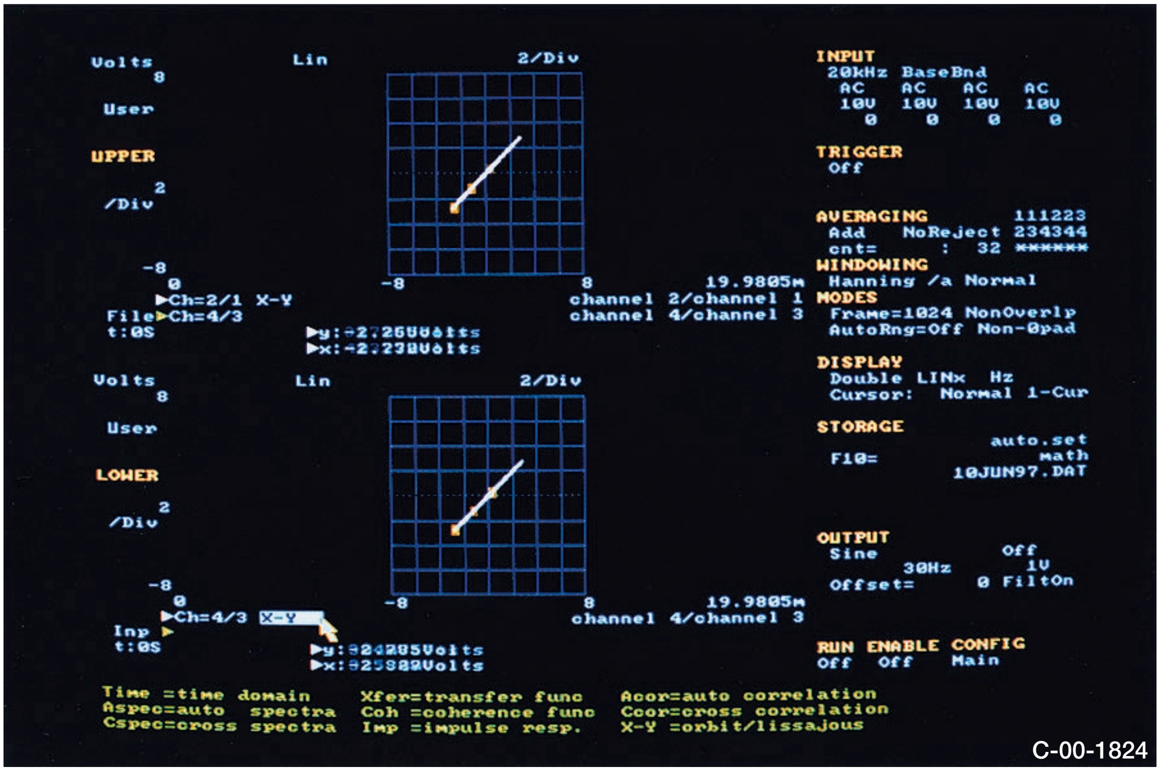
The rotor has a relatively massive thrust plate affixed to its top end. Attaching a massive test article to the rotor effectively shifts its c.m. towards the test article. Consequently, the c.m. of the rotor is not generally at its geometric center. Because this shift in the c.m. can adversely affect the stability of the rotor, it must be taken into account, especially in the centralized (modal) control mode. Press the **l** key to display the screen depicted in figure 14(a) and then press the **c** key to effect appropriate weighting of the outputs to the upper and lower bearings. The "<c>CG factor:" parameter has a default value of 0.00 and can vary between –0.5 and +0.5. Values above zero correspond to a c.m. closer to the upper bearing, and values below zero correspond to a c.m. closer to the lower bearing. The bearing closer to the c.m. should exert a greater force than the bearing farther from the c.m. Adjust the "<c>CG factor:" based on either an experimental measurement or a finite-element analysis to determine its position. See source code lines 1185 and 1536 where MCG and PCG, respectively, are the "<c>CG factor:" variables.

14.0 ROTOR EXCITATION IN STATIONARY AND ROTATING FRAMES

The code is designed to apply excitation signals concurrently to the upper and lower bearings. At each bearing, excitation signals are applied simultaneously to the *x*- and *y*-axes. This simultaneous excitation produces a resultant force vector with a magnitude and an angular orientation. The direction of this force vector can be fixed in a nonrotating frame of reference by setting the desired phase angle ("<n>PHSE ANG:" in fig. 15). The force vector can also be made to rotate with the test article by engaging the "[<r>ONE_PR_REV]" logic block (fig. 14(b) and source code lines 1082–1117). This block of code makes it possible to synchronize a rotating force vector with the rotation of the shaft. A tiny mirror attached to the shaft reflects a pulse of laser light once every rotation of the shaft. A sensor then converts the light pulses to electrical pulses. These pulses are sent to an input channel on a Datel board where they are used to trigger the "[<r>ONE_PR_REV]" logic block (the "[<r>ONE_PR_REV]" signal is applied to channel 2 on the Datel input board at address 0x366). The logic block calculates the angular rotation of the shaft during one loop time of the code (source code line 1094) based on the number of loops between successive pulses. The shaft angular rotation per loop is henceforth used to drive the angular rotation of an excitation force vector in synchrony with the rotating shaft (source code line 1048). The rotating force vector can be made to excite at a specified angle ("<{}>phi ANG:" in fig. 14(a)) vis-à-vis the long axis of the test article. The phi angle ranges from 0° to 360°. In addition, the direction of rotation of the force vector can be toggled.

Manual adjustment of the phase angle "<n>PHSE ANG:" in figure 15 (in increments of 5°) is accomplished by pressing the **n** key to increase the angle in the "Anti clkwse" (anticlockwise) direction or by depressing the **Shift** key while simultaneously pressing said key to decrease the angle. The "<{}>phi ANG:" angle in figure 14(a) is increased (in increments of 5°) by pressing the **}** key and is decreased by pressing the **{** key. The "[<r>ONE_PR_REV]" logic (fig. 14(b)) is toggled on or off by pressing the **r** key. Toggle the rotation direction of the force vector by depressing the **Shift** key while pressing the **f** key.

A shaft can be excited in many modes, two common ones being the bounce and tilt. These two modes were implemented in FATMaCC. If the "[<r>ONE_PR_REV]" is engaged, the bounce mode describes a motion that, if the ends of the shaft were traced, approximates a vertical cylinder. In the tilt mode, the excitation force vector in the top bearing is 180° out of phase with the excitation in the lower bearing. Consequently, the shaft centerline traces out a conical surface. Figure 16 shows the paths of the shaft in the bounce or tilt mode and the position of the thrust bearing. In these displays, the "[<r>ONE_PR_REV]" is turned off and the shaft is being excited at a phase angle



```
Volts                 Lin             2/Div      INPUT
     8                                            20kHz BaseBnd
 User                                              AC    AC    AC    AC
                                                   10V   10V   10V   10V
UPPER                                                0     0     0     0
                                                 TRIGGER
     2                                            Off
 /Div
                                                 AVERAGING           111223
                                                  Add   NoReject 234344
                                                  cnt=        :  32 ******
   -8                                            WINDOWING
     0                       -8      8   19.9805m  Hanning /a Normal
    ▶Ch=2/1 X-Y          channel 2/channel 1   MODES
 File▶Ch=4/3             channel 4/channel 3    Frame=1024 NonOverlp
 t:0S              ▶y:-2.7260Volts              AutoRng=Off Non-0pad
                   ▶x:-2.7230Volts
Volts                 Lin             2/Div      DISPLAY
     8                                            Double LINx  Hz
 User                                             Cursor:  Normal 1-Cur

LOWER                                            STORAGE
                                                                auto.set
     2                                            F10=              math
 /Div                                                      10JUN97.DAT

                                                 OUTPUT
                                                  Sine             Off
   -8                                                   30Hz        1V
     0                       -8      8   19.9805m  Offset=        0 FiltOn
    ▶Ch=4/3 X-Y          channel 4/channel 3
 Inp ▶
 t:0S              ▶y:-2.4285Volts              RUN ENABLE CONFIG
                   ▶x:-2.5880Volts              Off  Off    Main
 Time =time domain    Xfer=transfer func   Acor=auto correlation
 Aspec=auto  spectra  Coh =coherence func  Coor=cross correlation
 Cspec=cross spectra  Imp =impulse resp.   X-Y =orbit/lissajous
```

C-00-1824

Figure 17.—Tektronix *x*,*y*-screen display of upper and lower bearing rotor displacement.

(“<n>PHSE ANG:”) of 45°. Figure 17 is the Tektronix *x*,*y*-display of the upper and lower bearing rotor displacement. The tilt/bounce mode is toggled by depressing the **Shift** key while simultaneously pressing the **r** key.

15.0 EXCITATION FUNCTIONS AND FREQUENCY ADJUSTMENT

The heart of the excitation-generating scheme is the sine and cosine functions. The signal block (source code lines 744–986) is designed to produce a periodic signal whose period is proportional to a nondimensional parameter PL, or period length (appendix A). If PL is identically 1.0, the period is equal to the time to perform 500 loops in the code. A loop time of 50 μs yields an excitation frequency of 40 Hz, which is approximated by 500 steps in the output signal. The steps or discreteness is evident in the sine curve depicted in figure 18 where the frequency is 200.6 Hz. Other frequencies are obtained by choosing PL in inverse proportion to the desired frequency. Each loop increments the *x*-value of the function argument by 1.0/500, or 0.002 (source code lines 813, 840, 868, 896, 925, 953, and 983).

For experiments requiring excitation signals, 11 functions are available: sine, sine squared, cosine, cosine squared, random, square pulse train, square wave, triangular wave, square pulse, triangular pulse, or saw tooth (source code lines 744–986). Select the desired function “[< >Excitation ON]”by pressing the number keys [4,5,6,7, 8,9, or 0]. Pressing the number **4** key initially engages the trigonometric block and brings up the “sine” function in an off state. Continually pressing the **4** key cycles through sine squared, cosine, cosine squared, random (fig. 19) and back to sine (fig. 18). To toggle this function block on or off, depress the **Shift** key and simultaneously press the **4** key. Key **5** selects the “square pulse train,” **6** selects the “square wave,” **7** selects the “triangular wave,” **8** selects the “square pulse,” **9** selects the “triangular pulse,” and **0** selects the “saw tooth wave.” See appendix A for an analytical presentation of these functions.

Selecting the **8** key automatically activates the pulse width toggle flag. Pressing the **s** key decreases the pulse width (fig. 12) and depressing the **Shift** key while simultaneously pressing the **s** key increases the pulse width. Functions 5 to 9 and 0 are each toggled off by pressing the respective key.

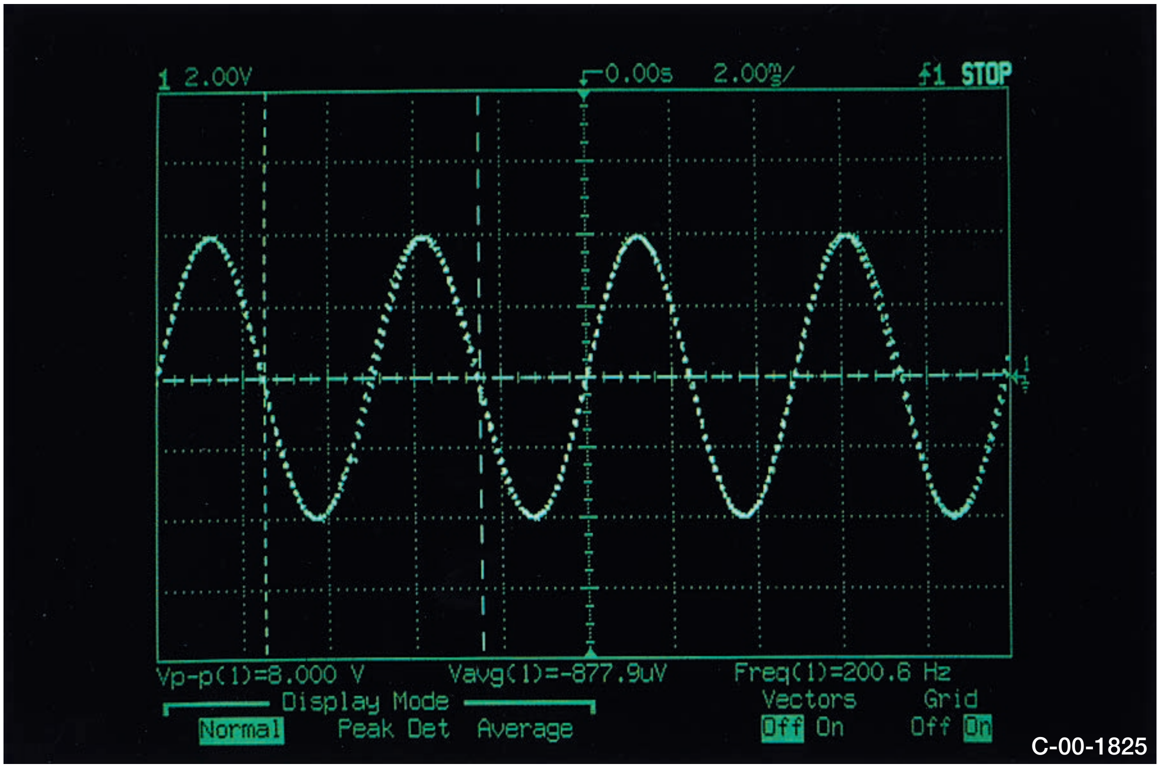


Figure 18.—Hewlett Packard digital scope display of sine curve excitation signal.

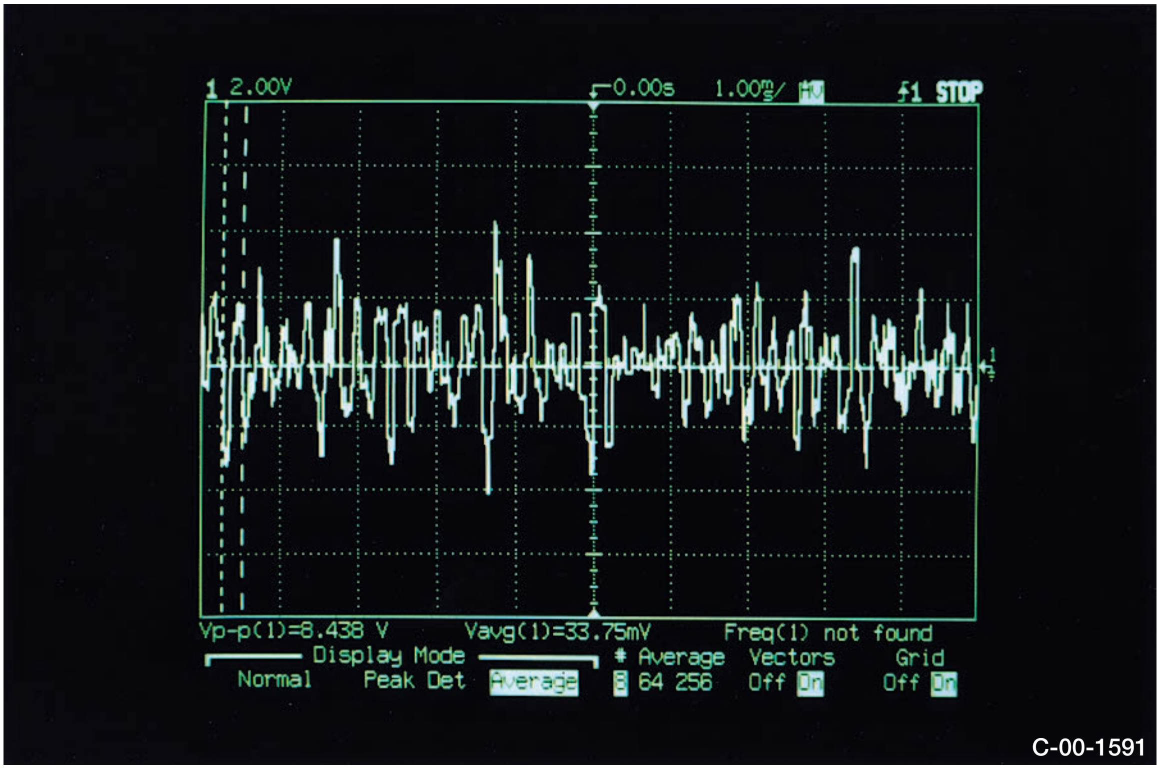


Figure 19.—Hewlett Packard digital scope display of random curve excitation signal.

Input the desired frequency in 10-Hz increments by pressing the **x** key. Press the **k** key to make fine adjustments in 0.1-Hz increments (see fig. 12 display "<x>Frq_inpt:"). The specified frequency is used (in conjunction with the loop time determined from the DOS clock) to generate the signal frequency via the aforementioned functions (source code lines 2859 and 2878). Because the DOS clock is coarse, it tends to cause undesirable variation in the signal frequency. Thus, an averaging method called "dynamic averaging" (D.A.) is employed to improve the stability of the signal frequency. During D.A., the loop time (as measured against the DOS clock) is averaged continuously over 15 successive loop time updates (see sec. 21.0 for the rate of loop time updating). The resulting averaged value is then used in generating the signal frequency. D.A. is automatically engaged when a frequency is inputted using the **x** or **k** key. The D.A. displayed at the end of the period length (PL) field is confirmation of this (fig.13). For D.A., refer to source code lines 2847–2869.

The second option, which may be toggled at any time, is called "intermittent averaging" (I.A.). This method is somewhat less effective on slower processors (those with clock speeds below 533 MHz) in smoothing out the DOS clock variations discussed earlier. The averaging mechanism employed herein requires that the user set the update count limit (UCL). When a value greater than 15 is entered, the code will recalculate the signal frequency at a periodic rate determined by the expression [(UCL – 15) + 15]. This periodicity tends to make the signal frequency change abruptly at each successive update because of slight variations in the averaged loop time values. On the fastest processor (1 GHz and above with an improved DOS clock), this presents less a problem. Whenever a frequency is entered while the code is in the D.A or I.A. mode, the "o" in the parameter "<o>1/PL:" turns red and blinks for the duration of 15 counts. During the red blinking phase, no experimental measurements should be taken as the code is still averaging the loop time. After 15 counts, the "o" turns green and stops blinking. Measurements should resume at this point.

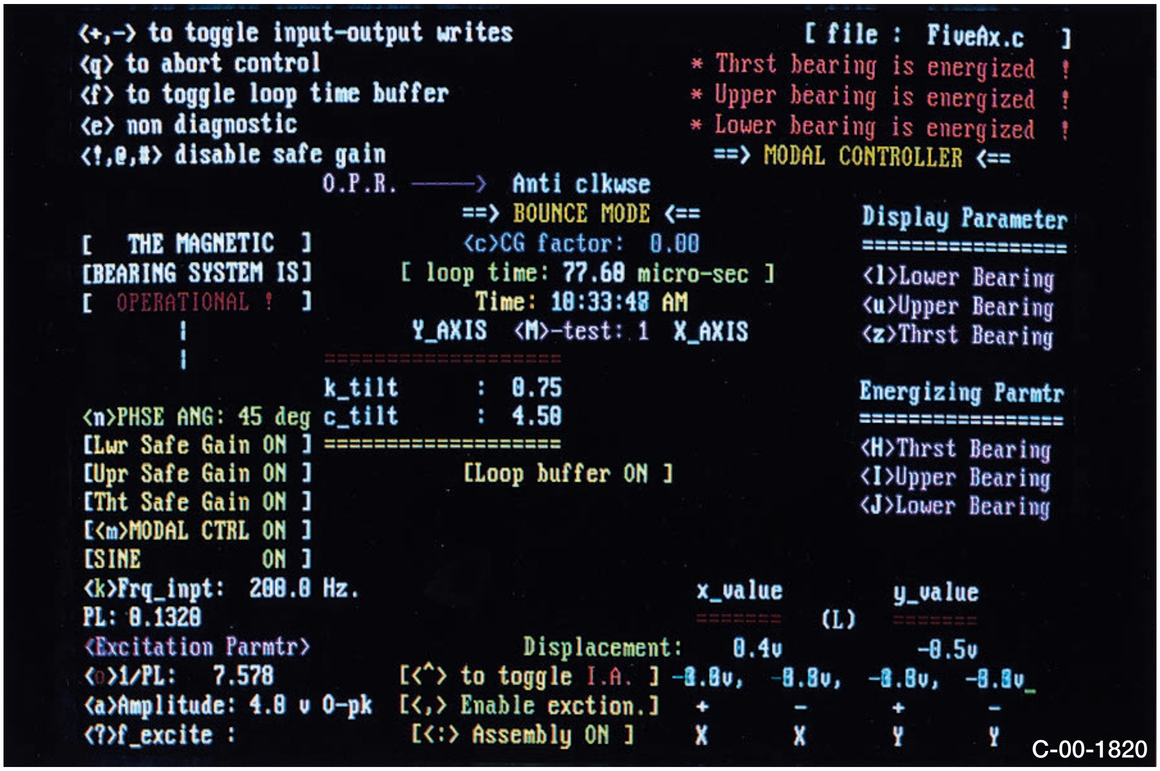
What distinguishes D.A. from I.A. is that in the D.A. mode, the loop time is averaged continuously, producing a relatively smooth and stable signal. On the other hand, in the I.A. mode, the averaged loop time value remains constant between each update, resulting in a minor discontinuity at the instant of the update. The number 15 in the preceeding expression is the maximum number of *DOS-clock-determined loop times* that were averaged. UCL is adjusted upwards by the ***** key or downwards by the **&** key. This adjustment is only possible when the intermittent averaging option is toggled. For I.A., refer to source code lines 2873–2898.

The third option for generating a signal frequency is called the "standard method" (SM). This method produces the most stable signal frequency because the *O*-value (source code lines 3736–3755) is calculated directly. The two previously discussed methods determined the *O*-value by averaging the loop time. The drawback with the standard method is that the signal frequency is obtained by changing the PL in increments of 0.002. This discreteness makes it impossible at times to obtain a desired frequency. In the previous methods, the exact frequency can be specified and the computer then determines the *O*-value. Pressing the **o** key increases the frequency and depressing the **Shift** key while simultaneously pressing the **o** key decreases the frequency. The approximate frequency is displayed under the header "<Excitation Parmtr>." Use a digital oscilloscope for a more accurate measure of the output frequency. Connect the oscilloscope to the signal output connector ("SIGNAL GEN") located on the test rig control panel (fig. 16, lower bearing output panel).

After selecting a desired frequency, increase the signal amplitude by pressing the **a** key or decrease the signal amplitude by depressing the **Shift** key while simultaneously pressing the **a** key. The maximum amplitude available is 5 V, (0 to peak). The next step is to output the signal to the magnetic bearings, which is accomplished by pressing the **,** key. Observe the "[<,> Enable exction.]" display at the bottom of the screen (fig. 14(a)).

16.0 MODAL CONTROL TOGGLE

After correctly setting all the critical parameters discussed in sections 6.0 to 15.0, engage the modal control by toggling the **m** key (see fig. 20 for the corresponding screen display). The transition to modal control is seamless and without any noticeable changes in the levitation of the rotor. Modal control may also be toggled in the nondiagnostic display mode. Make any necessary fine adjustments to the "<c>CG factor:."



```
<+,-> to toggle input-output writes                    [ file :  FiveAx.c   ]
<q> to abort control                               * Thrst bearing is energized  !
<f> to toggle loop time buffer                     * Upper bearing is energized  !
<e> non diagnostic                                 * Lower bearing is energized  !
<!,@,#> disable safe gain                            ==> MODAL CONTROLLER <==
                  O.P.R. ———> Anti clkwse
                             ==> BOUNCE MODE <==               Display Parameter
[   THE MAGNETIC   ]          <c>CG factor:  0.00              =================
[BEARING SYSTEM IS]      [ loop time: 77.60 micro-sec ]        <l>Lower Bearing
[  OPERATIONAL !   ]            Time: 10:33:40 AM              <u>Upper Bearing
        !                  Y_AXIS  <M>-test: 1  X_AXIS         <z>Thrst Bearing
        !         ====================
                  k_tilt       :  0.75                         Energizing Parmtr
<n>PHSE ANG: 45 deg c_tilt     :  4.50                         =================
[Lwr Safe Gain ON ] ====================                       <H>Thrst Bearing
[Upr Safe Gain ON ]            [Loop buffer ON ]               <I>Upper Bearing
[Tht Safe Gain ON ]                                            <J>Lower Bearing
[<m>MODAL CTRL ON ]
[SINE         ON ]
<k>Frq_inpt:  200.0 Hz.                           x_value          y_value
PL: 0.1320                                        =======    (L)   =======
<Excitation Parmtr>                 Displacement:    0.4v            -0.5v
<o>1/PL:    7.578          [<^> to toggle I.A. ] -0.0v,  -0.0v,  -0.0v,  -0.0v_
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]   +       -       +       -
<?>f_excite :               [<:> Assembly ON ]     X       X       Y       Y
```

C-00-1820

Figure 20.—Modal control display screen.

17.0 EXTERNALLY GENERATED EXCITATION SIGNAL TOGGLE

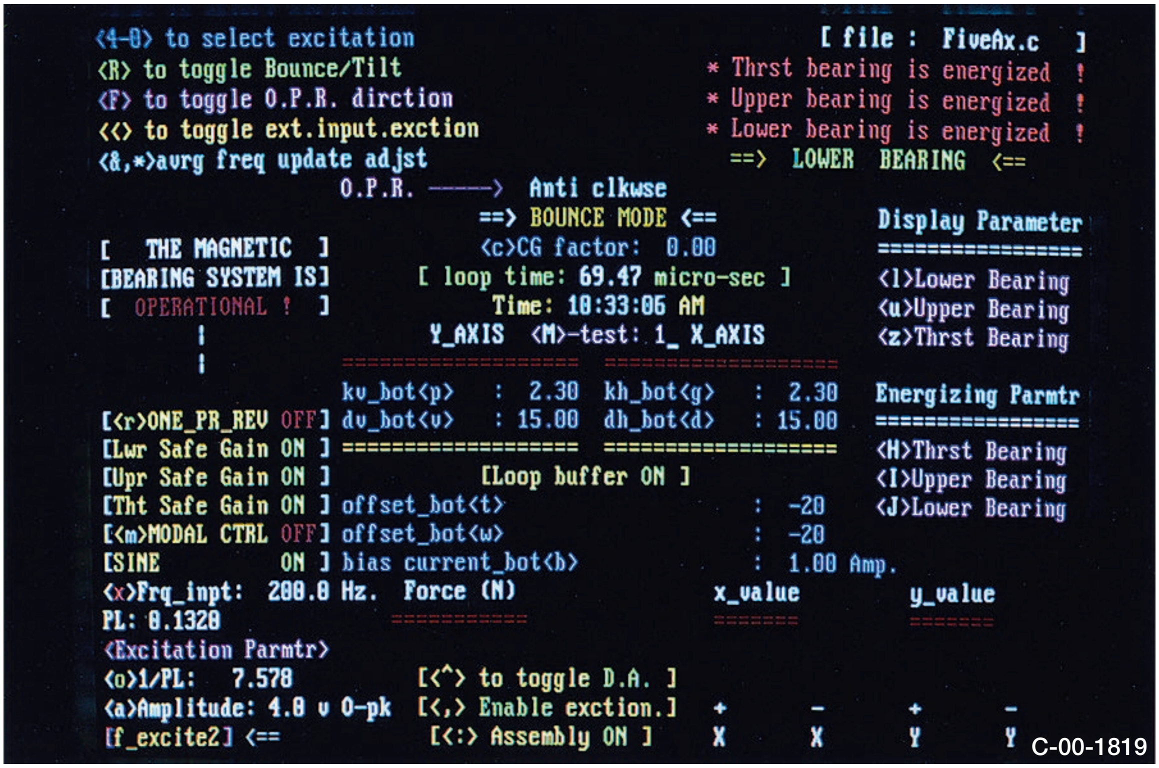
To switch to an external signal source such as a signal generator, press the **<** key. The label “f_excite2” appears at the bottom left of the screen (fig. 21), thus confirming the signal source status. The external signal source should be connected to channel 3 on the Datel input board 2 at address 0x366.

18.0 INTERNALLY GENERATED EXCITATION SIGNAL TOGGLE

Press the **?** key to toggle the screen display of the outputs from a selected signal function (fig. 22). Note the display (which is in digital counts as the code cycles through 0 to 500 steps) at the right of the “<?>f_excite:” label, and the current *cumulative* number of period lengths PL, which is displayed at the right of header “<Excitation Parmtr>.” This option should be used only for code diagnosis because the code is slowed 60 ms to make it possible to observe the signal output. The code may respond sluggishly to key commands during this mode of operation.

19.0 SIGNAL EXPORTATION TOGGLE

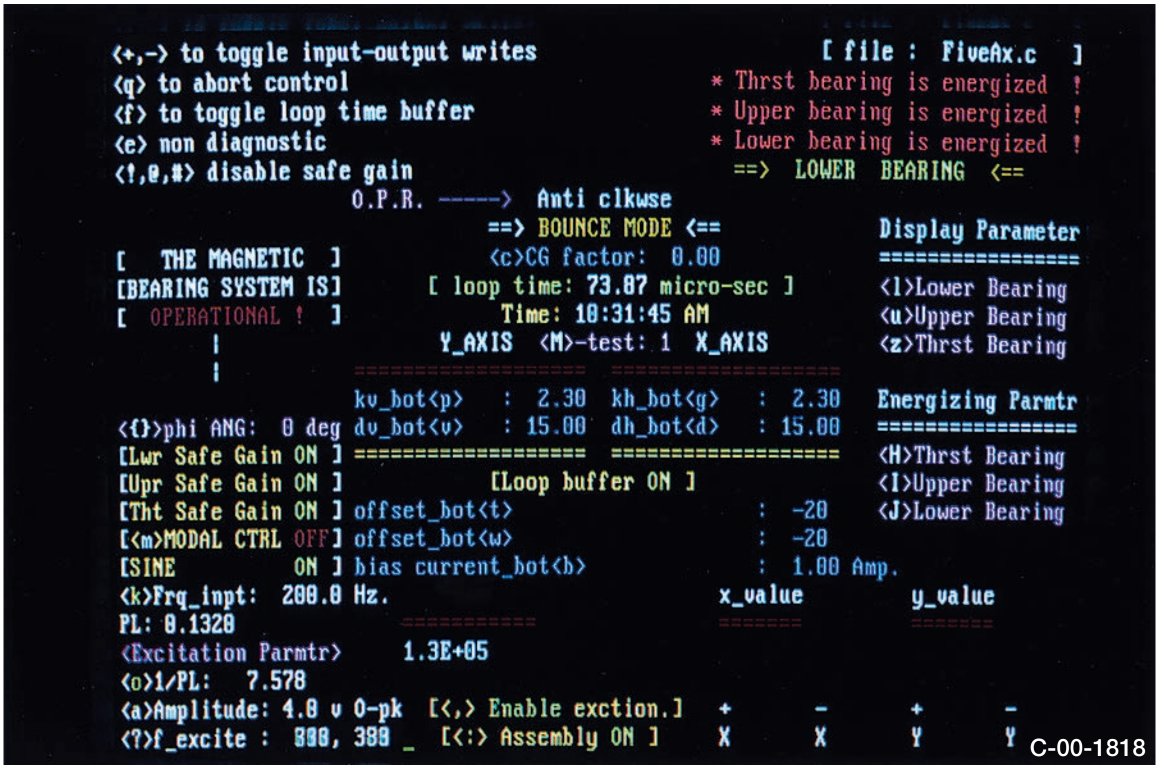
The excitation signals, whether generated in the code or imported from an external signal generator, may be exported for display on an oscilloscope. To toggle this option, depress the **Shift** key while simultaneously pressing the **m** key. In figure 12, the **0** displayed at the “<M>-test:” label changes to **1** to indicate an “on” status (fig. 14(a)). A **0** represents an “off” status. This signal can be obtained from either channel 0 on the Metrabyte board at address 0x330 or more conveniently from the “bnc” connector labeled “SIGNAL GEN,” which is located on the lower bearing output panel in figure 16.



```
<4-0> to select excitation                                [ file :  FiveAx.c   ]
<R> to toggle Bounce/Tilt                           * Thrst bearing is energized  !
<F> to toggle O.P.R. dirction                       * Upper bearing is energized  !
<<> to toggle ext.input.exction                     * Lower bearing is energized  !
<&,*>avrg freq update adjst                           ==>  LOWER  BEARING  <==
                  O.P.R. -----> Anti clkwse
                            ==> BOUNCE MODE <==                Display Parameter
[   THE MAGNETIC  ]          <c>CG factor:  0.00               =================
[BEARING SYSTEM IS]      [ loop time: 69.47 micro-sec ]        <l>Lower Bearing
[  OPERATIONAL !  ]             Time: 10:33:06 AM              <u>Upper Bearing
        |                   Y_AXIS  <M>-test: 1_ X_AXIS        <z>Thrst Bearing
        |         ===================  ===================
                  kv_bot<p>    :  2.30  kh_bot<g>    :  2.30   Energizing Parmtr
[<r>ONE_PR_REV OFF] dv_bot<v>  : 15.00  dh_bot<d>    : 15.00   =================
[Lwr Safe Gain ON ] ===================  ===================   <H>Thrst Bearing
[Upr Safe Gain ON ]            [Loop buffer ON ]               <I>Upper Bearing
[Tht Safe Gain ON ] offset_bot<t>                    :  -20    <J>Lower Bearing
[<m>MODAL CTRL OFF] offset_bot<w>                    :  -20
[SINE          ON ] bias current_bot<b>              :  1.00 Amp.
<x>Frq_inpt:  200.0 Hz.  Force (N)            x_value          y_value
PL: 0.1320                ===========         =======          =======
<Excitation Parmtr>
<o>1/PL:    7.578      [<^> to toggle D.A. ]
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]  +       -        +       -
[f_excite2] <==           [<:> Assembly ON ]     X       X        Y       Y
```

C-00-1819

Figure 21.—Lower bearing display screen showing selection of external signal source ([f_excite2]).



```
<+,-> to toggle input-output writes                       [ file :  FiveAx.c   ]
<q> to abort control                                * Thrst bearing is energized  !
<f> to toggle loop time buffer                      * Upper bearing is energized  !
<e> non diagnostic                                  * Lower bearing is energized  !
<!,@,#> disable safe gain                             ==>  LOWER  BEARING  <==
                  O.P.R. -----> Anti clkwse
                            ==> BOUNCE MODE <==                Display Parameter
[   THE MAGNETIC  ]          <c>CG factor:  0.00               =================
[BEARING SYSTEM IS]      [ loop time: 73.87 micro-sec ]        <l>Lower Bearing
[  OPERATIONAL !  ]             Time: 10:31:45 AM              <u>Upper Bearing
        |                   Y_AXIS  <M>-test: 1  X_AXIS        <z>Thrst Bearing
        |         ===================  ===================
                  kv_bot<p>    :  2.30  kh_bot<g>    :  2.30   Energizing Parmtr
<{}>phi ANG:  0 deg dv_bot<v>  : 15.00  dh_bot<d>    : 15.00   =================
[Lwr Safe Gain ON ] ===================  ===================   <H>Thrst Bearing
[Upr Safe Gain ON ]            [Loop buffer ON ]               <I>Upper Bearing
[Tht Safe Gain ON ] offset_bot<t>                    :  -20    <J>Lower Bearing
[<m>MODAL CTRL OFF] offset_bot<w>                    :  -20
[SINE          ON ] bias current_bot<b>              :  1.00 Amp.
<k>Frq_inpt:  200.0 Hz.                       x_value          y_value
PL: 0.1320                ===========         =======          =======
<Excitation Parmtr>     1.3E+05
<o>1/PL:    7.578
<a>Amplitude: 4.0 v 0-pk  [<,> Enable exction.]  +       -        +       -
<?>f_excite :  000, 300 _  [<:> Assembly ON ]    X       X        Y       Y
```

C-00-1818

Figure 22.—Lower bearing display screen of a selected internal signal function (note outputs 200.0 Hz, 1.3E+05, 7.578).

20.0 SAFE GAIN TOGGLE

Extreme adjustments to the stiffness and/or damping values (see sec. 9.0) may result in the rotor experiencing unstable levitation. Hence, each bearing control block has a safety logic mechanism known as “safe gain” (source code lines 1493–1496; 1844–1847; and 2644–2647). The safe gain logic checks to see if the input value from the proximeter probes exceeds a predetermined upper limit. If this value is exceeded, the stiffness/damping parameters are instantly restored to values that have previously been shown to permit stable levitation. The safe gain parameters should be kept on at all times (fig. 12). Depressing the **Shift** key while simultaneously pressing the **1**, **2**, and **3** keys will turn off the safe gain parameter of each bearing.

21.0 LOOP TIME AND CURRENT TIME DISPLAY

The code cycles through 75 000 loops, after which it does a *current time* (as per the DOS clock) and a loop time update (source code lines 2767–2807; 2815–2820; and 2844–2845). The loop time is the time the code takes to complete one control loop cycle (fig. 12).

22.0 DISPLAY OF ROTOR DISPLACEMENT

Simultaneously press the **Shift** and **+** keys to display (under the header “<u>Force (N)</u>”) the value of the control force command on the rotor along with its instantaneous displacement values (fig. 14(a)). Press the **–** key to turn off the display. These keys also activate and deactivate the displacement display while the code is running in modal control mode. A blinking yellow **w** (fig. 14(b)) will appear in the displacement field if a bearing writeout is unintentionally left activated while the user is viewing the parameter of another bearing. The code may respond sluggishly to key commands during this mode of operation.

23.0 NONDIAGNOSTIC MODE DISPLAY

The nondiagnostic display (fig. 13) is a minimal display mode that may be toggled after adjusting all the critical parameters. When this display is selected, only the nondiagnostic parameter keys are active, except for the safe gain keys. The parameters that are not displayed will be inoperative until the diagnostic mode is again toggled. The “[<r>ONE_PR_REV],” “MODAL CNTRL,” “< >EXCITATION,” and “<,> Enable exction.” parameters are all automatically deactivated but may be reactivated if needed.

APPENDIX A

GRAPHICAL AND MATHEMATICAL REPRESENTATIONS OF EXCITATION SIGNALS

The following are the graphical and mathematical representations of the excitation signals that were implemented in the code. The amplitude *A* was replaced by the variable t04 (source code lines 750, 757, 764, 771, 796, 825, 853, 881, 910, 938, and 967), and its value ranges from 0.0 to 1024.0 digital counts (i.e., 0 to 5 V in 0.1-V increments). "*O*" is 1/PL. By changing the value of PL between 0.002 and 1.0, a wide range of frequencies may be obtained. Each loop of the code increments the *x*-value by 0.002 until it exceeds the upper limit 1.75×10^{308}, at which point *x* is reinitialized to zero.

Sine:

$$f(x) = A \times \sin(2.0 \times \pi \times O \times x) \tag{14}$$

Sine squared:

$$f(x) = A \times \sin(2.0 \times \pi \times O \times x) \times \sin(2.0 \times \pi \times O \times x) \tag{15}$$

Cosine:

$$f(x) = A \times \cos(2.0 \times \pi \times O \times x) \tag{16}$$

Cosine squared:

$$f(x) = A \times \cos(\pi \times O \times x) \times \cos(\pi \times O \times x) \tag{17}$$

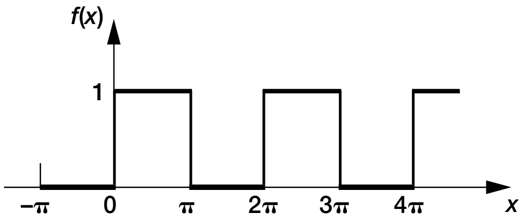
Random:

$$f(x) = A \times \sin(2.0 \times \pi \times \text{f\_excite3}) \times \left[\sin(2.0 \times \pi \times O \times x) + \sin(2.0 \times \pi \times \text{f\_excite4} \times O)\right] \tag{18}$$

where f_excite3 and f_excite4 are random number variables (source code lines 785 and 789). The second sine term coupled with the third produces a curve with a random beat frequency, the amplitude of which is further modulated by the first sine term.

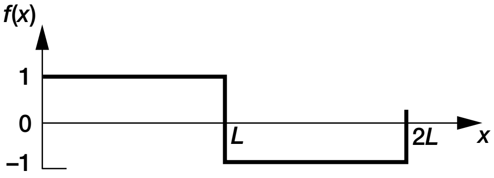
Squared pulse train:

$$f(x) = A \times \left(1 + \frac{4}{\pi}\left\{\left[\sum_{k=0}^{40}\left(\frac{1}{2k+1}\right) \times \sin\left[2.0 \times (2k+1) \times \pi \times O \times x\right]\right]\right\}\right) \tag{19}$$



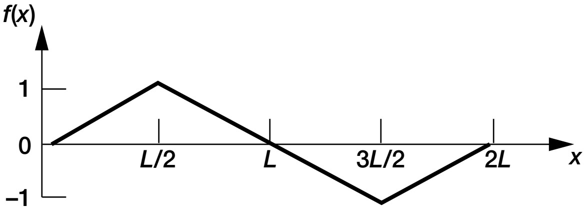
Square wave:

$$f(k) = A \times \frac{4}{\pi}\left\{\sum_{k=0}^{40}\left(\frac{1}{2k+1}\right) \times \sin\left[2.0 \times (2k+1) \times \pi \times O \times x\right]\right\} \tag{20}$$



Triangular wave:

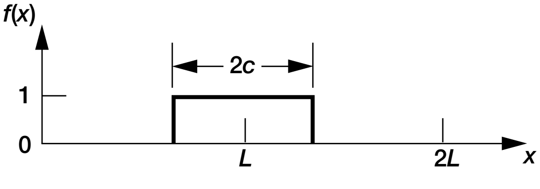
$$f(x) = A \times \left(\frac{8}{\pi^2}\left\{\sum_{k=0}^{40}\left[\frac{(-1)^k}{(2k+1)^2}\right] \times \sin\left[2.0 \times (2k+1) \times \pi \times O \times x\right]\right\}\right) \tag{21}$$



Single square pulse:

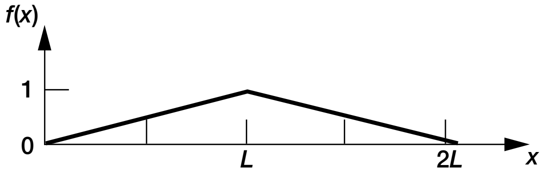
$$f(x) = A \times \left[O \times C + \frac{2}{\pi}\left(\sum_{k1=1}^{40}\left\{\left[\frac{(-1)^{k1}}{k1}\right] \times \sin(k1 \times \pi \times O \times C) \times \cos(2.0 \times k1 \times \pi \times O \times x)\right\}\right)\right] \tag{22}$$

where C is the pulse width PW.



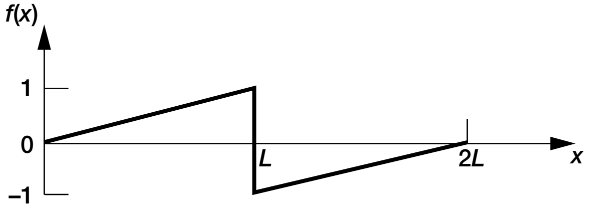
Single triangular pulse:

$$f(x) = A \times \left(0.5 - \frac{4.0}{\pi^2} \left\{ \sum_{k=0}^{40} \left[\frac{1}{(2k+1)^2} \right] \times \cos\left[2.0 \times (2k+1) \times \pi \times O \times x \right] \right\} \right) \tag{23}$$



Saw tooth:

$$f(x) = A \times \frac{2}{\pi} \left\{ \sum_{k=0}^{40} \left[\frac{(-1)^{k1+1}}{k1} \right] \times \sin(2.0 \times k1 \times \pi \times O \times x) \right\} \tag{24}$$



APPENDIX B

SOURCE CODE

This program was designed and written by Carlos R. Morrison (9/28/2000). It incorporates three control blocks for levitating and controlling three magnetic bearings: lower, upper, and thrust. Additionally, the code allows one to toggle any 1 of 11 excitation signals. Each signal is used in conjunction with the “ONE_PR_REV” (one-per-revolution) logic block that was originally conceived by Dr. Gerald Brown. The code also has an enhanced graphical user interface for ease of use.

FIVEAXW.C

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<math.h>
#include<time.h>
#include<stdlib.h>

/*------------------------VARIABLE DECLARATION----------------------------*/

int board,lchan1,lchan2,lchan3,pchan1,pchan2,pchan3,erstat,xbot,ybot,xtop,
    ytop,zth,zth1,zth2,x_bot_old1,x_bot_old2,x_bot_old3,x_bot_old4,
    x_bot_old5,y_old_bot,y_old_top,y_old_th,y_bot_old1,y_bot_old2,
    y_bot_old3,y_bot_old4,y_bot_old5,x_top_old1,x_top_old2,x_top_old3,
    x_top_old4,x_top_old5,y_top_old1,y_top_old2,y_top_old3,y_top_old4,
    y_top_old5,z_th_old1,z_th_old2,z_th_old3,z_th_old4,z_th_old5,Base1,
    Base2,out_chan1_0,out_chan1_1,out_chan1_2,out_chan1_3,out_chan1_4,
    out_chan1_5,out_chan2_0,out_chan2_1,out_chan2_2,out_chan2_3,out_chan2_4,
    out_chan2_5,i_bot,i_top,i_th,j,tBias_bot,tBias_top,tBias_th,wBias_bot,
    wBias_top,wBias_th,nw_bot,nw_top,nw_th,fig,out_min,out_max,n,jjj,
    bias_current_bot,bias_current_top,bias_current_th,nmax,lmax,l,
    PD_tBias_bot,PD_tBias_top,PD_tBias_th,PD_wbias_bot,PD_wbias_top,
    PD_wbias_th,valuenoise,FIFO1,FIFO2,zero,one,two,hh,g,vv=15,k,k1,m,m1,
    m2,m3,m4,p,x0,d_max_th,d,v,ROUND,flag1,flag2,flag3,flag4,flag5,
    flag6,flag7,flag8,flag9,flag10,flag12,flag13,flag15,flag11,flag22,
    flag33,flag44,flag16,flag18,flag19,flag20,flag21,flag23,flag24,flag25,
    flag4a,flag4b,flag4c,flag4d,flag_A,flag_B,flag_C,flag_D,flag_E,flag_F,
    flag_G,flag_H,flag_I,flag_J,flag_K,flag_L,flag_M,flag_N,flag_AA,flag_BB,
    flag_CC,flag_DD,flag_EE,flag_FF,flag_GG,flag_HH,flag_II,flag_JJ,flagJJ,
    thp,flag_jj,flagKK,flagLL,flagMM,flagNN,out_bot,out_top,out_th,diag,t48,
    round2,cir,cir2,sg1,sg2,sg3,excite,f_excite,f_excite2,num,n_x,SSS,th,
    i_rev,one_per_rev,trigger=21,rise,N_ticks,j_rev,X_P_O_B,X_N_O_B,Y_P_O_B,
    Y_N_O_B,X_P_O_T,X_N_O_T,Y_P_O_T,Y_N_O_T,TC,test_signal,switch1,excite_cos,
    excite_sin,maxv,set=1,rr=0,qq=0,i;

double I_lim,loop_time,last_time,micro,junk,ibias_bot,ibias_top,ibias_th,
       dh_bot,kh_top,dh_top,dh_th,dv_bot,kh_bot,kh_th,kv_bot,kv_top,kv_th,
       dv_top,dv_th,x_force_bot,y_force_bot,x_force_top,y_force_top,
       z_force_th,xbotderiv,ybotderiv,xtopderiv,ytopderiv,zthderiv,
       x_pos_output_bot,x_neg_output_bot,x_pos_output_top,x_neg_output_top,
       up_output_th,down_output_th,y_pos_output_bot,y_pos_output_top,
       y_neg_output_bot,y_neg_output_top,z,xbotsum,ybotsum,xtopsum,zthsum,
       ytopsum,igainbot,igaintop,igainth,igainmod,safe,zsafe,x,O,frequency,
       period,PL,ex,f_ex,volt,C,PW,PWW,freq,t04,THETA,f_excite_cos,
       f_excite_sin,PI2_o_Nticks,PI2,phi,i_rev1,pp=0.0,Yav,Xav,xbot_force_tr,
       xtop_force_tr,ybot_force_tr,ytop_force_tr,dotXav,dotYav,oldoldXav,
       oldXav,oldoldYav,oldYav,ThetaX,ThetaY,L,xbot_force_rot,k_tilt,c_tilt,
       dotThetaX,xtop_force_rot,ybot_force_rot,dotThetaY,ytop_force_rot,
       oldoldThetaX,oldoldThetaY,oldThetaY,oldThetaX,xbot_force_modal_pos,
        xbot_force_modal_neg,xtop_force_modal_pos,xtop_force_modal_neg,
        ybot_force_modal_pos,ybot_force_modal_neg,ytop_force_modal_pos,II,JJ,
        ytop_force_modal_neg,F_XB_tr,F_XT_tr,F_YB_tr,F_YT_tr,excitef,
        LIM,OO=0.0,OL=0.0,ii=0.0,LT,L_T,CG,A1=0.0,A2=0.0,A3=0.0,A4=0.0,A5=0.0,
        A6=0.0,A7=0.0,A8=0.0,A9=0.0,A10=0.0,A11=0.0,A12=0.0,A13=0.0,A14=0.0,
        A15=0.0,B1=0.0,B2=0.0,B3=0.0,B4=0.0,B5=0.0,B6=0.0,B7=0.0,B8=0.0,B9=0.0,
        B10=0.0,B11=0.0,B12=0.0,B13=0.0,B14=0.0,B15=0.0,f_excite3,f_excite4,
        xy=0.0,COUNTMAX=15.0,MCG,PCG,cos(double x),sin(double x),ns;

struct time now,tt;

unsigned int ti_min,ti_second,ti_hund;

float round1(float u),randvalue,time1;

char resp,lu,respp,ig;

const int NUMBERS = 1;

int main(void)

/*---------------------------------INITIALIZE -----------------------------------*/

{
   clrscr();

// ******************* Datel Input Board (1) setup *******************
            // Board address: 0x300
   outportb(0x30e, 0x3a);             j = 1; while ( j<5000 ) j++;
   outportb(0x308,    2);             j = 1; while ( j<5000 ) j++;
   outportb(0x308,    0);             j = 1; while ( j<5000 ) j++;

   outportb(0x30e, 0x7a);             j = 1; while ( j<5000 ) j++;
   outportb(0x30a,    1);             j = 1; while ( j<5000 ) j++;
   outportb(0x30a,    0);             j = 1; while ( j<5000 ) j++;

   outportb(0x30e, 0xba);             j = 1; while ( j<5000 ) j++;
   outportb(0x30c,    1);             j = 1; while ( j<5000 ) j++;
   outportb(0x30c,    0);             j = 1; while ( j<5000 ) j++;

   outport (0x302, 0x40);             j = 1; while ( j<5000 ) j++;
   outport (0x306,    1);             j = 1; while ( j<5000 ) j++;
   outport (0x300,  0xe);             j = 1; while ( j<5000 ) j++;

// ******************* Datel Input Board (2) setup *******************
            // Board address: 0x360
   outportb(0x36e, 0x3a);             j = 1; while ( j<5000 ) j++;
   outportb(0x368,    2);             j = 1; while ( j<5000 ) j++;
   outportb(0x368,    0);             j = 1; while ( j<5000 ) j++;

   outportb(0x36e, 0x7a);             j = 1; while ( j<5000 ) j++;
   outportb(0x36a,    1);             j = 1; while ( j<5000 ) j++;
   outportb(0x36a,    0);             j = 1; while ( j<5000 ) j++;

   outportb(0x36e, 0xba);             j = 1; while ( j<5000 ) j++;
   outportb(0x36c,    1);             j = 1; while ( j<5000 ) j++;
   outportb(0x36c,    0);             j = 1; while ( j<5000 ) j++;

   outport (0x362, 0x40);           j = 1; while ( j<5000 ) j++;
   outport (0x366,    1);           j = 1; while ( j<5000 ) j++;
   outport (0x360,  0xe);           j = 1; while ( j<5000 ) j++;

   FIFO1 = 0x306;// Base = 300, FIFO1 = base + 6;
   FIFO2 = 0x366;// Base = 360, FIFO2 = base + 6;

// ****************  Metrabyte Output Board (1) setup ****************

   Base1 = 0x330;// Board address: 0x330 Lower Bearing + Thrust up (Z+)
   out_chan1_0 = Base1 +  0;
   out_chan1_1 = Base1 +  2;
   out_chan1_2 = Base1 +  4;
   out_chan1_3 = Base1 +  6;
   out_chan1_4 = Base1 +  8;
   out_chan1_5 = Base1 + 10;

   t48 = 2048;// 2048 => Ten volts

   outport(out_chan1_0, t48);// Code's signal output
   outport(out_chan1_1, t48);// +X_L
   outport(out_chan1_2, t48);// -X_L
   outport(out_chan1_3, t48);// +Y_L
   outport(out_chan1_4, t48);// -Y_L
   outport(out_chan1_5, t48);// +Z_TH

// ****************  Metrabyte Output Board (2) setup ****************

   Base2 = 0x390;// Board address: 0x390 Upper Bearing + Thrust down (Z-)
// out_chan2_0 = Base2 +  0;
   out_chan2_1 = Base2 +  2;
   out_chan2_2 = Base2 +  4;
   out_chan2_3 = Base2 +  6;
   out_chan2_4 = Base2 +  8;
   out_chan2_5 = Base2 + 10;

// outport(out_chan2_0, t48);
   outport(out_chan2_1, t48);// +X_U
   outport(out_chan2_2, t48);// -X_U
   outport(out_chan2_3, t48);// +Y_U
   outport(out_chan2_4, t48);// -Y_U
   outport(out_chan2_5, t48);// -Z_TH


// ********************** GENERAL VARIABLE INITIALIZATION ******************

   safe  = 32600;
   zsafe = 16300;
   nmax = 500; lmax = 150; l = 0;
   micro = (1000000.0 / nmax / lmax);
   I_lim = 4.0;
   out_min = -round1(2.0 * I_lim * 204.8) + t48;
   out_max =  round1(2.0 * I_lim * 204.8) + t48;
   loop_time = 0.78; hh = 0;
   zero = 0; one = 1; two = 2;
   LIM = 1.75 * pow(10,308);// max # of period lengths (upper limit)
   x0 = 21;/*(0.1)*//*103(.5v)*//*205(1v)*//*1435(7v)*/
   k = 0;
```

```
k1 = 1;
x = 0.0;
f_excite = 0.0;
excite = 0.0;
f_excite_sin = 0.0;
f_excite_cos = 0.0;
JJ = 1.0;
II = 1.0;
ex = 0.0;
O = 1.0;
frequency = 0.0;
PWW = 0.0;
PW = 0.0;
i_rev = 0;
j_rev = 0;
THETA = 0.0;
th = 0;
PI2 = 2 * M_PI;
phi = 0.0;
L = 1.0;
TC = 9;
test_signal = 0;
t04 = 0.0;
freq = 0.0;
PL = 1.0;
CG = 0.0;
MCG = 0.5 - CG;
PCG = 0.5 + CG;
cir = 23;
cir2 = 55;
flag5 = 0;
flag6 = 0;
flag7 = 0;
flag8 = 0;
flag9 = 0;
flag12 = 0;
flag13 = 0;
flag10 = 0;// Disable modal block
flag15 = 1;
flag16 = 1;// Assembly condition (on)
flag18 = 0;
flag19 = 0;
flag20 = 1;
flag21 = 0;// Disable excitation block
flag23 = 1;
flag24 = 1;// Enable loop_time averaging
flag25 = 1;// Toggle loop_time averaging
flag_A = 1;// Assembly toggle set to off
flag_B = 1;
flag_C = 1;
flag_D = 1;
flag_E = 1;
flag_F = 1;
flag_G = 1;
flag_H = 1;
flag_I = 1;
flag_J = 1;
flag_K = 0;
```

```
    flag_L = 0;
    flag_M = 1;
    flag_N = 1;
    flag_AA = 0;
    flag_BB = 1;
    flag_GG = 1;
    flag_HH = 0;
    flag_II = 0;
    flagJJ = 1;
    flag_JJ = 1;
    flag_jj = 1;
    flagKK = 1;
    flagLL = 1;
    flagMM = 0;
    flagNN = 1;
    switch1 = 0;

// ****************** BOTTOM BEARING VARIABLE INITIALIZATION **************

    kv_bot = 2.3;
    kh_bot = kv_bot;
    dh_bot = 15.0;
    dv_bot = dh_bot;
    ibias_bot = 1.0;// Amperes
    bias_current_bot = round1(ibias_bot * 2.0 * 204.8);// two Volts => one Amp.

    // Remember amplifier gain is 0.5A/V

    PD_tBias_bot = -20; PD_wbias_bot = -20;// Initial differential biases
    tBias_bot = PD_tBias_bot; wBias_bot = PD_wbias_bot;
    nw_bot = 0;// For writeout, set nw_bot = 1
    sg1 = 1;// Lower Bearing safe gain set

// **************** TOP BEARING VARIABLE INITIALIZATION ******************

    kv_top = 2.3;
    kh_top = kv_top;
    dh_top = 15.0;
    dv_top = dh_top;
    ibias_top = 1.0;// Amperes
    bias_current_top = round1(ibias_top * 2.0 * 204.8);// Two Volts => one Amp

    // Remember amplifier gain is 0.5A/V

    PD_tBias_top = -20; PD_wbias_top = -20;// Initial differential biases
    tBias_top = PD_tBias_top; wBias_top = PD_wbias_top;
    nw_top = 0;
    sg2 = 1;// Upper Bearing safe gain set

// **************** THRUST BEARING VARIABLE INITIALIZATION ****************

    kv_th = 2.3;
    dv_th = 15.0;
    ibias_th = 1.5;// Amperes multiplication factor
    igainth    = 0.0002;
    bias_current_th = round1(ibias_th * 2.0 * 204.8);// Two Volts => one Amp

    // Remember amplifier gain is 0.5A/V
```

```

   PD_tBias_th = -20;                    // Initial differential biases
   tBias_th = PD_tBias_th;
   nw_th = 0;
   sg3 = 1;// Thrust Bearing safe gain set

// **************************************************************************

    flag1  = 0;
    flag2  = 0;
    flag3  = 0;
    flag4  = 1;

    flag4a = 0;
    flag4b = 0;
    flag4c = 0;
    flag4d = 1;

    flag11 = 1;// Enable lower bearing write out block
    flag22 = 0;// Disable upper bearing write out block
    flag33 = 0;// Disable thrust bearing write out block
    flag44 = 0;// Enable D.A/I.A. display

// ------------------------------SHOW MENU---------------------------------

    clrscr();

    gotoxy(45,6);textcolor(4);
    gotoxy(59,1);textcolor(15);
    cprintf("[ file :  FiveAx.c   ]");
    gotoxy(29,13);textcolor(15);
    cprintf("**********************");
    gotoxy(29,14);textcolor(15);
    cprintf("*                    *");
    gotoxy(29,15);textcolor(15);
    cprintf("*                    *");
    gotoxy(29,16);textcolor(15);
    cprintf("**********************");
    gotoxy(35,14);textcolor(14);
    cprintf("FIVE- AXIS");
    gotoxy(32,15);textcolor(14);
    cprintf("BEARING FACILITY");

G:  gotoxy(31,5);textcolor(10);
    cprintf("DIAGNOSTIC (y/n)?:");
    respp = getch();
    gotoxy(31,5);
    printf("                   ");// Erase "DIAGNOSTIC (y/n)?:"
    if (respp == 'y' || resp == 'Y')
    {
      SSS = 1;
      diag  = 1;

      clrscr();

      goto H;
    }

```

```
     else

     if (respp == 'n' || resp == 'N')
     {
       clrscr();

       SSS = 0;
       gotoxy(1,1);textcolor(15);
       cprintf("<x/k> to adjust frequency");
       gotoxy(1,2);textcolor(15);
       cprintf("<q> to abort control");
       gotoxy(1,3);textcolor(15);
       cprintf("<m> to toggle modal cntrl");
       gotoxy(1,4);textcolor(15);
       cprintf("<?> to toggle f_excite");
       gotoxy(1,5);textcolor(15);
       cprintf("<4-0> to select excitation");
       gotoxy(59,1);textcolor(15);
       cprintf("[ file :  FiveAx.c   ]");
       gotoxy(31,2);textcolor(11);
       cprintf("DIAGNOSTIC TOGGLE<E>");
       gotoxy(1,22);textcolor(13);
       cprintf("<Excitation Parmtr>");
       gotoxy(1,14);textcolor(10);
       cprintf("< >PHSE ANG:%3u deg",th);
       gotoxy(2,14);textcolor(15);
       cprintf("n");
       gotoxy(1,23);textcolor(15);
       cprintf("<o>freq:%8.2f  Hz",frequency);
       gotoxy(1,20);textcolor(15);
       cprintf("<x>Frq_inpt:%7.1f Hz.",freq);
       gotoxy(1,25);textcolor(15);
       cprintf("<s>to adjust Pulse Width");
       gotoxy(1,24);textcolor(15);
       cprintf("<a>Amplitude:%4.1f v O-pk",volt);
       gotoxy(27,23);textcolor(14);
       cprintf("[<^> to toggle D.A. ]");
       gotoxy(27,24);textcolor(14);
       cprintf("[<,> Enable exction.]");
       gotoxy(28,25);textcolor(14);
       cprintf("[<:> Assembly    ]");
       gotoxy(42,25);textcolor(10);
       cprintf("ON");

       nw_bot = 0;
       nw_top = 0;
       nw_th  = 0;

       diag  = 0;
       flag1 = 1;// Lower bearing block activated
       flag2 = 1;// Upper bearing block activated
       flag3 = 1;// Thrust bearing block activated

       goto H;
     }
       goto G;

H:  if (diag == 1)
```

```
        {
          gotoxy(59,1);textcolor(15);
          cprintf("[ file :  FiveAx.c   ]");
          gotoxy(1,1);textcolor(15);
          cprintf("<+,-> to toggle input-output writes");
          gotoxy(1,2);textcolor(15);
          cprintf("<q> to abort control");
          gotoxy(1,3);textcolor(15);
          cprintf("<f> to toggle loop time buffer");
          gotoxy(1,4);textcolor(15);
          cprintf("<e> non diagnostic");
          gotoxy(1,5);textcolor(15);
          cprintf("<!,@,#> disable safe gain");
          gotoxy(19,11);textcolor(15);
          cprintf("         Y_AXIS               X_AXIS");
          gotoxy(36,11);textcolor(13);
          cprintf("< >-test: %1u",test_signal);
          gotoxy(37,11);textcolor(15);
          cprintf("M");
          gotoxy(21,12);textcolor(4);
          cprintf("==================  ===================");
          gotoxy(21,15);textcolor(14);
          cprintf("==================  ===================");
          gotoxy(52,5);textcolor(14+128);
          cprintf("==>                  <==");
          gotoxy(57,5);textcolor(10);
          cprintf("LOWER  BEARING");
          gotoxy(31,8);textcolor(9);
          cprintf(" <c>CG factor: %5.2f",CG);
          gotoxy(32,16);textcolor(14);
          cprintf("[loop buffer    ]");
          gotoxy(45,16);textcolor(10);
          cprintf("ON ");
          gotoxy(21,13);textcolor(9);
          cprintf("kv_bot<p>   :%6.2f", kv_bot);
          gotoxy(42,13);textcolor(9);
          cprintf("kh_bot<g>   :%6.2f", kh_bot);
          gotoxy(21,14);textcolor(9);
          cprintf("dv_bot<v>   :%6.2f", dv_bot);
          gotoxy(42,14);textcolor(9);
          cprintf("dh_bot<d>   :%6.2f", dh_bot);
          gotoxy(21,17);textcolor(9);
          cprintf("offset_bot<t>                      :");
          gotoxy(55,17);textcolor(9);
          cprintf("%5d", tBias_bot);
          gotoxy(21,18);textcolor(9);
          cprintf("offset_bot<w>                      :");
          gotoxy(55,18);textcolor(9);
          cprintf("%5d", wBias_bot);
          gotoxy(21,19);textcolor(9);
          cprintf("offset current_bot<b>              :");
          gotoxy(55,19);textcolor(9);
          cprintf("%6.2f Amp.", ibias_bot);
          gotoxy(51,20);textcolor(15);
          cprintf("x_value         y_value");
          gotoxy(51,21);textcolor(4);
          cprintf("=======         =======");
          gotoxy(49,24);textcolor(15);
        cprintf("  +      -      +      -  ");
        gotoxy(49,25);textcolor(15);
        cprintf("  X      X      Y      Y  ");
        gotoxy(64, 7);textcolor(11);cprintf("Display Parameter");
        gotoxy(64, 8);textcolor(15);cprintf("=================");
        gotoxy(64, 9);textcolor(13);cprintf("< >Lower Bearing");
        gotoxy(65, 9);textcolor(15);cprintf("l");
        gotoxy(64,10);textcolor(13);cprintf("< >Upper Bearing");
        gotoxy(65,10);textcolor(15);cprintf("u");
        gotoxy(64,11);textcolor(13);cprintf("< >Thrst Bearing");
        gotoxy(65,11);textcolor(15);cprintf("z");
        gotoxy(64,13);textcolor(11);cprintf("Energizing Parmtr");
        gotoxy(64,14);textcolor(15);cprintf("=================");
        gotoxy(64,15);textcolor(13);cprintf("< >Thrst Bearing");
        gotoxy(65,15);textcolor(15+128);cprintf("H");
        gotoxy(64,16);textcolor(13);cprintf("< >Upper Bearing");
        gotoxy(65,16);textcolor(15+128);cprintf("I");
        gotoxy(64,17);textcolor(13);cprintf("< >Lower Bearing");
        gotoxy(65,17);textcolor(15+128);cprintf("J");
        gotoxy(26,20);textcolor(15);
        cprintf("Force (N)");
        gotoxy(25,21);textcolor( 4);
        cprintf("===========");
        gotoxy(1,20);textcolor(15);
        cprintf("<x>Frq_inpt:%7.1f Hz.");
        gotoxy(1,25);textcolor(15);
        cprintf("<s>to adjust Pulse Width");
        gotoxy(1,15);textcolor(15);
        cprintf("[                 ]");
        gotoxy(1,14);textcolor(10);
        cprintf("< >PHSE ANG:%3u deg",th);
        gotoxy(2,14);textcolor(15);
        cprintf("n");
        gotoxy(2,15);textcolor(14);
        cprintf("Lwr Safe Gain   ");
        gotoxy(16,15);textcolor(10);
        cprintf("ON ");
        gotoxy(1,16);textcolor(15);
        cprintf("[                 ]");
        gotoxy(2,16);textcolor(14);
        cprintf("Upr Safe Gain   ");
        gotoxy(16,16);textcolor(10);
        cprintf("ON ");
        gotoxy(1,17);textcolor(15);
        cprintf("[                 ]");
        gotoxy(2,17);textcolor(14);
        cprintf("Tht Safe Gain   ");
        gotoxy(16,17);textcolor(10);
        cprintf("ON ");
        gotoxy(1,18);textcolor(15);
        cprintf("[                 ]");
        gotoxy(2,18);textcolor(14);
        cprintf("< >MODAL CTRL   ");
        gotoxy(3,18);textcolor(15+128);
        cprintf("m");
        gotoxy(16,18);textcolor(12+128);
        cprintf("OFF");
        gotoxy(1,19);textcolor(15);
```

```
      cprintf("[                 ]");
      gotoxy(2,19);textcolor(14);
      cprintf("< >EXCITATION    ");
      gotoxy(16,19);textcolor(12+128);
      cprintf("OFF");
      gotoxy(1,22);textcolor(13);
      cprintf("<Excitation Parmtr>");
      gotoxy(1,23);textcolor(15);
      cprintf("<o>freq:%8.2f Hz",frequency);
      gotoxy(1,24);textcolor(15);
      cprintf("<a>Amplitude:%4.1f v O-pk",volt);
      gotoxy(27,23);textcolor(14);
      cprintf("[<^> to toggle D.A. ]");
      gotoxy(27,24);textcolor(14);
      cprintf("[<,> Enable exction.]");
      gotoxy(28,25);textcolor(14);
      cprintf("[<:> Assembly    ]");
      gotoxy(42,25);textcolor(10);
      cprintf("ON");
    }// End if (diag == 1)
      gotoxy(27, 9);textcolor(10);
      cprintf("[ loop time:       micro-sec ]");
      gotoxy(1, 8);textcolor(15);cprintf("[   THE MAGNETIC  ]");
      gotoxy(1, 9);textcolor(15);cprintf("[BEARING SYSTEM IS]");
      gotoxy(1,10);textcolor(15);cprintf("[                  ]");
      gotoxy(9,11);textcolor(15);cprintf("|");
      gotoxy(9,12);textcolor(15);cprintf("|");

      if(flag4 == 0)
      {
       gotoxy(4,10);textcolor(12+128);
       cprintf("OPERATIONAL !  ");
      }
      else
      {
        gotoxy(4,10);textcolor(12+128);
        cprintf("OPERATIONAL !\a  ");
      }

      if(diag == 1)
      {
        flag_CC = 1;
        flag1  = 0;
        flag4a = 1;// Turn on Lower Bearing buffer
        gotoxy(48,4);textcolor(14+128);
        cprintf("  * Lower bearing not energized !");

        flag_DD = 1;
        flag2  = 0;
        flag4b = 1;// Turn on Upper Bearing buffer
        gotoxy(48,3);textcolor(14+128);
        cprintf("  * Upper bearing not energized !");

        flag_EE = 1;
        flag3  = 0;
        flag4c = 1;// Turn on Thrust Bearing buffer
        gotoxy(48,2);textcolor(14+128);
        cprintf("  * Thrst bearing not energized !");
        }
        else

        if (diag == 0)
        {
          gotoxy(31,8);textcolor(9);
          cprintf(" <c>CG factor: %5.2f",CG);
          gotoxy(26,13);textcolor(14);
          cprintf("==>                         <==");
          gotoxy(30,13);textcolor(12+128);
          cprintf("THRST BEARING ENERGIZED");
          gotoxy(26,14);textcolor(14);
          cprintf("==>                         <==");
          gotoxy(30,14);textcolor(12+128);
          cprintf("UPPER BEARING ENERGIZED");
          gotoxy(26,15);textcolor(14);
          cprintf("==>                         <==");
          gotoxy(30,15);textcolor(12+128);
          cprintf("LOWER BEARING ENERGIZED");
        }
          gotoxy(1,15);textcolor(15);
          cprintf("[                 ]");
          gotoxy(2,15);textcolor(14);
          cprintf("Lwr Safe Gain   ");
          gotoxy(16,15);textcolor(10);
          cprintf("ON ");
          gotoxy(1,16);textcolor(15);
          cprintf("[                 ]");
          gotoxy(2,16);textcolor(14);
          cprintf("Upr Safe Gain   ");
          gotoxy(16,16);textcolor(10);
          cprintf("ON ");
          gotoxy(1,17);textcolor(15);
          cprintf("[                 ]");
          gotoxy(2,17);textcolor(14);
          cprintf("Tht Safe Gain   ");
          gotoxy(16,17);textcolor(10);
          cprintf("ON ");
          gotoxy(1,18);textcolor(15);
          cprintf("[                 ]");
          gotoxy(2,18);textcolor(14);
          cprintf("< >MODAL CTRL   ");
          gotoxy(3,18);textcolor(15+128);
          cprintf("m");
          gotoxy(16,18);textcolor(12+128);
          cprintf("OFF");
          gotoxy(1,19);textcolor(15);
          cprintf("[                 ]");
          gotoxy(2,19);textcolor(14);
          cprintf("< >EXCITATION    ");
          gotoxy(16,19);textcolor(12+128);
          cprintf("OFF");

C:

// --------------------------- CONTROL LOOP ------------------------------

loop:
```

```
             i_bot=1;i_top=1;i_th=1; n=0;
while (n <= nmax)
{
  if(diag == 0)
  {
    if (n == 1)
    {
      gotoxy(cir-1, 21);
      textcolor(9); cprintf(" >>> ");
      if(cir == 52)
      {
        gotoxy(cir-1, 21);
        cir = 25;
      }// End of if(cir == 52)
      cir++;
//    *******************************
      gotoxy(cir2, 21);
      textcolor(9); cprintf(" <<< ");
      if(cir2 == 25)
      {
        gotoxy(cir2, 21);
        cir2 = 52;
      }// End of if(cir2 == 25)
      cir2--;
    }// End of if(n == 1)
  }// End of if(diag == 0)

// ******************** Datel Board data input block ***********************

      if(flag16 == 0)// Non assembly condition
      {
        xbot        =  - inport(FIFO1);// - x0;// Channel 0
        ybot        =    inport(FIFO1);// + x0;// Channel 1

        xtop        =  - inport(FIFO1);// - x0;// Channel 2
        ytop        =    inport(FIFO1);// + x0;// Channel 3

//      ================================================

        zth1        =  - inport(FIFO2);// - x0;// Channel 0
        zth2        =    inport(FIFO2);// + x0;// Channel 1

        one_per_rev =    inport(FIFO2);// + x0;// Channel 2
        f_excite2   =    inport(FIFO2);// + x0;// Channel 3
      }

      else

      if(flag16 == 1)// Activates assembly block
      {
        asm{
             mov dx, [FIFO1]// Channel 0
             in  ax, dx
             neg ax
             sub ax, [x0]
             mov [xbot], ax
           }
        asm{
```

```
       mov dx, [FIFO1]// Channel 1
       in  ax, dx
       add ax, [x0]
       mov [ybot], ax
      }
    asm{
       mov dx, [FIFO1]// Channel 2
       in  ax, dx
       neg ax
       sub ax, [x0]
       mov [xtop], ax
      }
    asm{
       mov dx, [FIFO1]// Channel 3
       in  ax, dx
       add ax, [x0]
       mov [ytop], ax
      }
    asm{
       mov dx, [FIFO2]// Channel 0
       in  ax, dx
       neg ax
       sub ax, [x0]
       mov [zth1], ax
      }
    asm{
       mov dx, [FIFO2]// Channel 1
       in  ax, dx
       add ax, [x0]
       mov [zth2], ax
      }
    asm{
       mov dx, [FIFO2]// Channel 2
       in  ax, dx
       mov [one_per_rev], ax
      }
    asm{
       mov dx, [FIFO2]// Channel 3
       in  ax, dx
       mov [f_excite2], ax
      }
   }

// ***************** End Datel Board data input block **********************

// ********************* Signal generation block ***************************

  if(switch1 == 0)// Shuts down excitation function block when an
  {               // external excitation (switch=1) source is used
   if(flag5 == 1)// <4>
   {
    if(flag_AA == 1)
    {
     f_ex = t04 * sin(O*x*2.0*M_PI);// Sine
    }

    else

```

```
        if(flag_AA == 2)
        {
          f_ex = t04 * pow(sin(O*x/*2.0*/*M_PI),2);// Sine squared
        }

        else

        if(flag_AA == 3)
        {
          f_ex = t04 * cos(O*x*2.0*M_PI);// Cosine
        }

        else

        if(flag_AA == 4)
        {
          f_ex = t04 * pow(cos(O*x/*2.0*/*M_PI),2);// Cosine squared
        }

        else

        if(flag_AA == 5)
        {
          xy = xy + 1.0;
          srand(xy);

          if(flag21 == 1)// Excitation switch
          {
            for(i = 1; i <= 2; i++)
            {
              f_excite4  = (float(rand())/RAND_MAX);
            }
            for(i = 1; i <= 2/*NUMBERS*/; i++)
            {
              f_excite3 = float(rand())/RAND_MAX;
            }
          }// End of if(flag21 == 1)

          if(xy >= LIM)
             xy = 0.0;

             f_ex = t04 * sin(2.0*M_PI*f_excite3)*(sin(O*x*2.0*M_PI) +
                          sin(O*2.0*M_PI*f_excite4));// Random sine
        }// End of if(flag_AA == 5)
        {
          g = ceil(f_ex);
          z = f_ex + 0.5;

          if(g >= z)
             v = floor(f_ex);
          else
             v = g;

          if(flag21 == 1)// Excitation On/Off switch
          {
            f_excite = v;
          }
        }
```

```
              x = x + 0.002;
        }// End of if(flag5 == 1)// <4>

        else

        if(flag6 == 1)// <5>
        {
          while (k <= 40)// Forty terms in series
          {
            ex = ex + (1.0/(2.0*k+1.0))*sin(2.0*(2.0*k+1)*O*M_PI*x);
            k++;                            // Square wave pulse train
          }
            f_ex = t04 + t04 * (4.0/M_PI) * ex;
          {
            g = ceil(f_ex);
            z = f_ex + 0.5;

            if(g >= z)
               v = floor(f_ex);
            else
               v = g;

            if(flag21 == 1)// Excitation On/Off switch
            {
              f_excite = v / 2;
            }
          }
              x = x + 0.002;
              k = 0;
        }// End of if(flag6 == 1)// <5>

        else

        if(flag7 == 1)// <6>
        {
          while (k <= 40)// Forty terms in series
          {
            ex = ex + (1.0/(2.0*k+1.0))*sin(2.0*(2.0*k+1)*O*M_PI*x);
            k++;                                          // Square wave
          }
            f_ex = t04 * (4.0/M_PI) * ex;
          {
            g = ceil(f_ex);
            z = f_ex + 0.5;

            if(g >= z)
               v = floor(f_ex);
            else
               v = g;

            if(flag21 == 1)// Excitation On/Off switch
            {
              f_excite = v;
            }
          }
              x = x + 0.002;
              k = 0;
        }// End of if(flag7 == 1)// <6>
```

```
else

if(flag8 == 1)// <7>
{
  while (k <= 40)// Forty terms in series
  {
    ex = ex + (pow(-1,k)/pow((2.0*k+1.0),2))*sin(2.0*(2*k+1)*O*M_PI*x);
    k++;                                               // Saw tooth
  }
    f_ex = t04 * (8.0/pow(M_PI,2)) * ex;
  {
    g = ceil(f_ex);
    z = f_ex + 0.5;

    if(g >= z)
       v = floor(f_ex);
    else
       v = g;

    if(flag21 == 1)// Excitation On/Off switch
    {
      f_excite = v;
    }
  }
      x = x + 0.002;
      k = 0;
}// End of if(flag8 == 1)// <7>

else

if(flag9 == 1)// <8>
{
  C = PW;
  while (k1 <= 40)// Forty terms in series
  {
    ex = ex + (pow(-1,k1)/k1)*sin(k1*O*M_PI*C)*cos(2.0*k1*O*M_PI*x);
    k1++;                                     // Single square pulse
  }
    f_ex = t04 * (O * C + (2.0/M_PI) * ex);
  {
    g = ceil(f_ex);
    z = f_ex + 0.5;

    if(g >= z)
       v = floor(f_ex);
    else
       v = g;

    if(flag21 == 1)// Excitation switch
    {
      f_excite = v;
    }
  }
      x = x + 0.002;
      k1 = 1;
}// End of if(flag9 == 1)// <8>
```

```
      else

      if(flag12 == 1)// <9>
      {
        while (k <= 40)// Forty terms in series
        {
          ex = ex + (1.0/pow((2.0*k+1.0),2))*cos(2.0*(2.0*k+1.0)*O*M_PI*x);
          k++;                                          // Single triangular pulse
        }
          f_ex = t04 * (0.5 - (4.0/pow(M_PI,2)) * ex);
        {
          g = ceil(f_ex);
          z = f_ex + 0.5;

          if(g >= z)
             v = floor(f_ex);
          else
             v = g;

          if(flag21 == 1)// Excitation On/Off switch
          {
            f_excite = v;
          }
        }
            x = x + 0.002;
            k = 0;
      }// End of if(flag12 == 1)// <9>

      else

      if(flag13 == 1)// <0>
      {
        while (k1 <= 40)// Forty terms in series
        {
          ex = ex + (pow(-1,(k1+1))/(k1*1.0))*sin(2.0*k1*O*M_PI*x);// Saw tooth
//        ex = ex + (1/k1)*sin(k1*O*M_PI*x);
          k1++;
        }
          f_ex = t04 * (2.0/M_PI) * ex;
//        f_ex = t04 * (0.5 - 1.0/M_PI * ex);
        {
          g = ceil(f_ex);
          z = f_ex + 0.5;

          if(g >= z)
             v = floor(f_ex);
          else
             v = g;

          if(flag21 == 1)// Excitation On/Off switch
          {
            f_excite = v;
          }
        }
            x = x + 0.002;
            k1 = 1;
      }// End of if(flag13 == 1)// <0>
    }// End of if(switch1 == 0)
```

```

// ********************* End of signal generation block *******************

// ********************* External Excitation input Block ******************

      if(switch1 == 1)// External excitation flag
      {
        if(flag21 == 1)// Excitation On/Off switch
        {
          f_excite = f_excite2;// Datel input channel #3 on board #2
        }
      }

//   ***************** End of External Excitation Block ***************

//           ***************************************************
//           * This block is used to output the excitation signal *
//           ***************************************************

      if(test_signal == 1)
      {
        if(flag16 == 0)
           outport(out_chan1_0,(f_excite + t48));// Board 1

        else

        if(flag16 == 1)
        {
          asm{
               mov dx, [out_chan1_0]
               mov ax, [f_excite]
               add ax, [t48]
               out dx, ax
        }
      }
/*
        if(flag16 == 0)
           outport(out_chan2_0, (f_excite + t48);// Board 2

        else

        if(flag16 == 1)
        {
        asm{
             mov dx, [out_chan2_0]
             mov ax, [f_excite]
             out dx, ax
           }
        }
*/
      }// End of if(test_signal == 1)

//    ************** End of signal generator block **************

//                  *********************************
//                  * This block is used to generate *
//                  * the One - Per - Rev  signal    *
//                  *********************************
```

```
if(flag_II == 1)// one_per_rev set to on
{
  THETA = II * (PI2_o_Nticks * i_rev);
}

{
  f_excite_cos =  f_excite * cos(THETA);// X - AXIS
  f_excite_sin =  f_excite * sin(ns*THETA);// Y - AXIS
}

if(flag18 == 1)
{
  delay(60);// Delay 60 milli sec. - used for diagnostic purposes

  if(diag == 1)// Display # of period length(s) only in diagnostic mode
  {
    gotoxy(25,22);textcolor(11);
    cprintf("%5.1E     ",x - 0.002);
    gotoxy(25,23);
    printf("           ");// Erase y: display
  }
    gotoxy(14,25);textcolor(15);
    cprintf("%5d,%4u ",f_excite,n);
}// End of if(flag18 == 1)

if(n == 500)// Test for maximum # of loops in one period length
{
  x = x + 0.002;

  if(x > LIM)
  {
    x = 0.0;// Resets x to zero
  }
}// Ene of if(n == 500)
    ex = 0;// Summed ex values zeroed

if(flag_II == 1)// one_per_rev set to on
{
  if(one_per_rev < trigger)// No pulse condition, One_per_rev is < 0.1v
     rise = 1;
  if(rise == 1)
  if(one_per_rev >= trigger)// --> A pulse
  {
    rise = 0;
    N_ticks = j_rev;// # of loops in one revolution of the shaft
    if(N_ticks == 0)
       N_ticks = 1;

    PI2_o_Nticks = PI2/N_ticks;// Shaft radians per loop
    i_rev1 = (phi/360.0) * N_ticks;// phi: (0.0 --> 360.0) deg.
    {
      g = ceil(i_rev1);
      z = i_rev1 + 0.5;

      if(g >= z)
         v = floor(i_rev1);
      else
```

```
               v = g;

               i_rev = v;// After one shaft rotation i_rev = 0 if phi = 0
          }
               j_rev = 0;// After one revolution of shaft.
        }// End of if (one_per_rev >= trigger).

               i_rev++;  // Loop counter for one shaft rotation
                         // used to calculate (THETA).

               j_rev++;  // Loop counter for one shaft rotation
                         // used to calculate (PI2_o_Nticks).
               if(i_rev > N_ticks)
                  i_rev = i_rev - N_ticks;
      }// End of if(flag_II == 1)

//    ************** End of One - Per - Rev block **************

  if(flag16 == 0)// Non assembly condition.
  {
    // Commands board (1) to read next input value
    outport(0x300, one);
    outport(FIFO1, two);
    outport(0x300, 0xe);

    // Commands board (2) to read next input value
    outport(0x360, one);
    outport(FIFO2, two);
    outport(0x360, 0xe);
  }

  else

  if(flag16 == 1)// Assembly condition
  {               // Commands board (1) to read next input value
    asm{
         mov dx, 0x300
         mov ax, [one]
         out dx, ax
       }
    asm{
         mov dx, [FIFO1]
         mov ax, [two]
         out dx, ax
       }
    asm{
         mov dx, 0x300
         mov ax, 0xe
         out dx, ax
       }
// *************************
      // Commands board (2) to read next input value
    asm{
         mov dx, 0x360
         mov ax, [one]
         out dx, ax
       }
    asm{
         mov dx, [FIFO2]
         mov ax, [two]
         out dx, ax
       }
    asm{
         mov dx, 0x360
         mov ax, 0xe
         out dx, ax
       }
  }// End of if(flag16 == 1)

if(flag10 == 0)// Non modal condition
{
// **************************** LOWER BEARING ****************************

if(flag1 == 1)
{

//       * * * Begin x_force_bot calc * * *

  xbotderiv   = xbot - x_bot_old3;

//       * * * Calculate x_force_bot * * *

  x_force_bot = (((kh_bot * xbot + dh_bot * xbotderiv) * MCG)
                                - tBias_bot) + f_excite_cos;
  x_pos_output_bot = - x_force_bot - bias_current_bot;
  x_neg_output_bot = - x_force_bot + bias_current_bot;

//       * * *  OUTPUTS FOR x_direction_bot  * * *

// **********ROUNDING BLOCK**********
      g = ceil(x_pos_output_bot);
      z = x_pos_output_bot + 0.5;

      if(g >= z)
         v = floor(x_pos_output_bot);
      else
         v = g;

      round2 = v + t48;
// **********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan1_1, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_1]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }// End of if(round2 < out_min)

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan1_1, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_1]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }// End of if(round2 > out_max)

     else

     {
       if(flag16 == 0)
          outport(out_chan1_1, round2);// HORIZ.(RIGHT)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_1]
              mov ax, [round2]
              out dx, ax
            }
       }
     }
// **********ROUNDING BLOCK***********
       g = ceil(x_neg_output_bot);
       z = x_neg_output_bot + 0.5;

       if(g >= z)
          v = floor(x_neg_output_bot);
       else
          v = g;

       round2 = v + t48;
// **********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan1_2, out_min);

       else

       if(flag16 == 1)
```

```
        {
         asm{
              mov dx, [out_chan1_2]
              mov ax, [out_min]
              out dx, ax
            }
        }
      }// End of if(round2 < out_min)

      else

      if(round2 > out_max)
      {
        if(flag16 == 0)
           outport(out_chan1_2, out_max);

        else

        if(flag16 == 1)
        {
         asm{
              mov dx, [out_chan1_2]
              mov ax, [out_max]
              out dx, ax
            }
        }
      }// End of if(round2 > out_max)

      else

      {
        if(flag16 == 0)
           outport(out_chan1_2, round2);// HORIZ.(LEFT)

        else

        if(flag16 == 1)
        {
         asm{
              mov dx, [out_chan1_2]
              mov ax, [round2]
              out dx, ax
            }
        }
      }

// x_bot_old5 = x_bot_old4;
// x_bot_old4 = x_bot_old3;
   x_bot_old3 = x_bot_old2;
   x_bot_old2 = x_bot_old1;
   x_bot_old1 = xbot;

//              * * * End x_force_bot * * *

//       * * * Begin y_force_bot calc * * *

  ybotderiv   = ybot - y_bot_old3;

```

```
//       * * * Calculate y_force_bot * * *

  y_force_bot = (((kv_bot * ybot + dv_bot * ybotderiv) * MCG)
                                 - wBias_bot) + f_excite_sin;
  y_pos_output_bot =  y_force_bot - bias_current_bot;
  y_neg_output_bot =  y_force_bot + bias_current_bot;

//                 * * * OUTPUTS FOR y_direction_bot  * * *

// **********ROUNDING BLOCK**********
       g = ceil(y_pos_output_bot);
       z = y_pos_output_bot + 0.5;

       if(g >= z)
          v = floor(y_pos_output_bot);
       else
          v = g;

       round2 = v + t48;
// ***********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan1_3, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_3]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }// End of if(round2 < out_min)

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan1_3, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_3]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }// End of if(round2 > out_max)

     else
     {
       if(flag16 == 0)
          outport(out_chan1_3, round2);// VERT.(TOP)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_3]
              mov ax, [round2]
              out dx, ax
            }
       }
     }
// *********ROUNDING BLOCK*********
       g = ceil(y_neg_output_bot);
       z = y_neg_output_bot + 0.5;

       if(g >= z)
          v = floor(y_neg_output_bot);
       else
          v = g;

       round2 = v + t48;
// ***********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan1_4, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_4]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }// End of if(round2 < out_min)

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan1_4, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_4]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }// End of if(round2 > out_max)

     else

     {
       if(flag16 == 0)
          outport(out_chan1_4, round2);// VERT.(BOTTOM)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_4]
              mov ax, [round2]
              out dx, ax
            }
       }
     }

// y_bot_old5 = y_bot_old4;
// y_bot_old4 = y_bot_old3;
   y_bot_old3 = y_bot_old2;
   y_bot_old2 = y_bot_old1;
   y_bot_old1 = ybot;

//         * * * End y_force_bot * * *

//                      * * * Safe Gain * * *
     if (sg1 == 1)
     goto L1;

     else

     goto L2;

L1:  {
       if ((xbot * xbot + ybot * ybot) > safe)
       {
         kh_bot = 1.5; kv_bot = kh_bot;
         dh_bot = 9.0; dv_bot = dh_bot;
       }
       goto L2;
     }
//                      * * * End Safe Gain * * *

}// End of if(flag1 == 1)

L2:

if(diag == 1)
{
  if(flag4d == 1)
  {
```

```
  if(flag4a == 1)
  {
//   junk = exp(1.34567);
//   junk = exp(1.34567);
     junk = exp(1.34567);
     junk = exp(1.34567);
     junk = exp(1.34567);
     junk = exp(1.34567);
     junk = cos(1.34567);
     junk = cos(1.34567);
//   junk = cos(1.34567);
//   junk = cos(1.34567);
//   junk = cos(1.34567);
  }// End of if(flag4a == 1)
 }// End of if(flag4d == 1)
}// End of if(diag == 1)

// *************************** UPPER BEARING *******************************

if(flag2 == 1)
{
//     * * * Begin x_force_top calc * * *

 xtopderiv = xtop - x_top_old3;

//     * * * Calculate x_force_top * * *

 x_force_top = (((kh_top * xtop + dh_top * xtopderiv) * PCG)
              - tBias_top) + JJ * f_excite_cos;
 x_pos_output_top = - x_force_top - bias_current_top;
 x_neg_output_top = - x_force_top + bias_current_top;

//     * * *  OUTPUTS FOR x_direction_top  * * *

// **********ROUNDING BLOCK**********
   g = ceil(x_pos_output_top);
   z = x_pos_output_top + 0.5;

   if(g >= z)
     v = floor(x_pos_output_top);
   else
     v = g;

   round2 = v + t48;
// ***********************************

  if(round2 < out_min)
  {
   if(flag16 == 0)
     outport(out_chan2_1, out_min);

   else

   if(flag16 == 1)
   {
    asm{
       mov dx, [out_chan2_1]
       mov ax, [out_min]
```

```
              out dx, ax
             }
        }
      }// End of if(round2 < out_min)

      else

      if(round2 > out_max)
      {
       if(flag16 == 0)
          outport(out_chan2_1, out_max);

       else

       if(flag16 == 1)
       {
        asm{
             mov dx, [out_chan2_1]
             mov ax, [out_max]
             out dx, ax
           }
       }
      }// End of if(round2 > out_max)

      else

      {
       if(flag16 == 0)
          outport(out_chan2_1, round2);// HORIZ.(RIGHT)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_1]
              mov ax, [round2]
              out dx, ax
            }
       }
      }
// **********ROUNDING BLOCK**********
       g = ceil(x_neg_output_top);
       z = x_neg_output_top + 0.5;

       if(g >= z)
          v = floor(x_neg_output_top);
       else
          v = g;

       round2 = v + t48;
// **********************************

      if(round2 < out_min)
      {
       if(flag16 == 0)
          outport(out_chan2_2, out_min);

        else

        if(flag16 == 1)
        {
          asm{
               mov dx, [out_chan2_2]
               mov ax, [out_min]
               out dx, ax
             }
        }
      }// End of if(round2 < out_min)

      else

      if(round2 > out_max)
      {
        if(flag16 == 0)
           outport(out_chan2_2, out_max);

        else

        if(flag16 == 1)
        {
          asm{
               mov dx, [out_chan2_2]
               mov ax, [out_max]
               out dx, ax
             }
        }
      }// End of if(round2 > out_max)

      else

      {
        if(flag16 == 0)
           outport(out_chan2_2, round2);// HORIZ.(LEFT)

        else

        if(flag16 == 1)
        {
          asm{
               mov dx, [out_chan2_2]
               mov ax, [round2]
               out dx, ax
             }
        }
      }

// x_top_old5 = x_top_old4;
// x_top_old4 = x_top_old3;
   x_top_old3 = x_top_old2;
   x_top_old2 = x_top_old1;
   x_top_old1 = xtop;

//         * * * End x_force_top * * *

//       * * * Begin y_force_top calc * * *
```

```
  ytopderiv = ytop - y_top_old3;

//       * * * Calculate y_force_top * * *

  y_force_top = (((kv_top * ytop + dv_top * ytopderiv) * PCG)
                                 - wBias_top) + f_excite_sin;
  y_pos_output_top =  y_force_top - bias_current_top;
  y_neg_output_top =  y_force_top + bias_current_top;

//       * * *  OUTPUTS FOR y_direction_top  * * *

// *********ROUNDING BLOCK**********
       g = ceil(y_pos_output_top);
       z = y_pos_output_top + 0.5;

       if(g >= z)
          v = floor(y_pos_output_top);
       else
          v = g;

       round2 = v + t48;
// ************************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan2_3, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_3]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan2_3, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_3]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }

     else

     {
       if(flag16 == 0)
          outport(out_chan2_3, round2);// VERT.(TOP)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_3]
              mov ax, [round2]
              out dx, ax
            }
       }
     }
// **********ROUNDING BLOCK***********
       g = ceil(y_neg_output_top);
       z = y_neg_output_top + 0.5;

       if(g >= z)
          v = floor(y_neg_output_top);
       else
          v = g;

       round2 = v + t48;
// ************************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan2_4, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_4]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan2_4, out_max);

       else

       if(flag16 == 1)
```

```
        {
          asm{
               mov dx, [out_chan2_4]
               mov ax, [out_max]
               out dx, ax
             }
        }
      }

      else

      {
        if(flag16 == 0)
           outport(out_chan2_4, round2);// VERT.(BOTTOM)

        else

        if(flag16 == 1)
        {
          asm{
               mov dx, [out_chan2_4]
               mov ax, [round2]
               out dx, ax
             }
        }
      }

// y_top_old5 = y_top_old4;
// y_top_old4 = y_top_old3;
   y_top_old3 = y_top_old2;
   y_top_old2 = y_top_old1;
   y_top_old1 = ytop;

//         * * * End y_force_top * * *

//                       * * * Safe Gain * * *
     if (sg2 == 1)
     goto U1;

     else

     goto U2;

U1:  {
       if ((xtop * xtop + ytop * ytop) > safe)
       {
         kh_top = 1.5; kv_top = kh_top;
         dh_top = 9.0; dv_top = dh_top;
       }
       goto U2;
     }
//                       * * * End Safe Gain * * *

}// End of if(flag2 == 1)

U2:

if(diag == 1)
```

```
{
  if(flag4d == 1)
  {
    if(flag4b == 1)
    {
//    junk = exp(1.34567);
//    junk = exp(1.34567);
      junk = exp(1.34567);
      junk = exp(1.34567);
      junk = exp(1.34567);
      junk = exp(1.34567);
      junk = cos(1.34567);
      junk = cos(1.34567);
//    junk = cos(1.34567);
//    junk = cos(1.34567);
//    junk = cos(1.34567);
    }// End of if(diag == 1)
  }// End of if(flag4d == 1)
}// End of if(flag4b == 1)
}// End of if(flag10 == 0)

// ************************** MODAL CONTROL BLOCK **************************

if(flag10 == 1)// Modal condition
{
// ******************  Centralized Rigid Body Translation  ******************

  Xav = xbot * MCG + xtop * PCG;
  Yav = ybot * MCG + ytop * PCG;

  xbot_force_tr = (-(kh_bot + kh_top) * Xav - (dh_bot + dh_top) * dotXav);
  xtop_force_tr = (-(kh_bot + kh_top) * Xav - (dh_bot + dh_top) * dotXav);

  ybot_force_tr = (-(kv_bot + kv_top) * Yav - (dv_bot + dv_top) * dotYav);
  ytop_force_tr = (-(kv_bot + kv_top) * Yav - (dv_bot + dv_top) * dotYav);

  F_XB_tr = xbot_force_tr * MCG;// F1_X
  F_XT_tr = xtop_force_tr * PCG;// F2_X

  F_YB_tr = ybot_force_tr * MCG;// F1_Y
  F_YT_tr = ytop_force_tr * PCG;// F2_Y

// ******************  Centralized Rigid Body Rotation  ********************

  ThetaX = xbot - xtop;
  ThetaY = ybot - ytop;

  k_tilt = kh_top * MCG * MCG + kh_bot * PCG * PCG;
  c_tilt = dh_top * MCG * MCG + dh_bot * PCG * PCG;

  xtop_force_rot =  k_tilt * ThetaX + c_tilt * dotThetaX;
  xbot_force_rot = -k_tilt * ThetaX - c_tilt * dotThetaX;

  ytop_force_rot =  k_tilt * ThetaY + c_tilt * dotThetaY;
  ybot_force_rot = -k_tilt * ThetaY - c_tilt * dotThetaY;

// *********************  Centralized force summed  ************************

```

```
  xbot_force_modal_pos =   F_XB_tr + xbot_force_rot  + bias_current_bot;
  xbot_force_modal_neg = -(F_XB_tr + xbot_force_rot) + bias_current_bot;

  ybot_force_modal_pos =   F_YB_tr + ybot_force_rot  - bias_current_bot;
  ybot_force_modal_neg = -(F_YB_tr + ybot_force_rot) - bias_current_bot;
//----------------------------------------------------------------------
  xtop_force_modal_pos =   F_XT_tr + xtop_force_rot  + bias_current_top;
  xtop_force_modal_neg = -(F_XT_tr + xtop_force_rot) + bias_current_top;

  ytop_force_modal_pos =   F_YT_tr + xtop_force_rot  - bias_current_top;
  ytop_force_modal_neg = -(F_YT_tr + xtop_force_rot) - bias_current_top;
//----------------------------------------------------------------------
  x_pos_output_bot = xbot_force_modal_pos + f_excite_cos * -1;
  x_neg_output_bot = xbot_force_modal_neg + f_excite_cos * -1;

  y_pos_output_bot = ybot_force_modal_pos + f_excite_sin;
  y_neg_output_bot = ybot_force_modal_neg + f_excite_sin;
//----------------------------------------------------------------------
  x_pos_output_top = xtop_force_modal_pos + f_excite_cos * -1;
  x_neg_output_top = xtop_force_modal_neg + f_excite_cos * -1;

  y_pos_output_top = ytop_force_modal_pos + JJ * f_excite_sin;
  y_neg_output_top = ytop_force_modal_neg + JJ * f_excite_sin;

// Note that f_excite_cos is multiplied by -1 to give
// the correct One - Per - Rev vector rotation direction.

// ********** ROUNDING BLOCK - x_pos_output_bot **********
  g = ceil(x_pos_output_bot);
  z = x_pos_output_bot + 0.5;

  if(g >= z)
     v = floor(x_pos_output_bot);
  else
     v = g;

     X_P_O_B = v + t48;
//    ************
  if(X_P_O_B < out_min)
  {
    if(flag16 == 0)
       outport(out_chan1_1, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_1]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(X_P_O_B < out_min)

  else

  if(X_P_O_B > out_max)
  {
    if(flag16 == 0)
       outport(out_chan1_1, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_1]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(X_P_O_B > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan1_1, X_P_O_B);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_1]
           mov ax, [X_P_O_B]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - x_neg_output_bot **********
  g = ceil(x_neg_output_bot);
  z = x_neg_output_bot + 0.5;

  if(g >= z)
     v = floor(x_neg_output_bot);
  else
     v = g;

     X_N_O_B = v + t48;
//    ************
  if(X_N_O_B < out_min)
  {
    if(flag16 == 0)
       outport(out_chan1_2, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_2]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(X_N_O_B < out_min)

  else

  if(X_N_O_B > out_max)
  {
    if(flag16 == 0)
       outport(out_chan1_2, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_2]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(X_N_O_B > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan1_3, X_N_O_B);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_2]
           mov ax, [X_N_O_B]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - y_pos_output_bot **********
  g = ceil(y_pos_output_bot);
  z = y_pos_output_bot + 0.5;

  if(g >= z)
     v = floor(y_pos_output_bot);
  else
     v = g;

     Y_P_O_B = v + t48;
//      ************
  if(Y_P_O_B < out_min)
  {
    if(flag16 == 0)
       outport(out_chan1_3, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_3]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(Y_P_O_B < out_min)

  else

  if(Y_P_O_B > out_max)
  {
    if(flag16 == 0)
       outport(out_chan1_3, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_3]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(Y_P_O_B > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan1_3, Y_P_O_B);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_3]
           mov ax, [Y_P_O_B]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - y_neg_output_bot **********
  g = ceil(y_neg_output_bot);
  z = y_neg_output_bot + 0.5;

  if(g >= z)
    v = floor(y_neg_output_bot);
  else
    v = g;

    Y_N_O_B = v + t48;
//    ************
  if(Y_N_O_B < out_min)
  {
    if(flag16 == 0)
       outport(out_chan1_4, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_4]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(Y_N_O_B < out_min)

  else

  if(Y_N_O_B > out_max)
  {
    if(flag16 == 0)
       outport(out_chan1_4, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_4]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(Y_N_O_B > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan1_4, Y_N_O_B);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan1_4]
           mov ax, [Y_N_O_B]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - x_pos_output_top **********
  g = ceil(x_pos_output_top);
  z = x_pos_output_top + 0.5;

  if(g >= z)
     v = floor(x_pos_output_top);
  else
```

```
     v = g;

     X_P_O_T = v + t48;
//    ************
  if(X_P_O_T < out_min)
  {
    if(flag16 == 0)
       outport(out_chan2_1, out_min);

    else

    if(flag16 == 1)
    {
      asm{
          mov dx, [out_chan2_1]
          mov ax, [out_min]
          out dx, ax
         }
    }
  }// End of if(X_P_O_T < out_min)

  else

  if(X_P_O_T > out_max)
  {
    if(flag16 == 0)
       outport(out_chan2_1, out_max);

    else

    if(flag16 == 1)
    {
      asm{
          mov dx, [out_chan2_1]
          mov ax, [out_max]
          out dx, ax
         }
    }
  }// End of if(X_P_O_T > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan2_1, X_P_O_T);

    else

    if(flag16 == 1)
    {
      asm{
          mov dx, [out_chan2_1]
          mov ax, [X_P_O_T]
          out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - x_neg_output_top **********
```

```
  g = ceil(x_neg_output_top);
  z = x_neg_output_top + 0.5;

  if(g >= z)
     v = floor(x_neg_output_top);
  else
     v = g;

     X_N_O_T = v + t48;
//    ************
  if(X_N_O_T < out_min)
  {
    if(flag16 == 0)
       outport(out_chan2_2, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_2]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(X_N_O_T < out_min)

  else

  if(X_N_O_T > out_max)
  {
    if(flag16 == 0)
       outport(out_chan2_2, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_2]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(X_N_O_T > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan2_2, X_N_O_T);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_2]
           mov ax, [X_N_O_T]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - y_pos_output_top **********
  g = ceil(y_pos_output_top);
  z = y_pos_output_top + 0.5;

  if(g >= z)
     v = floor(y_pos_output_top);
  else
     v = g;

     Y_P_O_T = v + t48;
//     ************
  if(Y_P_O_T < out_min)
  {
    if(flag16 == 0)
       outport(out_chan2_3, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_3]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(Y_P_O_T < out_min)

  else

  if(Y_P_O_T > out_max)
  {
    if(flag16 == 0)
       outport(out_chan2_3, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_3]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(Y_P_O_T > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan2_3, Y_P_O_T);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_3]
           mov ax, [Y_P_O_T]
           out dx, ax
         }
    }
  }
// ********** ROUNDING BLOCK - y_neg_output_top **********
  g = ceil(y_neg_output_top);
  z = y_neg_output_top + 0.5;

  if(g >= z)
     v = floor(y_neg_output_top);
  else
     v = g;

     Y_N_O_T = v + t48;
//     ************
  if(Y_N_O_T < out_min)
  {
    if(flag16 == 0)
       outport(out_chan2_4, out_min);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_4]
           mov ax, [out_min]
           out dx, ax
         }
    }
  }// End of if(Y_N_O_T < out_min)

  else

  if(Y_N_O_T > out_max)
  {
    if(flag16 == 0)
       outport(out_chan2_4, out_max);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_4]
           mov ax, [out_max]
           out dx, ax
         }
    }
  }// End of if(Y_N_O_T > out_max)

  else

  {
    if(flag16 == 0)
       outport(out_chan2_4, Y_N_O_T);

    else

    if(flag16 == 1)
    {
      asm{
           mov dx, [out_chan2_4]
           mov ax, [Y_N_O_T]
           out dx, ax
         }
    }
  }

  dotXav = Xav - oldoldXav;
  oldoldXav = oldXav;
  oldXav = Xav;

  dotYav = Yav - oldoldYav;
  oldoldYav = oldYav;
  oldYav = Yav;

  dotThetaX = ThetaX - oldoldThetaX;
  oldoldThetaX = oldThetaX;
  oldThetaX = ThetaX;

  dotThetaY = ThetaY - oldoldThetaY;
  oldoldThetaY = oldThetaY;
  oldThetaY = ThetaY;
}// End of if(flag10 == 1)

// *****************************END MODAL CONTROL***************************

// ****************************** THRUST BEARING ***************************

if(flag3 == 1)
{
//       * * * Begin z_force_th calc * * *

  zth = (zth1 + zth2) / 2.0;
  zthderiv = zth - z_th_old3;
  zthsum = zthsum + igainth * zth;

//       * * * Calculate z_force_th * * *

  z_force_th = (kv_th * zth + dv_th * zthderiv) / 2.0 + zthsum
                                                      - tBias_th;
  up_output_th   =   z_force_th - bias_current_th;
  down_output_th =   z_force_th + bias_current_th;

//       * * *  OUTPUTS FOR z_direction_th  * * *

// **********ROUNDING BLOCK**********
       g = ceil(up_output_th);
       z = up_output_th + 0.5;

       if(g >= z)
          v = floor(up_output_th);
       else
          v = g;

       round2 = v + t48;
// ***********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan1_5, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_5]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan1_5, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_5]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }

     else

     {
       if(flag16 == 0)
          outport(out_chan1_5, round2);// VERT.(UP)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan1_5]
              mov ax, [round2]
              out dx, ax
            }
       }
     }
// ***********************************
       g = ceil(down_output_th);
       z = down_output_th + 0.5;

       if(g >= z)
          v = floor(down_output_th);
       else
          v = g;

       round2 = v + t48;
// ***********************************

     if(round2 < out_min)
     {
       if(flag16 == 0)
          outport(out_chan2_5, out_min);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_5]
              mov ax, [out_min]
              out dx, ax
            }
       }
     }

     else

     if(round2 > out_max)
     {
       if(flag16 == 0)
          outport(out_chan2_5, out_max);

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_5]
              mov ax, [out_max]
              out dx, ax
            }
       }
     }

     else

     {
       if(flag16 == 0)
          outport(out_chan2_5, round2);// VERT.(DOWN)

       else

       if(flag16 == 1)
       {
         asm{
              mov dx, [out_chan2_5]
              mov ax, [round2]
              out dx, ax
            }
       }
     }

// z_th_old5 = z_th_old4;
// z_th_old4 = z_th_old3;
   z_th_old3 = z_th_old2;
   z_th_old2 = z_th_old1;
   z_th_old1 = zth;

//         * * * End z_force_th * * *

}// End of if(flag3 == 1)

//                      * * * Safe Gain * * *
     if (sg3 == 1)
     goto T1;

     else

     goto T2;

T1:  {
       if ((zth * zth) > zsafe)
       {
         kv_th = 1.5;
         dv_th = 9.0;
       }
       goto T2;
     }
//                      * * * End Safe Gain * * *

T2:

if(diag == 1)
{
  if(flag4d == 1)
  {
    if(flag4c == 1)
    {

//    junk = exp(1.34567);
//    junk = exp(1.34567);
      junk = exp(1.34567);
      junk = exp(1.34567);
      junk = cos(1.34567);
      junk = cos(1.34567);
//    junk = cos(1.34567);
//    junk = cos(1.34567);
//    junk = cos(1.34567);
//    junk = cos(1.34567);
    }// End of if(flag4c == 1)
  }// End of if(flag4d == 1)
}// End of if(diag == 1)
  if(flag11 == 1)// Lower bearing write out activation flag
  {
    if(nw_bot == 1)
    {
      if(i_bot == 1)
      {
        gotoxy(51,22);textcolor(11);
        cprintf("%6.1fv         %6.1fv", xbot / 204.8, ybot / 204.8);
        gotoxy(49,23);textcolor(11);
        cprintf("%4.1fv,%6.1fv,%6.1fv,%6.1fv", x_pos_output_bot / 204.8,
                                              x_neg_output_bot / 204.8,
                                              y_pos_output_bot / 204.8,
                                              y_neg_output_bot / 204.8);

        if(flag10 == 0)// Activates when modal is off
        {
          gotoxy(25,22);textcolor(11);
          cprintf("%9.2fv",x_force_bot / 204.8);
          gotoxy(25,23);
          cprintf("%9.2fv",y_force_bot / 204.8);
        }
      }// End of if(i_bot == 1)
        i_bot = i_bot + 1;

        if(i_bot == 1025)
           i_bot = 1;
    }// End of if(nw_bot == 1)
  }// End of if(flag11 == 1)

  else

  if(flag22 == 1)// Upper bearing write out activation flag
  {
    if(nw_top == 1)
    {
      if(i_top == 1)
      {
        gotoxy(51,22);textcolor(11);
        cprintf("%6.1fv         %6.1fv", xtop / 204.8, ytop / 204.8);
        gotoxy(49,23);textcolor(11);
        cprintf("%4.1fv,%6.1fv,%6.1fv,%6.1fv", x_pos_output_top / 204.8,
                                              x_neg_output_top / 204.8,
                                              y_pos_output_top / 204.8,
                                              y_neg_output_top / 204.8);

        if(flag10 == 0)// Activates when modal is off
        {
          gotoxy(25,22);textcolor(11);
          cprintf("%9.2fv",x_force_top / 204.8);
          gotoxy(25,23);
          cprintf("%9.2fv",y_force_top / 204.8);
        }
      }// End of if(i_top == 1)
```

```
        i_top = i_top + 1;

        if(i_top == 1025)
           i_top = 1;
    }// End of if(nw_top == 1)
  }// End of if(flag22 == 1)

  else

  if(flag33 == 1)// Thrust bearing write out activation flag
  {
    if(nw_th == 1)
    {
      if(i_th == 1)
      {
        gotoxy(51,22);textcolor(11);
        cprintf("%6.1fv      ", zth / 204.8);
        gotoxy(49,23);textcolor(11);
        cprintf("%4.1fv,%6.1fv", up_output_th / 204.8,
                               down_output_th / 204.8);

        if(flag10 == 0)// Activates when modal is off
        {
          gotoxy(25,22);textcolor(11);
          cprintf("%9.2fv",z_force_th / 204.8);
        }
      }// End of if(i_th == 1)
        i_th = i_th + 1;

        if(i_th == 1025)
           i_th = 1;
    }// End of if(nw_th == 1)
  }// End of if(flag33 == 1)

n++;

}// End of while (n <= nmax) loop

// ************** Time & Loop time update block **************

     gettime(&tt);

     if(tt.ti_hour == 0)
     {
       hh = -12;
       gotoxy(48,10);textcolor(14);
       cprintf("AM");
     }

     else

     if(tt.ti_hour >= 1 && tt.ti_hour < 12)
     {
       hh = 0;
       gotoxy(48,10);textcolor(14);
       cprintf("AM");
     }

```

```
else

if(tt.ti_hour == 12)
{
  hh = 0;
  gotoxy(48,10);textcolor(14);
  cprintf("PM");
}

else

if(tt.ti_hour > 12 && tt.ti_hour < 24)
{
  hh = 12;
  gotoxy(48,10);textcolor(14);
  cprintf("PM");
}
  gotoxy(33,10);textcolor(14);
  cprintf("Time:");

  gotoxy(39,10);textcolor(11);
  cprintf("%2d:%02d:%02d\n",
  tt.ti_hour-hh, tt.ti_min, tt.ti_sec);

if(flag_L == 1)
{
  gotoxy(1,13);textcolor(14+128);
  cprintf("   QUIT(y/n)?: ");
}

if(l == lmax)// Time update block
{
  gettime(&now);
  last_time = time1;
  time1 = now.ti_sec + 0.01 * now.ti_hund + 60.0 * now.ti_min;
  loop_time =((time1 - last_time) * micro);

  if(abs(loop_time) < 800.0)
  {
    if(flag10 == 1 && diag == 1)
    {
      gotoxy(34,13);textcolor(15);
      cprintf("%6.2f", k_tilt);
      gotoxy(34,14);textcolor(15);
      cprintf("%6.2f",c_tilt);
    }
    if(nw_bot == 1 || nw_top == 1 || nw_th == 1)
    {
      gotoxy(62,22);textcolor(128+14);
      cprintf("w");
    }

    if(nw_bot == 0 && nw_top == 0 &&
       nw_th  == 0 && flag_B == 1 &&
       flag10 == 0 || flag10 == 1)
    {
      gotoxy(27,23);textcolor(14);
      cprintf("[<^> to toggle D.A. ]");
```

```
        }
        gotoxy(39,9);textcolor(15);
        cprintf("%6.2f",loop_time);

        if(flag24 == 1 && flag_K == 1)// Dynamic Averaging block
        {
          ii = ii + 1.0;

          A1  =  A2; A2  =  A3; A3  =  A4; A4  = A5;
          A5  =  A6; A6  =  A7; A7  =  A8; A8  = A9;
          A9  = A10; A10 = A11; A11 = A12; A12 = A13;
          A13 = A14; A14 = A15; A15 = loop_time;

          L_T = A1+A2+A3+A4+A5+A6+A7+A8+A9+A10+A11+A12+A13+A14+A15;
          LT = L_T / 15.0;// Average loop time

          PL = 1000000.0 / (freq*LT*500);// Period length
          O =  1/PL;// O = (1/period length), used in signal generation
                    // block

          qq = qq + 1;
          if(qq > vv)
          {
            qq = 0;
            ii = 0.0;
          }
        }// End of if(flag24 == 1 && flag_K == 1)

        else

        if(flag24 == 0 && flag_K == 1)// Intermittent Averaging block
        {
          if(rr == 0 && ii <= 15.0)
          {
            ii = ii + 1.0;// Counter
            OO = 1000000.0 / (freq*loop_time*500);// Period length
            OL = OL + OO;// Accumulated period length
            L_T = L_T + loop_time;
            if(ii == 15.0)
            {
              PL = OL / ii;// Average period length
              LT = L_T / ii;// Average loop time
              O  = 1.0 / PL;
              rr = 1;
              OL = 0.0;
              L_T = 0.0;
            }
          }// End of if(rr == 0 && ii <= 15.0)
          qq = qq + 1;
          if(qq > vv)
          {
            rr = 0;
            qq = 0;
            ii = 0.0;
          }
        }// End of if(flag24 == 0 && flag_K == 1)

        if(flag_K == 1)
```

```
      {
        if(flag_H == 1)
        {
          gotoxy(1,21);textcolor(15);
          cprintf("PL: %6.4f,%4.1f,%3u ",PL,ii,vv);
        }
        else
        if(flag_H == 0)
        {
          gotoxy(1,21);textcolor(15);
          cprintf("PL: %6.4f",PL);
        }
      }// End of if(flag_K == 1)

      if(resp == 'o' || resp == 'O')
      {
        frequency = (1000000.0/(PL*loop_time*500));
        gotoxy(1,21);textcolor(15);
        cprintf("PL: %6.4f                ",PL);
        gotoxy(1,23);textcolor(15);
        cprintf("<o>freq:%8.2f Hz.",frequency);
      }
      else
      {
        gotoxy(1,23);textcolor(15);
        cprintf("< >1/PL:             ");
        gotoxy(10,23);textcolor(15);
        cprintf("%7.3f", 0);

        if(ii < COUNTMAX)
        {
          gotoxy(2,23);textcolor(12+128);
          cprintf("o");
        }
        else
        {
          COUNTMAX = -1.0;
          gotoxy(2,23);textcolor(10);
          cprintf("o");
        }
      }
    // **************************************
    if(diag == 0)
    {
      flag_HH = flag_HH + 1;

      if(flag_HH == 1)
      {
        TC = 10;
        gotoxy(37,19);textcolor(TC);
        cprintf("  NASA  ");
      }
      else
      if(flag_HH == 2)
      {
        TC = 11;
        gotoxy(37,19);textcolor(TC);
        cprintf(" GLENN  ");
      }
      else
      if(flag_HH == 3)
      {
        TC = 13;
        gotoxy(37,19);textcolor(TC);
        cprintf("RESEARCH");
      }
      else
      if(flag_HH == 4)
      {
        TC = 14;
        gotoxy(37,19);textcolor(TC);
        cprintf(" CENTER ");
      }
      if(flag_HH >= 4)
         flag_HH = 0;
    }// End of if(diag == 0)
    // ***************************************
      if(flag_BB == 1)
      {
        gotoxy(1,2);textcolor(15);
        cprintf("<q> to abort control     ");

        if(flag_B == 1 && flag44 == 0 || diag == 0)
        {
          if(flag10 == 0 && nw_bot == 0 &&
             nw_top == 0 && nw_th  == 0 || flag10 == 1)
            {
              gotoxy(42,23);textcolor(12);
              cprintf("I.A.");
            }
        }
        if(diag == 0)
        {
          gotoxy(1,5);textcolor(15);
          cprintf("<4-0> to select excitation ");

          gotoxy(1,3);textcolor(15);
          cprintf("<m> to toggle modal cntrl    ");

          gotoxy(1,4);textcolor(15);
          cprintf("<?> to toggle f_excite         ");
        }
        if(diag == 1)
        {
          gotoxy(1,1);textcolor(15);
          cprintf("<+,-> to toggle input-output writes");

          gotoxy(1,3);textcolor(15);
          cprintf("<f> to toggle loop time buffer");

          gotoxy(1,4);textcolor(15);
          cprintf("<e> non diagnostic             ");

          gotoxy(1,5);textcolor(15);
          cprintf("<!,@,#> disable safe gain   ");
        }
```

```
          if(switch1 == 1)
          {
            gotoxy(1,25);textcolor(13);
            cprintf("[        ]       ");
            gotoxy(2,25);textcolor(14);
            cprintf("f_excite2");
            gotoxy(13,25);textcolor(15+128);
            cprintf("<==");
          }
          else
          if(switch1 == 0)
          {
            gotoxy(1,25);textcolor(15);
            cprintf("<?>f_excite :           ");
          }
          gotoxy(2,20);textcolor(10);
          cprintf("k");

          if(flag_N == 1)
          {
            if(flag_jj == 1)
            {
              gotoxy(1,14);textcolor(10);
              cprintf("< >PHSE ANG:%3u deg ",th);
              gotoxy(2,14);textcolor(15);
              cprintf("n");
              flag_jj = 0;
            }
            else
            if(flag_jj == 0)
            {
              gotoxy(1,14);textcolor(13);
              cprintf("<  >phi ANG:%3u deg",thp);
              gotoxy(2,14);textcolor(15);
              cprintf("{}");
              flag_jj = 1;
            }
          }
          flag_BB = 0;
        }// End of if(flag_BB == 1)
        else
        if(flag_BB == 0)
        {
          if(diag == 1)
          {
            gotoxy(1,1);textcolor(9);
            cprintf("<4-0> to select excitation         ");
          }

          if(flag_B == 1 && flag44 == 0 || diag == 0 )
          {
            if(flag10 == 0 && nw_bot == 0 &&
               nw_top == 0 && nw_th  == 0 || flag10 == 1)
            {
              gotoxy(42,23);textcolor(10);
              cprintf("D.A.");
            }
          }
```

```
           gotoxy(1,2);textcolor(10);
           cprintf("<R> to toggle Bounce/Tilt");

           if(diag == 1 || diag == 0)
           {
             if(flagMM == 0 && switch1 == 0)
             {
               gotoxy(1,25);textcolor(15);
               cprintf("<s>to adjust Pulse Width");
             }
             gotoxy(1,3);textcolor(13);
             cprintf("<F> to toggle O.P.R. dirction ");

             gotoxy(1,4);textcolor(14);
             cprintf("<<> to toggle ext.input.exction");

             gotoxy(1,5);textcolor(11);
             cprintf("<&,*>avrg freq update adjst");
           }// End of if(diag == 1 || diag == 0)
           gotoxy(2,20);textcolor(12);
           cprintf("x");
           flag_BB = 1;

           if(flag_N == 1)
           {
             gotoxy(1,14);textcolor(15);
             cprintf("[                 ]");
             gotoxy(2,14);textcolor(14);
             cprintf("< >ONE_PR_REV");
             gotoxy(3,14);textcolor(10);
             cprintf("r");
             gotoxy(16,14);textcolor(12+128);
             cprintf("OFF");
           }
         }// End of if(flag_BB == 0)
       // ***************************************
       }// End of if(abs(loop_time) < 800.0)
         l = 0;
     }// End of if(l == lmax)
         l++;

         hh = kbhit();

         if(hh == 0)
         goto loop;

         else

         {
           resp = getch();
           hh = 0;
         }

if(diag == 1 && SSS == 1)
{
    if(resp == 'q' || resp == 'Q')
    {
       flag_L = 1;
       goto loop;
    }
    if(flag_L == 1)
    {
      if(resp == 'y' || resp == 'Y')
      goto ramp_down;

      if(resp == 'n' || resp == 'N')
      {
        gotoxy(1,13);textcolor(14);
        cprintf("                   ");// ERASE QUIT(y/n)?:
        flag_L = 0;
        goto loop;
      }
    }

    if(resp == 'p') goto kv_up;        if(resp == 'P') goto kv_down;
    if(resp == 'd') goto dh_up;        if(resp == 'D') goto dh_down;
    if(resp == 'g') goto kh_up;        if(resp == 'G') goto kh_down;
    if(resp == 'v') goto dv_up;        if(resp == 'V') goto dv_down;
    if(resp == 'w') goto wBias_up;     if(resp == 'W') goto wBias_down;
    if(resp == 't') goto tBias_up;     if(resp == 'T') goto tBias_down;
    if(resp == 'b') goto bias_up;      if(resp == 'B') goto bias_down;
    if(resp == 'f') goto buffer;       if(resp == 'M') goto test_signal;
    if(resp == '+') goto writeout;     if(resp == '-') goto nowrite;
    if(resp == ')') goto igain_up;     if(resp == '(') goto igain_down;
    if(resp == '!') goto disable_safe1;if(resp == '1') goto enable_safe1;
    if(resp == '@') goto disable_safe2;if(resp == '2') goto enable_safe2;
    if(resp == '#') goto disable_safe3;if(resp == '3') goto enable_safe3;
    if(resp == 'e') goto non_diagnostic;

    if(resp == 'H') goto thrust_bearing;
    if(resp == 'I') goto upper_bearing;
    if(resp == 'J') goto lower_bearing;

    if(resp == 'l') goto l_on;
    if(resp == 'u') goto u_on;
    if(resp == 'z') goto z_on;
}// End of if(diag == 1 && SSS == 1)

if(diag == 1 || diag == 0)
{
    if(resp == 'q' || resp == 'Q')
    {
       flag_L = 1;
       goto loop;
    }
    if(flag_L == 1)
    {
      if(resp == 'y' || resp == 'Y')
      goto ramp_down;

      if(resp == 'n' || resp == 'N')
      {
        gotoxy(1,13);textcolor(14);
        cprintf("                   ");// ERASE QUIT(y/n)?:
        flag_L = 0;
        goto loop;
      }
    }

    if(resp == 'c') goto cg_factor_up;
    if(resp == 'C') goto cg_factor_down;
    if(resp == 'E') goto diagnostic;
    if(resp == 'm') goto modal;
    if(resp == 'o') goto frequency_up;
    if(resp == 'O') goto frequency_down;
    if(resp == 'a') goto amplitude_up;
    if(resp == 'A') goto amplitude_down;
    if(resp == ':') goto assembly;
    if(resp == '?') goto display;
    if(resp == ',') goto excitation;
    if(resp == '<') goto excitation_switch;
    if(resp == '*') goto vv_up;
    if(resp == '&') goto vv_down;
    if(resp == '$') goto excite1_toggle;
    if(resp == '^') goto loop_time_average_toggle;
    if(resp == '{') goto phi_down;
    if(resp == '}') goto phi_up;

    if(resp == '4') goto excite1;
    if(resp == '5') goto excite2;
    if(resp == '6') goto excite3;
    if(resp == '7') goto excite4;
    if(resp == '8') goto excite5;
    if(resp == '9') goto excite6;
    if(resp == '0') goto excite7;

    if(resp == 's') goto pulse_width_up;
    if(resp == 'S') goto pulse_width_down;
    if(resp == 'x') goto frequency_input_up;
    if(resp == 'X') goto frequency_input_down;
    if(resp == 'k') goto freq_fine_adjust_up;
    if(resp == 'K') goto freq_fine_adjust_down;
    if(resp == 'n') goto THETA_up;
    if(resp == 'N') goto THETA_down;
    if(resp == 'r') goto one_per_rev;
    if(resp == 'R') goto Tilt_Bounce_Mode;
    if(resp == 'F') goto one_p_rev_dir;
    goto loop;
}// End of if(diag == 1 || diag == 0)

loop_time_average_toggle:{
                           if(flag_K == 1)
                           {
                             if(flag25 == 1)
                             {
                               flag24 = 0;
                               flag25 = 0;
                               flag_H = 1;
                               gotoxy(21,21);textcolor(12);
                               cprintf("I.A.");
                               rr = 0;
                               OL = 0.0;
                               L_T = 0.0;
                               qq = 0;
                               ii = 0.0;
                             }
                             else
                             if(flag25 == 0)
                             {
                               flag24 = 1;
                               flag25 = 1;
                               flag_H = 1;
                               gotoxy(1,21);textcolor(15);
                               cprintf("PL: %6.4f,%4.1f,%3u",PL,ii,vv=15);
                               gotoxy(21,21);textcolor(10);
                               cprintf("D.A.");
                             }
                             goto loop;
                           }// End of if(flag_K = 1)
                           goto loop;
                         }
vv_up:{
        if(flag24 == 0 && flag_K == 1)
        {
          OL = 0.0;
          L_T = 0.0;
          rr = 0;
          qq = 0;
          ii = 0.0;
          vv = vv + 1;
          if(vv >= 100)
             vv = 100;
          gotoxy(17,21);textcolor(15);
          cprintf("%3u",vv);
          goto loop;
        }
        goto loop;
      }
vv_down:{
          if(flag24 == 0 && flag_K == 1)
          {
            OL = 0.0;
            L_T = 0.0;
            rr = 0;
            qq = 0;
            ii = 0.0;
            vv = vv - 1;
            if(vv <= 15)
               vv = 15;
            gotoxy(17,21);textcolor(15);
            cprintf("%3u",vv);
            goto loop;
          }
          goto loop;
        }
excitation_switch:{
                    if(flagNN == 1)
                    {
                      switch1 = 1;
                      flagNN = 0;
                      gotoxy(1,25);textcolor(13);
                      cprintf("[         ]       ");
                      gotoxy(2,25);textcolor(14);
                      cprintf("f_excite2");
                      gotoxy(13,25);textcolor(15);
                      cprintf("<==         ");
                    }
                    else
                    if(flagNN == 0)
                    {
                      COUNTMAX = 15.0;
                      OL = 0.0;
                      L_T = 0.0;
                      rr = 0;
                      qq = 0;
                      ii = 0.0;
                      switch1 = 0;
                      flagNN = 1;
                      gotoxy(1,25);textcolor(15);
                      cprintf("<?>f_excite :%5d",f_excite);
                    }
                    goto loop;
                  }
test_signal:{
              if(flagLL == 1)
              {
                test_signal = 1;
                flagLL = 0;
                gotoxy(36,11);textcolor(13);
                cprintf("<M>-test: %1u",test_signal);
                gotoxy(46,11);textcolor(12);
                cprintf("%1u",test_signal);
              }
              else
              if(flagLL == 0)
              {
               test_signal = 0;
               flagLL = 1;
               gotoxy(37,11);textcolor(15);
               cprintf("M");
               gotoxy(46,11);textcolor(10);
               cprintf("%1u",test_signal);
              }
              goto loop;
            }
one_p_rev_dir:{
                if(flag_N == 0)
                {
                  gotoxy(21,6);textcolor(13);
                  cprintf("O.P.R.");
                  gotoxy(28,6);textcolor(13+128);
                  cprintf("----->");

                  if(flagKK == 1)
                  {
                    II = -1.0;
                    gotoxy(36,6);textcolor(11);
                    cprintf("Anti clkwse");
                    flagKK = 0;
                  }
```

```
                   else
                   if(flagKK == 0)
                   {
                     II = 1.0;
                     gotoxy(36,6);textcolor(11);
                     cprintf("  Clckwse  ");
                     flagKK = 1;
                   }
                   goto loop;
                 }
                 goto loop;
               }
Tilt_Bounce_Mode:{
                    if(flagJJ == 1)
                    {
                      JJ = 1.0;
                      flagJJ = 0;
                      gotoxy(32,7);textcolor(15);
                      cprintf("==>             <==");
                      gotoxy(36,7);textcolor(14+128);
                      cprintf("BOUNCE MODE");
                    }
                    else
                    if(flagJJ == 0)
                    {
                      JJ = -1.0;
                      flagJJ = 1;
                      gotoxy(32,7);textcolor(15);
                      cprintf("==>             <==");
                      gotoxy(36,7);textcolor(13+128);
                      cprintf(" TILT MODE ");
                    }
                    goto loop;
                  }
one_per_rev:{
              if(flag_M == 1)// Toggle on flag
              {
                ns = 1.0;// Condition for correct manual vector rotation
                gotoxy(1,14);textcolor(15);
                cprintf("[                  ]");
                gotoxy(2,14);textcolor(14);
                cprintf("< >ONE_PR_REV");
                gotoxy(3,14);textcolor(10);
                cprintf("r");
                gotoxy(16,14);textcolor(10);
                cprintf("ON");
                flag_II = 1;// one_per_rev set to on
                flag_M  = 0;
                flag_N  = 0;
                goto loop;
              }
              else
              if(flag_M == 0)// Toggle off flag
              {
                gotoxy(16,14);textcolor(12+128);
                cprintf("OFF");
                flag_II = 0;
                flag_M  = 1;
                flag_N  = 1;
                THETA = 0.0;
                th = 0;
                goto loop;
              }
            }
THETA_up:{
           if(flag_II == 0)// One - Per - Rev is Off
           {
             ns = -1.0;// Condition for correct manual vector rotation
             THETA = THETA + 5.0 * M_PI/180.0;
             th = th + 5;
             if(THETA >= 2.0 * M_PI)
             {
               THETA = 2.0 * M_PI;
               th = 360;
             }
             gotoxy(1,14);textcolor(10);
             cprintf("< >PHSE ANG:    deg");
             gotoxy(2,14);textcolor(15);
             cprintf("n");
             gotoxy(13,14);textcolor(15);
             cprintf("%3u",th);
             goto loop;
           }// End of if(flag_II == 0)
           goto loop;
          }
THETA_down:{
             if(flag_II == 0)
             {
               ns = -1.0;// Condition for correct manual vector rotation
               THETA = THETA - 5.0 * M_PI/180.0;
               th = th - 5;
               if(THETA <= 0.0 && th <= 0)
               {
                 THETA = 0.0;
                 th = 0;
               }
               gotoxy(1,14);textcolor(10);
               cprintf("< >PHSE ANG:    deg");
               gotoxy(2,14);textcolor(15);
               cprintf("n");
               gotoxy(13,14);textcolor(15);
               cprintf("%3u",th);
               goto loop;
             }// End of if(flag_II == 0)
             goto loop;
           }
phi_up:{
         if(flag_II == 0)
         {
           phi = phi + 5.0 * M_PI/180.0;
           thp = thp + 5;
           if(phi >= 2.0 * M_PI)
           {
             phi = 2.0 * M_PI;
             thp = 360;
           }
           gotoxy(1,14);textcolor(13);
           cprintf("<  >phi ANG:    deg");
           gotoxy(2,14);textcolor(15);
           cprintf("{}");
           gotoxy(13,14);textcolor(15);
           cprintf("%3u",thp);
           goto loop;
         }// End of if(flag_II == 0)
         goto loop;
       }
phi_down:{
           if(flag_II == 0)
           {
             phi = phi - 5.0 * M_PI/180.0;
             thp = thp - 5;
             if(phi <= 0.0 && thp <= 0)
             {
               phi = 0.0;
               thp = 0;
             }
             gotoxy(1,14);textcolor(13);
             cprintf("<  >phi ANG:    deg");
             gotoxy(2,14);textcolor(15);
             cprintf("{}");
             gotoxy(13,14);textcolor(15);
             cprintf("%3u",thp);
             goto loop;
           }// End of if(flag_JJ == 0)
           goto loop;
         }
assembly:{
           if(flag_A == 0)
           {
             flag16 = 1;
             gotoxy(42,25);textcolor(10);
             cprintf("ON ");
             flag_A = 1;
             goto loop;
           }
           else
           if(flag_A == 1)
           {
             flag16 = 0;
             gotoxy(42,25);textcolor(12+128);
             cprintf("OFF");
             flag_A = 0;
             goto loop;
           }
         }
display:{
          if(nw_bot == 0 && nw_top == 0 && nw_th == 0)
          {
            if(flag_B == 1)
            {
              flag18 = 1;
              flagMM = 1;
              gotoxy(26,20);textcolor(15);
              cprintf("         ");// Erase "Force(N)"
              if(diag == 1)
              {
                gotoxy(27,23);// Erase [<^> to toggle D.A. ]
                cprintf("                     ");
              }
              gotoxy(1,25);textcolor(15);
              cprintf("<?>f_excite :%5d",f_excite);
              flag_B = 0;
              goto loop;
            }
            else
            if(flag_B == 0)
            {
              flag18 = 0;
              flagMM = 0;
              if(diag == 1)
              {
               gotoxy(26,20);textcolor(15);
               cprintf("Force (N)");
              }
              gotoxy(25,22);
              printf("         ");// Erase period length x: values
              gotoxy(27,23);textcolor(14);
              cprintf("[<^> to toggle D.A. ]");
              gotoxy(1,25);textcolor(15);
              cprintf("<s>to adjust Pulse Width");
              flag_B = 1;
              goto loop;
            }
          }
          goto loop;
        }
excitation:{
             if(flag_C == 1)
             {
               flag21 = 1;
               gotoxy(32,24);textcolor(10);
               cprintf("Enable");
               flag_C = 0;
               goto loop;
             }
             else
             if(flag_C == 0)
             {
               flag21 = 0;
               gotoxy(32,24);textcolor(12);
               cprintf("Dsable");
               flag_C = 1;
               goto loop;
             }
           }
amplitude_up:{
               t04  = t04  + 102.4*0.2;
               volt = volt + 0.1;
               if(t04 > 1024)
               {
                 t04  = 1024;
                 volt = 5.0;
               }
               gotoxy(14,24);textcolor(15);
               cprintf("%4.1f",volt);
               goto loop;
             }
amplitude_down:{
                 t04  = t04  - 102.4*0.2;
                 volt = volt - 0.1;
                 if(t04 <= 0.0)
                 {
                   t04  = 0.0;
                   volt = 0.0;
                 }
                 gotoxy(14,24);textcolor(15);
                 cprintf("%4.1f",volt);
                 goto loop;
               }
frequency_input_up:{
                     COUNTMAX = 15.0;
                     flag_K = 1;
                     flag24 = 1;
                     vv = 15;// used only for default display of
                             // D.A. mode upper limit
                     if(freq == 1)
                        freq = 0;
                     freq = freq + 10.0;
                     if(freq > 5000.0)
                     freq = 5000.0;

                     gotoxy(2,20);textcolor(12);
                     cprintf("x");

                     gotoxy(13,20);textcolor(15);
                     cprintf("%7.1f Hz.",freq);

                     if(flag_H == 1)
                     {
                       gotoxy(21,21);textcolor(10);
                       cprintf("D.A.");
                     }

                     rr = 0;
                     OL = 0.0;
                     L_T = 0.0;
                     qq = 0;
                     ii = 0.0;

                     goto loop;
                   }
freq_fine_adjust_up:{
                      COUNTMAX = 15.0;
                      flag_K = 1;
                      flag24 = 1;
                      vv = 15;// used only for default display of
                              // D.A. mode upper limit
                      freq = freq + 0.1;
                      if(freq > 5000.0)
                         freq = 5000.0;

                      gotoxy(2,20);textcolor(10);
                      cprintf("k");

                      gotoxy(13,20);textcolor(15);
                      cprintf("%7.1f Hz.",freq);

                      if(flag_H == 1)
                      {
                        gotoxy(21,21);textcolor(10);
                        cprintf("D.A.");
                      }

                      rr = 0;
                      OL = 0.0;
                      L_T = 0.0;
                      qq = 0;
                      ii = 0.0;

                      goto loop;
                    }
frequency_input_down:{
                       COUNTMAX = 15.0;
                       flag_K = 1;
                       flag24 = 1;
                       vv = 15;// used only for default display of
                               // D.A. mode upper limit
                       freq = freq - 10.0;
                       if(freq <= 0)
                          freq = 10.0;

                       gotoxy(2,20);textcolor(12);
                       cprintf("x");

                       gotoxy(13,20);textcolor(15);
                       cprintf("%7.1f Hz.",freq);

                       if(flag_H == 1)
                       {
                         gotoxy(21,21);textcolor(10);
                         cprintf("D.A.");
                       }

                       rr = 0;
                       OL = 0.0;
                       L_T = 0.0;
                       qq = 0;
                       ii = 0.0;

                       goto loop;
                     }
freq_fine_adjust_down:{
                        COUNTMAX = 15.0;
                        flag_K = 1;
                        flag24 = 1;
                        vv = 15;// Used only for default display of
                                // D.A. mode upper limit
                        freq = freq - 0.1;
                        if(freq < 0.0)
                           freq = 10.0;

                        gotoxy(2,20);textcolor(10);
                        cprintf("k");

                        gotoxy(13,20);textcolor(15);
                        cprintf("%7.1f Hz.",freq);

                        if(flag_H == 1)
                        {
                          gotoxy(21,21);textcolor(10);
                          cprintf("D.A.");
                        }

                        rr = 0;
                        OL = 0.0;
                        L_T = 0.0;
                        qq = 0;
                        ii = 0.0;

                        goto loop;
                      }
frequency_up:{
               flag_K = 0;
               PL = PL - 0.002;
               if(PL <= 0.0)
                  PL = 0.002;
               O = 1.0/PL;
               gotoxy(1,21);textcolor(15);
               cprintf("PL: %6.4f              ",PL);
               goto loop;
             }
frequency_down:{
                 flag_K = 0;
                 PL = PL + 0.002;
                 if(PL > 1.0)
                    PL = 1.0;
                 O = 1.0/PL;
                 gotoxy(1,21);textcolor(15);
                 cprintf("PL: %6.4f              ",PL);
                 goto loop;
               }
pulse_width_up:{
                 if(flag9 == 1)
                 {
                   flag_H = 0;
                   PWW = PWW + 1.0;
                   PW = 1.0/(2.0*PWW);
                   gotoxy(13,21);textcolor(15);
                   cprintf("PW: %6.4f  ",PW);
                   goto loop;
                 }
                 goto loop;
               }
pulse_width_down:{
                   if(flag9 == 1)
                   {
                     flag_H = 0;
                     PWW = PWW - 1.0;
                     if(PWW <= 0.0)
                     PWW = 1.0;
                     PW = 1.0/(2.0*PWW);
                     gotoxy(13,21);textcolor(15);
                     cprintf("PW: %6.4f  ",PW);
                     goto loop;
                   }
                   goto loop;
                 }
excite1:{
          flag6 = 0;
          flag7 = 0;
          flag8 = 0;
          flag9 = 0;
          flag12 = 0;
          flag13 = 0;
          COUNTMAX = 15.0;
          flag_H = 1;

          flag_AA = flag_AA + 1;

          if(flag_AA > 5)
          {
            flag_AA = 1;
          }

          if(flag_AA == 1)
          {
            gotoxy(2,19);textcolor(14);
            cprintf("SINE             ");

            if(flag5 == 1)
            {
              gotoxy(16,19);textcolor(10);
              cprintf("ON ");
            }
            else
            if(flag5 == 0)
            {
              gotoxy(16,19);textcolor(12+128);
              cprintf("OFF");
            }
          }
          else
          if(flag_AA == 2)
          {
            gotoxy(2,19);textcolor(14);
            cprintf("SINE SQUARED     ");

            if(flag5 == 1)
            {
              gotoxy(16,19);textcolor(10);
              cprintf("ON ");
            }
            else
            if(flag5 == 0)
          {
            gotoxy(16,19);textcolor(12+128);
            cprintf("OFF");
          }
        }
        else
        if(flag_AA == 3)
        {
          gotoxy(2,19);textcolor(14);
          cprintf("COSINE           ");

          if(flag5 == 1)
          {
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
          }
          else
          if(flag5 == 0)
          {
            gotoxy(16,19);textcolor(12+128);
            cprintf("OFF");
          }
        }
        else
        if(flag_AA == 4)
        {
          gotoxy(2,19);textcolor(14);
          cprintf("COSINE SQARED    ");

          if(flag5 == 1)
          {
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
          }
          else
          if(flag5 == 0)
          {
            gotoxy(16,19);textcolor(12+128);
            cprintf("OFF");
          }
        }
        else
        if(flag_AA == 5)
        {
          gotoxy(2,19);textcolor(14);
          cprintf("RANDOM           ");

          if(flag5 == 1)
          {
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
          }
          else
          if(flag5 == 0)
          {
            gotoxy(16,19);textcolor(12+128);
            cprintf("OFF");
          }
          }
          goto loop;
        }
excite1_toggle:{
                 if(flag_D == 1)
                 {
                   flag5 = 1;// <4>
                   flag6 = 0;
                   flag7 = 0;
                   flag8 = 0;
                   flag9 = 0;
                   flag12 = 0;
                   flag13 = 0;
                   num = 4;
                   gotoxy(16,19);textcolor(10);
                   cprintf("ON ");
                   gotoxy(13,21);textcolor(15);
                   cprintf("            ");// Erase "PW: %6.4f  "
                   gotoxy(21,21);textcolor(10);
                   cprintf("D.A.");
                   flag_D = 0;
                   flag_E = 1;
                   flag_F = 1;
                   flag_G = 1;
                   flag_H = 1;
                   flag_I = 1;
                   flag_J = 1;

                   rr = 0;
                   OL = 0.0;
                   L_T = 0.0;
                   qq = 0;
                   ii = 0.0;

                   goto loop;
                 }
                 else
                 if(flag_D == 0)
                 {
                   if(flag5 == 1)
                   {
                     flag5 = 0;
                     gotoxy(16,19);textcolor(12+128);
                     cprintf("OFF");
                     flag_D = 1;
                     flag_E = 1;
                     flag_F = 1;
                     flag_G = 1;
                     flag_H = 1;
                     flag_I = 1;
                     flag_J = 1;
                   }
                   goto loop;
                 }
               }
excite2:{
          if(flag_E == 1)
          {
            COUNTMAX = 15.0;
            flag5 = 0;
            flag6 = 1;// <5>
            flag7 = 0;
            flag8 = 0;
            flag9 = 0;
            flag12 = 0;
            flag13 = 0;
            gotoxy(2,19);textcolor(14);
            cprintf("< >EXCITATION    ");
            gotoxy(3,19);textcolor(14);
            cprintf("5");
            num = 5;
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
            gotoxy(13,21);textcolor(15);
            cprintf("            ");
            gotoxy(21,21);textcolor(10);
            cprintf("D.A.");
            flag_E = 0;
            flag_D = 1;
            flag_F = 1;
            flag_G = 1;
            flag_H = 1;
            flag_I = 1;
            flag_J = 1;

            rr = 0;
            OL = 0.0;
            L_T = 0.0;
            qq = 0;
            ii = 0.0;

            goto loop;
          }
          else
          if(flag_E == 0)
          {
            if(flag6 == 1)
            {
              flag6 = 0;
              gotoxy(16,19);textcolor(12+128);
              cprintf("OFF");
              flag_E = 1;
              flag_D = 1;
              flag_F = 1;
              flag_G = 1;
              flag_H = 1;
              flag_I = 1;
              flag_J = 1;
            }
            goto loop;
          }
        }
excite3:{
          if(flag_F == 1)
          {
            COUNTMAX = 15.0;
            k = 0;
            flag5 = 0;
            flag6 = 0;
            flag7 = 1;// <6>
            flag8 = 0;
            flag9 = 0;
            flag12 = 0;
            flag13 = 0;
            gotoxy(2,19);textcolor(14);
            cprintf("< >EXCITATION    ");
            gotoxy(3,19);textcolor(14);
            cprintf("6");
            num = 6;
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
            gotoxy(13,21);textcolor(15);
            cprintf("             ");
            gotoxy(21,21);textcolor(10);
            cprintf("D.A.");
            flag_F = 0;
            flag_D = 1;
            flag_E = 1;
            flag_G = 1;
            flag_H = 1;
            flag_I = 1;
            flag_J = 1;

            rr = 0;
            OL = 0.0;
            L_T = 0.0;
            qq = 0;
            ii = 0.0;

            goto loop;
          }
          else
          if(flag_F == 0)
          {
            if(flag7 == 1)
            {
              flag7 = 0;
              gotoxy(16,19);textcolor(12+128);
              cprintf("OFF");
              flag_F = 1;
              flag_D = 1;
              flag_E = 1;
              flag_G = 1;
              flag_H = 1;
              flag_I = 1;
              flag_J = 1;
            }
            goto loop;
          }
        }
excite4:{
          if(flag_G == 1)
          {
            COUNTMAX = 15.0;
            k = 0;
            flag5 = 0;
            flag6 = 0;
            flag7 = 0;
            flag8 = 1;// <7>
            flag9 = 0;
            flag12 = 0;
            flag13 = 0;
            gotoxy(2,19);textcolor(14);
            cprintf("< >EXCITATION     ");
            gotoxy(3,19);textcolor(14);
            cprintf("7");
            num = 7;
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
            gotoxy(13,21);textcolor(15);
            cprintf("            ");
            gotoxy(21,21);textcolor(10);
            cprintf("D.A.");
            flag_G = 0;
            flag_D = 1;
            flag_E = 1;
            flag_F = 1;
            flag_H = 1;
            flag_I = 1;
            flag_J = 1;

            rr = 0;
            OL = 0.0;
            L_T = 0.0;
            qq = 0;
            ii = 0.0;

            goto loop;
          }
          else
          if(flag_G == 0)
          {
            if(flag8 == 1)
            {
              flag8 = 0;
              gotoxy(16,19);textcolor(12+128);
              cprintf("OFF");
              flag_G = 1;
              flag_D = 1;
              flag_E = 1;
              flag_F = 1;
              flag_H = 1;
              flag_I = 1;
              flag_J = 1;
            }
            goto loop;
          }
        }
excite5:{
          if(flag_H == 1)
          {
            COUNTMAX = 15.0;
            k1 = 1;
            flag5 = 0;
            flag6 = 0;
            flag7 = 0;
            flag8 = 0;
            flag9 = 1;// <8>
            flag12 = 0;
            flag13 = 0;
            gotoxy(2,19);textcolor(14);
            cprintf("< >EXCITATION    ");
            gotoxy(3,19);textcolor(14);
            cprintf("8");
            num = 8;
            gotoxy(16,19);textcolor(10);
            cprintf("ON ");
            gotoxy(13,21);textcolor(15);
            cprintf("PW: %6.4f  ",PW);
            flag_H = 0;
            flag_D = 1;
            flag_E = 1;
            flag_F = 1;
            flag_G = 1;
            flag_I = 1;
            flag_J = 1;

            rr = 0;
            OL = 0.0;
            L_T = 0.0;
            qq = 0;
            ii = 0.0;

            goto loop;
          }
          else
          if(flag_H == 0)
          {
            if(flag9 == 1)
            {
              flag9 = 0;
              gotoxy(16,19);textcolor(12+128);
              cprintf("OFF");

              gotoxy(13,21);textcolor(10);
              cprintf("        D.A.");

              flag_H = 1;
              flag_D = 1;
              flag_E = 1;
              flag_F = 1;
              flag_G = 1;
              flag_I = 1;
              flag_J = 1;
            }
            goto loop;
          }
        }
excite6:{
          if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
```

```
{
  if(flag_I == 1)
  {
    COUNTMAX = 15.0;
    k = 0;
    flag5 = 0;
    flag6 = 0;
    flag7 = 0;
    flag8 = 0;
    flag9 = 0;
    flag12 = 1;// <9>
    flag13 = 0;
    gotoxy(2,19);textcolor(14);
    cprintf("< >EXCITATION    ");
    gotoxy(3,19);textcolor(14);
    cprintf("9");
    num = 9;
    gotoxy(16,19);textcolor(10);
    cprintf("ON ");
    gotoxy(13,21);textcolor(15);
    cprintf("               ");
    gotoxy(21,21);textcolor(10);
    cprintf("D.A.");
    flag_I = 0;
    flag_D = 1;
    flag_E = 1;
    flag_F = 1;
    flag_G = 1;
    flag_H = 1;
    flag_J = 1;

    rr = 0;
    OL = 0.0;
    L_T = 0.0;
    qq = 0;
    ii = 0.0;

    goto loop;
  }
  else
  if(flag_I == 0)
  {
    if(flag12 == 1)
    {
      flag12 = 0;
      gotoxy(16,19);textcolor(12+128);
      cprintf("OFF");
      flag_I = 1;
      flag_D = 1;
      flag_E = 1;
      flag_F = 1;
      flag_G = 1;
      flag_H = 1;
      flag_J = 1;
    }
    goto loop;
  }
}//End of if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
```

```
          goto loop;
        }
excite7:{
          if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
          {
            if(flag_J == 1)
            {
              COUNTMAX = 15.0;
              k1 = 1;
              flag5 = 0;
              flag6 = 0;
              flag7 = 0;
              flag8 = 0;
              flag9 = 0;
              flag12 = 0;
              flag13 = 1;// <0>
              gotoxy(2,19);textcolor(14);
              cprintf("< >EXCITATION    ");
              gotoxy(3,19);textcolor(14);
              cprintf("0");
              num = 0;
              gotoxy(16,19);textcolor(10);
              cprintf("ON ");
              gotoxy(13,21);textcolor(15);
              cprintf("            ");
              gotoxy(21,21);textcolor(10);
              cprintf("D.A.");
              flag_J = 0;
              flag_D = 1;
              flag_E = 1;
              flag_F = 1;
              flag_G = 1;
              flag_H = 1;
              flag_I = 1;

              rr = 0;
              OL = 0.0;
              L_T = 0.0;
              qq = 0;
              ii = 0.0;

              goto loop;
            }

            else

            if(flag_J == 0)
            {
              if(flag13 == 1)
              {
                flag13 = 0;
                gotoxy(16,19);textcolor(12+128);
                cprintf("OFF");
                flag_J = 1;
                flag_D = 1;
                flag_E = 1;
                flag_F = 1;
                flag_G = 1;
                flag_H = 1;
                flag_I = 1;
              }
              goto loop;
            }
          }// if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
          goto loop;
        }
modal:{
        rr = 0;
        OL = 0.0;
        L_T = 0.0;
        qq = 0;
        ii = 0.0;
        COUNTMAX = 15.0;

        if(diag == 1)
        {
          flag44 = 0;
          gotoxy(26,20);textcolor(15);
          cprintf("         ");// Erase "Force (N)"
          gotoxy(25,21);textcolor( 4);
          cprintf("           ");// Erasr "==========="
          gotoxy(42,12);textcolor(4);
          cprintf("                   ");// Erase "==================="
          gotoxy(42,15);textcolor(14);
          cprintf("                   ");// Erase "==================="
          gotoxy(22,22);textcolor(15);
          cprintf("  ");// Erase "x:"
          gotoxy(22,23);textcolor(15);
          cprintf("                           ");// Erase "y:"
          gotoxy(42,13);
          cprintf("                   ");// Erase "kh_bot<g>"
          gotoxy(42,14);
          cprintf("                   ");// Erase "dh_bot<d>"
          gotoxy(21,17);// Erase "offset_bot<t>"
          cprintf("                                        ");
          gotoxy(21,18);// Erase "offset_bot<w>"
          cprintf("                                        ");
          gotoxy(21,19);// Erase "bias_current_bot<b>"
          cprintf("                                              ");
          gotoxy(21,13);textcolor(11);
          cprintf("k_tilt      :");
          gotoxy(34,13);textcolor(15);
          cprintf("%6.2f", k_tilt);
          gotoxy(21,14);textcolor(11);
          cprintf("c_tilt      :");
          gotoxy(34,14);textcolor(15);
          cprintf("%6.2f",c_tilt);
          gotoxy(65, 9);textcolor(15);
          cprintf("l");
          gotoxy(65,10);textcolor(15);
          cprintf("u");
          gotoxy(65,11);textcolor(15);
          cprintf("z");
          gotoxy(61,21);textcolor(15);
          cprintf("( )");
          if(flag_GG == 1)
        {
          COUNTMAX = 15.0;
          flag10 = 1;
          gotoxy(52,5);textcolor(15);
          cprintf("==>                    <==");
          gotoxy(56,5);textcolor(14+128);
          cprintf("MODAL CONTROLLER");
          if(lu == 'l')
          {
            gotoxy(62,21);textcolor(15+128);
            cprintf("L");
          }
          else
          if(lu == 'u')
          {
            gotoxy(62,21);textcolor(15+128);
            cprintf("U");
          }
          gotoxy(16,18);textcolor(10);
          cprintf("ON ");
          gotoxy(25,22);
          cprintf("            ");// Erase x: along with output value
          flagJJ = 0;// Initialize toggle to "TILT MODE"
          flag_GG = 0;// Toggle condition
          goto loop;
        }
        else
        if(flag_GG == 0)
        {
          lu = 'l';
          flag10 = 0;
          flag15 = 1;
          gotoxy(16,18);textcolor(12+128);
          cprintf("OFF");
          gotoxy(57,5);
          cprintf("                           ");// Erase "MODAL CONTROLLER"
          gotoxy(52,5);textcolor(14+128);
          cprintf("==>                    <==");
          gotoxy(57,5);textcolor(10);
          cprintf("LOWER  BEARING");
          gotoxy(65, 9);textcolor(15+128);
          cprintf("l");
          gotoxy(61,21);textcolor(15);
          cprintf("    ");// Erase (L) & (U)
          gotoxy(31,8);textcolor(9);
          cprintf(" <c>CG factor: %5.2f",CG);
          gotoxy(21,13);textcolor(9);
          cprintf("kv_bot<p>    :%6.2f",kv_bot);
          gotoxy(42,13);textcolor(9);
          cprintf("kh_bot<g>    :%6.2f",kh_bot);
          gotoxy(21,14);textcolor(9);
          cprintf("dv_bot<v>    :%6.2f",dv_bot);
          gotoxy(42,14);textcolor(9);
          cprintf("dh_bot<d>    :%6.2f",dh_bot);
          gotoxy(21,17);textcolor(9);
          cprintf("offset_bot<t>                         :");
          gotoxy(55,17);textcolor(9);
          cprintf("%5d",tBias_bot);
```

```
            gotoxy(21,18);textcolor(9);
            cprintf("offset_bot<w>                      :");
            gotoxy(55,18);textcolor(9);
            cprintf("%5d",wBias_bot);
            gotoxy(21,19);textcolor(9);
            cprintf("offset current_bot<b>              :");
            gotoxy(55,19);textcolor(9);
            cprintf("%6.2f Amp.",        ibias_bot);
            gotoxy(26,20);textcolor(15);
            cprintf("Force (N)");
            gotoxy(25,21);textcolor( 4);
            cprintf("===========");
            gotoxy(51,20);textcolor(15);
            cprintf("x_value         y_value");
            gotoxy(51,21);textcolor(4);
            cprintf("=======         =======");
            if(nw_bot == 1)
            {
              gotoxy(22,22);textcolor(15);
              cprintf("x:");
              gotoxy(22,23);textcolor(15);
              cprintf("y:");
            }
            gotoxy(49,24);textcolor(15);
            cprintf("  +       -       +       -  ");
            gotoxy(49,25);textcolor(15);
            cprintf("  X       X       Y       Y  ");
            gotoxy(19,11);textcolor(15);
            cprintf("         Y_AXIS                    X_AXIS");
            gotoxy(36,11);textcolor(13);
            cprintf("< >-test: %1u",test_signal);
            gotoxy(37,11);textcolor(15);
            cprintf("M");
            gotoxy(21,12);textcolor(4);
            cprintf("===================  ===================");
            gotoxy(21,15);textcolor(14);
            cprintf("===================  ===================");

            flag15 = 1;
            flag11 = 1;// Lower bearing write out block activated
            flag22 = 0;// Upper bearing write out block deactivated
            flag33 = 0;// Thrust bearing write out block deactivated
            flag23 = 1;// Enable key press "9 & 0"
            flag_GG = 1;

            goto loop;
          }// End of if(flag_GG == 0)
        }// End (diag == 1)
        else
        if(diag == 0)
        {
          if(flag_GG == 1)
          {
            flag10 = 1;
            gotoxy(52,5);textcolor(15+128);
            cprintf("==>                   <==");
            gotoxy(56,5);textcolor(14);
            cprintf("MODAL CONTROLLER");
            gotoxy(16,18);textcolor(10);
            cprintf("ON ");
            flag_GG = 0;// Toggle condition
            goto loop;
          }
          else
          if(flag_GG == 0)
          {
            flag10 = 0;
            gotoxy(52,5);// Erase("==>                  <==")
            cprintf("                       ");
            gotoxy(16,18);textcolor(12+128);
            cprintf("OFF");
            flag_GG = 1;// Toggle condition
            goto loop;
          }
        }
        goto loop;
      }
disable_safe1:{
                sg1 = 0;
                gotoxy(16,15);textcolor(12+128);
                cprintf("OFF");
                goto loop;
              }
enable_safe1:{
               sg1 = 1;
               gotoxy(16,15);textcolor(10);
               cprintf("ON ");
               goto loop;
             }
disable_safe2:{
                sg2 = 0;
                gotoxy(16,16);textcolor(12+128);
                cprintf("OFF");
                goto loop;
              }
enable_safe2:{
               sg2 = 1;
               gotoxy(16,16);textcolor(10);
               cprintf("ON ");
               goto loop;
             }
disable_safe3:{
                sg3 = 0;
                gotoxy(16,17);textcolor(12+128);
                cprintf("OFF");
                goto loop;
              }
enable_safe3:{
               sg3 = 1;
               gotoxy(16,17);textcolor(10);
               cprintf("ON ");
               goto loop;
             }
cg_factor_up:{
               CG = CG + 0.01;
               if(CG > 0.5)
```

```
                  CG = 0.5;
               MCG = 0.5 - CG;
               PCG = 0.5 + CG;
               gotoxy(46,8);textcolor(15);
               cprintf("%5.2f", CG);
               goto loop;
             }
cg_factor_down:{
                CG = CG - 0.01;
                if(CG < -0.5)
                   CG = -0.5;
                MCG = 0.5 - CG;
                PCG = 0.5 + CG;
                gotoxy(46,8);textcolor(15);
                cprintf("%5.2f", CG);
                goto loop;
               }
igain_up:{
          if(flag3 == 1 && flag15 == 0)
          {
            igainth = igainth + 0.0001;
            gotoxy(44,8);textcolor(15);
            cprintf("%7.4f", igainth);
            goto loop;
          }
          goto loop;
        }
igain_down:{
             if(flag3 == 1 && flag15 == 0)
             {
               igainth = igainth - 0.0001;
               gotoxy(44,8);
               printf("%7.4f", igainth);
               goto loop;
             }
             goto loop;
           }
buffer:{
        if(flag_FF == 1)// Toggle flag
        {
          flag4d = 1;// Buffer on
          gotoxy(45,16);textcolor(10);
          cprintf("ON ");
          flag_FF = 0;
          goto loop;
        }
        else
        if(flag_FF == 0)// Toggle flag
        {
          flag4d = 0;// Buffer off
          gotoxy(45,16);textcolor(12+128);
          cprintf("OFF");
          flag_FF = 1;
          goto loop;
        }
       }
diagnostic:{
             gotoxy(37,19);textcolor(14);
```

```
cprintf("        ");// Erase NASA, GLENN, RESEARCH, CENTER
gotoxy(10,21);
cprintf
("                                                           ");
O      = 1.0;
flag5  = 0;// |
flag6  = 0;// |
flag7  = 0;// |
flag8  = 0;// |  Shut down excitor functions.
flag9  = 0;// |
flag12 = 0;// |
flag13 = 0;// |
flag16 = 1;// Assembly condition (on)
flag18 = 0;
flag21 = 0;// Excitation switch
flag10 = 0;// Turn off modal block
flag44 = 0;// Enable D.A./I.A. display

flag_A = 1;// Assembly toggle set to on
flag_B = 1;// f_excite toggle set to on
flag_C = 1;// Excitation toggle set to on
flag_D = 1;
flag_E = 1;
flag_F = 1;
flag_G = 1;
flag_H = 1;
flag_I = 1;
flag_J = 1;
flag_M = 1;
flag_N = 1;
flagKK = 1;
flag_II = 0;

flag4a = 0;// Shuts down Lower bearing buffer
flag4b = 0;// Shuts down Upper bearing buffer
flag4c = 0;// Shuts down Thrust bearing buffer

rr = 0;
OL = 0.0;
L_T = 0.0;
qq = 0;
ii = 0.0;
diag = 1;
SSS   = 1;// <---- Condition necessary to access
         //       diagnostic parameter controls.
flag_GG = 1;

COUNTMAX = 15.0;
gotoxy(31,8);textcolor(9);
cprintf(" <c>CG factor: %5.2f",CG);
gotoxy(32,16);textcolor(14);
cprintf("[loop buffer    ]");

if(flag4d == 1)
{
  gotoxy(45,16);textcolor(10);
  cprintf("ON ");
}
```

```
else
if(flag4d == 0)
{
  gotoxy(45,16);textcolor(12+128);
  cprintf("OFF");
}
gotoxy(30,2);
printf("                      ");// Erase DT
gotoxy(23,13);
cprintf("                               ");// Erase LBE
gotoxy(23,14);
cprintf("                               ");// Erase UBE
gotoxy(23,15);
cprintf("                               ");// Erase TBE
gotoxy(48,2);textcolor(12);
cprintf("  * Thrst bearing is energized  !");
gotoxy(48,3);textcolor(12);
cprintf("  * Upper bearing is energized  !");
gotoxy(48,4);textcolor(12);
cprintf("  * Lower bearing is energized  !");
gotoxy(52,5);textcolor(14+128);
cprintf("==>                  <==");
gotoxy(57,5);textcolor(10);
cprintf("LOWER  BEARING");
gotoxy(1,1);textcolor(15);
cprintf ("<+,-> to toggle input-output writes");
gotoxy(1,2);textcolor(15);
cprintf("<q> to abort control");
gotoxy(1,3);textcolor(15);
cprintf("<f> to toggle loop time buffer");
gotoxy(1,4);textcolor(15);
cprintf("<e> non diagnostic       ");
gotoxy(1,5);textcolor(15);
cprintf("<!,@,#> Disable safe gain  ");
gotoxy(19,11);textcolor(15);
cprintf("         Y_AXIS               X_AXIS");
gotoxy(36,11);textcolor(13);
cprintf("< >-test: %1u",test_signal);
gotoxy(37,11);textcolor(15);
cprintf("M");
gotoxy(21,12);textcolor(4);
cprintf("==================  ===================");
gotoxy(21,15);textcolor(14);
cprintf("==================  ===================");
gotoxy(21,13);textcolor(9);
cprintf("kv_bot<p>   :%6.2f",kv_bot);
gotoxy(42,13);textcolor(9);
cprintf("kh_bot<g>   :%6.2f",kh_bot);
gotoxy(21,14);textcolor(9);
cprintf("dv_bot<v>   :%6.2f",dv_bot);
gotoxy(42,14);textcolor(9);
cprintf("dh_bot<d>   :%6.2f",dh_bot);
gotoxy(21,17);textcolor(9);
cprintf("offset_bot<t>                      :");
gotoxy(55,17);textcolor(9);
cprintf("%5d",tBias_bot);
gotoxy(21,18);textcolor(9);
cprintf("offset_bot<w>                      :");
```

```
               gotoxy(55,18);textcolor(9);
               cprintf("%5d",wBias_bot);
               gotoxy(21,19);textcolor(9);
               cprintf("bias current_bot<b>              : ");
               gotoxy(55,19);textcolor(9);
               cprintf("%6.2f Amp.",        ibias_bot);
               gotoxy(51,20);textcolor(15);
               cprintf("x_value         y_value");
               gotoxy(51,21);textcolor(4);
               cprintf("=======         =======");
               gotoxy(49,24);
               textcolor(15);
               cprintf("  +       -       +       -  ");
               gotoxy(49,25);
               textcolor(15);
               cprintf("  X       X       Y       Y  ");
               gotoxy(64, 7);textcolor(11);cprintf("Display Parameter");
               gotoxy(64, 8);textcolor(15);cprintf("=================");
               gotoxy(64, 9);textcolor(13);cprintf("< >Lower Bearing");
               gotoxy(65, 9);textcolor(15);cprintf("l");
               gotoxy(64,10);textcolor(13);cprintf("< >Upper Bearing");
               gotoxy(65,10);textcolor(15);cprintf("u");
               gotoxy(64,11);textcolor(13);cprintf("< >Thrst Bearing");
               gotoxy(65,11);textcolor(15);cprintf("z");
               gotoxy(64,13);textcolor(11);cprintf("Energizing Parmtr");
               gotoxy(64,14);textcolor(15);cprintf("=================");
               gotoxy(64,15);textcolor(13);cprintf("<H>Thrst Bearing");
               gotoxy(65,15);textcolor(15);cprintf("H");
               gotoxy(64,16);textcolor(13);cprintf("< >Upper Bearing");
               gotoxy(65,16);textcolor(15);cprintf("I");
               gotoxy(64,17);textcolor(13);cprintf("< >Lower Bearing");
               gotoxy(65,17);textcolor(15);cprintf("J");
               gotoxy(2,18);textcolor(14);
               cprintf("< >MODAL CTRL    ");
               gotoxy(3,18);textcolor(15+128);
               cprintf("m");
               gotoxy(16,18);textcolor(12+128);
               cprintf("OFF");
               gotoxy(2,19);textcolor(14);
               cprintf("< >EXCITATION     ");
               gotoxy(2,19);textcolor(14);
               cprintf("<%u>EXCITATION     ",num);
               gotoxy(16,19);textcolor(12+128);
               cprintf("OFF");
               gotoxy(26,20);textcolor(15);
               cprintf("Force (N)");
               gotoxy(25,21);textcolor(4 );
               cprintf("==========");
               gotoxy(1,20);textcolor(15);
               cprintf("<x>Frq_inpt:%7.2f Hz.",freq);
               gotoxy(1,25);textcolor(15);
               cprintf("<s>to adjust Pulse Width");
               gotoxy(27,24);textcolor(14);
               cprintf("[<,> Enable exction.]");
               gotoxy(28,25);textcolor(14);
               cprintf("[<:> Assembly    ]");
               gotoxy(42,25);textcolor(10);
               cprintf("ON");
```

```
             goto loop;
           }
non_diagnostic:{
                 clrscr();
                 O      = 1.0;
                 flag5  = 0;// |
                 flag6  = 0;// |
                 flag7  = 0;// |
                 flag8  = 0;// |  Shut down excitor functions.
                 flag9  = 0;// |
                 flag12 = 0;// |
                 flag13 = 0;// |
                 flag16 = 1;// Assembly condition (on)
                 flag10 = 0;// Turn off modal block

                 flag_A = 1;// Assembly toggle set to on
                 flag_B = 1;// f_excite toggle set to on
                 flag_C = 1;// Excitation toggle set to on
                 flag_D = 1;
                 flag_E = 1;
                 flag_F = 1;
                 flag_G = 1;
                 flag_H = 1;
                 flag_I = 1;
                 flag_J = 1;
                 flag_M = 1;
                 flag_N = 1;
                 flagKK = 1;

                 flag_II = 0;
                 flag_CC = 0;
                 flag_DD = 0;
                 flag_EE = 0;

                 flag18 = 0;
                 flag21 = 0;// Excitation switch
                 flag4  = 0;
                 SSS    = 0;
                 flag_GG = 1;

                 COUNTMAX = 15.0;
                 gotoxy(1,1);textcolor(15);
                 cprintf("<x/k> to adjust frequency");
                 gotoxy(1,2);textcolor(15);
                 cprintf("<q> to abort control");
                 gotoxy(1,3);textcolor(15);
                 cprintf("<m> to toggle modal cntrl");
                 gotoxy(1,4);textcolor(15);
                 cprintf("<?> to toggle f_excite");
                 gotoxy(1,5);textcolor(15);
                 cprintf("<4-0> to select excitation");
                 gotoxy(59,1);textcolor(15);
                 cprintf("[ file :  FiveAx.c   ]");
                 gotoxy(23,14);
                 cprintf("                                  ");// Erase TBF
                 gotoxy(31,2);textcolor(11);
                 cprintf("DIAGNOSTIC TOGGLE<E>");
                 gotoxy(31,8);textcolor(9);
```

```
cprintf(" <c>CG factor: %5.2f",CG);
gotoxy(27, 9);textcolor(10);
cprintf("[ loop time:       micro-sec ]");
gotoxy(1,8);textcolor(15);cprintf("[   THE MAGNETIC  ]");
gotoxy(1,9);textcolor(15);cprintf("[BEARING SYSTEM IS]");
gotoxy(1,10);textcolor(15);cprintf("[                 ]");
gotoxy(9,11);textcolor(15);cprintf("|");
gotoxy(9,12);textcolor(15);cprintf("|");
gotoxy(4,10);textcolor(12+128);
cprintf("OPERATIONAL !  ");
gotoxy(26,13);textcolor(14);
cprintf("==>                         <==");
gotoxy(30,13);textcolor(12+128);
cprintf("THRST BEARING ENERGIZED");
gotoxy(26,14);textcolor(14);
cprintf("==>                         <==");
gotoxy(30,14);textcolor(12+128);
cprintf("UPPER BEARING ENERGIZED");
gotoxy(26,15);textcolor(14);
cprintf("==>                         <==");
gotoxy(30,15);textcolor(12+128);
cprintf("LOWER BEARING ENERGIZED");
nw_bot = 0;
nw_top = 0;
nw_th  = 0;
gotoxy(1,22);textcolor(13);
cprintf("<Excitation Parmtr>");
gotoxy(1,14);textcolor(10);
cprintf("< >PHSE ANG:%3u deg",th);
gotoxy(2,14);textcolor(15);
cprintf("n");
gotoxy(1,24);textcolor(15);
cprintf("<a>Amplitude:%4.1f v 0-pk",volt);
gotoxy(1,20);textcolor(15);
cprintf("<x>Frq_inpt:%7.2f Hz.",freq);
gotoxy(1,25);textcolor(15);
cprintf("<s>to adjust Pulse Width");
gotoxy(48,22);
printf("                                ");
gotoxy(46,23);
printf("                                   ");

diag  = 0;

flag1 = 1;// Lower bearing block activated
flag2 = 1;// Upper bearing block activated
flag3 = 1;// Thrust bearing block activated

flag11 = 1;
flag22 = 1;
flag33 = 1;
flag44 = 0;// Enable D.A./I.A. display

sg1 = 1;
sg2 = 1;
sg3 = 1;

gotoxy(1,15);textcolor(15);
```

```
                  cprintf("[                 ]");
                  gotoxy(2,15);textcolor(14);
                  cprintf("Lwr Safe Gain   ");
                  gotoxy(16,15);textcolor(10);
                  cprintf("ON ");
                  gotoxy(1,16);textcolor(15);
                  cprintf("[                 ]");
                  gotoxy(2,16);textcolor(14);
                  cprintf("Upr Safe Gain   ");
                  gotoxy(16,16);textcolor(10);
                  cprintf("ON ");
                  gotoxy(1,17);textcolor(15);
                  cprintf("[                 ]");
                  gotoxy(2,17);textcolor(14);
                  cprintf("Tht Safe Gain   ");
                  gotoxy(16,17);textcolor(10);
                  cprintf("ON ");
                  gotoxy(1,18);textcolor(15);
                  cprintf("[                 ]");
                  gotoxy(2,18);textcolor(14);
                  cprintf("MODAL CNTRL     ");
                  gotoxy(16,18);textcolor(12+128);
                  cprintf("OFF");
                  gotoxy(1,19);textcolor(15);
                  cprintf("[                 ]");
                  gotoxy(2,19);textcolor(14);
                  cprintf("<%u>EXCITATION    ",num);
                  gotoxy(16,19);textcolor(12+128);
                  cprintf("OFF");
                  gotoxy(27,24);textcolor(14);
                  cprintf("[<,> Enable exction.]");
                  gotoxy(28,25);textcolor(14);
                  cprintf("[<:> Assembly    ]");
                  gotoxy(42,25);textcolor(10);
                  cprintf("ON");
                  goto loop;
                }
lower_bearing:{
                 if(flag_CC == 1)
                 {
                   gotoxy(48,4);textcolor(12);
                   cprintf("  * Lower bearing is energized  !");
                   gotoxy(65,17);textcolor(15);cprintf("J");
                   flag1  = 1;
                   flag4a = 0;// Shuts down Lower bearing buffer
                   flag_CC = 0;
                   goto loop;
                 }
                 else
                 if(flag_CC == 0)
                 {
                   gotoxy(52,4);textcolor(14+128);
                   cprintf("Lower bearing not energized !");
                   gotoxy(65,17);textcolor(15+128);cprintf("J");
                   flag1 = 0;
                   flag4a = 1;// Turn on Lower bearing buffer
                   flag_CC = 1;
                   goto loop;
                }
              }
upper_bearing:{
                if(flag_DD == 1)
                {
                  gotoxy(48,3);textcolor(12);
                  cprintf("  * Upper bearing is energized  !");
                  gotoxy(65,16);textcolor(15);cprintf("I");
                  flag2  = 1;
                  flag4b = 0;// Shuts down Upper bearing buffer
                  flag_DD = 0;
                  goto loop;
                }
                else
                if(flag_DD == 0)
                {
                  gotoxy(52,3);textcolor(14+128);
                  cprintf("Upper bearing not energized !");
                  gotoxy(65,16);textcolor(15+128);cprintf("I");
                  flag2  = 0;
                  flag4b = 1;// Turn on Upper bearing buffer
                  flag_DD = 1;
                  goto loop;
                }
              }
thrust_bearing:{
                if(flag_EE == 1)
                {
                  gotoxy(48,2);textcolor(12);
                  cprintf("  * Thrst bearing is energized  !");
                  gotoxy(65,15);textcolor(15);cprintf("H");
                  flag3  = 1;
                  flag4c = 0;// Shuts down Thrust bearing buffer
                  flag_EE = 0;
                  goto loop;
                }
                else
                if(flag_EE == 0)
                {
                  gotoxy(52,2);textcolor(14+128);
                  cprintf("Thrst bearing not energized !");
                  gotoxy(65,15);textcolor(15+128);cprintf("H");
                  flag3  = 0;
                  flag4c = 1;// Surn on Thrust bearing buffer
                  flag_EE = 1;
                  goto loop;
                }
              }
l_on:{
       if(flag10 == 0)// Disable this block when in modal mode
       {
         gotoxy(31,8);textcolor(9);
         cprintf(" <c>CG factor: %5.2f",CG);
         gotoxy(52,5);textcolor(14+128);
         cprintf("==>                  <==");
         gotoxy(57,5);textcolor(10);
         cprintf("LOWER  BEARING");
         gotoxy(21,13);textcolor(9);
```

```
    cprintf("kv_bot<p>   :%6.2f",kv_bot);
    gotoxy(42,13);textcolor(9);
    cprintf("kh_bot<g>   :%6.2f",kh_bot);
    gotoxy(21,14);textcolor(9);
    cprintf("dv_bot<v>   :%6.2f",dv_bot);
    gotoxy(42,14);textcolor(9);
    cprintf("dh_bot<d>   :%6.2f",dh_bot);
    gotoxy(21,17);textcolor(9);
    cprintf("offset_bot<t>                      :");
    gotoxy(55,17);textcolor(9);
    cprintf("%5d",tBias_bot);
    gotoxy(21,18);textcolor(9);
    cprintf("offset_bot<w>                      :");
    gotoxy(55,18);textcolor(9);
    cprintf("%5d",wBias_bot);
    gotoxy(21,19);textcolor(9);
    cprintf("bias current_bot<b>              :");
    gotoxy(55,19);textcolor(9);
    cprintf("%6.2f Amp.",        ibias_bot);
    gotoxy(26,20);textcolor(15);
    cprintf("Force (N)");
    gotoxy(25,21);textcolor( 4);
    cprintf("===========");
    if(nw_bot == 1)
    {
      gotoxy(22,22);textcolor(15);
      cprintf("x:");
      gotoxy(22,23);textcolor(15);
      cprintf("y:");
    }
    gotoxy(51,20);textcolor(15);
    cprintf("x_value         y_value");
    gotoxy(51,21);textcolor(4);
    cprintf("=======         =======");
    gotoxy(27,23);
    cprintf("                      ");// Erase [<^> to toggle D.A. ]
    gotoxy(49,24);textcolor(15);
    cprintf("  +       -       +       -  ");
    gotoxy(49,25);textcolor(15);
    cprintf("  X       X       Y       Y  ");
    gotoxy(19,11);textcolor(15);
    cprintf("        Y_AXIS              X_AXIS");
    gotoxy(36,11);textcolor(13);
    cprintf("< >-test: %1u",test_signal);
    gotoxy(37,11);textcolor(15);
    cprintf("M");
    gotoxy(21,12);textcolor(4);
    cprintf("===================  ===================");
    gotoxy(21,15);textcolor(14);
    cprintf("===================  ===================");
    gotoxy(65,9);textcolor(15+128);
    cprintf("l");
    gotoxy(65,10);textcolor(15);
    cprintf("u");
    gotoxy(65,11);textcolor(15);
    cprintf("z");

    flag15 = 1;
```

```
       }// End of if(flag10 == 0)
       if(flag10 == 0 || flag10 == 1)
       {
         lu = 'l';
         if(nw_bot == 1)
         {
           gotoxy(37,22);textcolor(10);
           cprintf("Displacement:");
         }
         if(flag10 == 1)
         {
           gotoxy(62,21);textcolor(15+128);
           cprintf("L");
         }

         flag11 = 1;// Lower bearing write out block activated
         flag22 = 0;// Upper bearing write out block deactivated
         flag33 = 0;// Thrust bearing write out block deactivated
       }// End of if(flag10 == 0 || flag10 == 1)
       flag23 = 1;// Enable key press "9 & 0"
       goto loop;
     }
u_on:{
      if(flag10 == 0)// Disable this block when in modal mode
      {
        gotoxy(31,8);textcolor(9);
        cprintf(" <c>CG factor: %5.2f",CG);
        gotoxy(52,5);textcolor(14+128);
        cprintf("==>                  <==");
        gotoxy(57,5);textcolor(10);
        cprintf("UPPER  BEARING");
        gotoxy(21,13);textcolor(9);
        cprintf("kv_top<p>   :%6.2f",kv_top);
        gotoxy(42,13);textcolor(9);
        cprintf("kh_top<g>   :%6.2f",kh_top);
        gotoxy(21,14);textcolor(9);
        cprintf("dv_top<v>   :%6.2f",dv_top);
        gotoxy(42,14);textcolor(9);
        cprintf("dh_top<d>   :%6.2f",dh_top);
        gotoxy(21,17);textcolor(9);
        cprintf("offset_top<t>                      :");
        gotoxy(55,17);textcolor(9);
        cprintf("%5d",tBias_top);
        gotoxy(21,18);textcolor(9);
        cprintf("offset_top<w>                      :");
        gotoxy(55,18);textcolor(9);
        cprintf("%5d",wBias_top);
        gotoxy(21,19);textcolor(9);
        cprintf("bias current_top<b>              :");
        gotoxy(55,19);textcolor(9);
        cprintf("%6.2f Amp.",        ibias_top);
        gotoxy(26,20);textcolor(15);
        cprintf("Force  (N)");
        gotoxy(25,21);textcolor( 4);
        cprintf("===========");
        if(nw_top == 1)
        {
          gotoxy(22,22);textcolor(15);
          cprintf("x:");
          gotoxy(22,23);textcolor(15);
          cprintf("y:");
        }
        gotoxy(51,20);textcolor(15);
        cprintf("x_value        y_value");
        gotoxy(51,21);textcolor(4);
        cprintf("=======        =======");
        gotoxy(27,23);
        cprintf("                   ");// Erase [<^> to toggle D.A. ]
        gotoxy(49,24);textcolor(15);
        cprintf("  +       -       +       -  ");
        gotoxy(49,25);textcolor(15);
        cprintf("  X       X       Y       Y  ");
        gotoxy(19,11);textcolor(15);
        cprintf("        Y_AXIS             X_AXIS");
        gotoxy(36,11);textcolor(13);
        cprintf("< >-test: %1u",test_signal);
        gotoxy(37,11);textcolor(15);
        cprintf("M");
        gotoxy(21,12);textcolor(4);
        cprintf("==================  ==================");
        gotoxy(21,15);textcolor(14);
        cprintf("==================  ==================");
        gotoxy(65,9);textcolor(15);
        cprintf("l");
        gotoxy(65,10);textcolor(15+128);
        cprintf("u");
        gotoxy(65,11);textcolor(15);
        cprintf("z");

        flag15 = 1;
      }// End of if(flag10 == 0)
      if(flag10 == 0 || flag10 == 1)
      {
        lu = 'u';
        if(nw_top == 1)
        {
          gotoxy(37,22);textcolor(10);
          cprintf("Displacement:");
        }
        if(flag10 == 1)
        {
          gotoxy(62,21);textcolor(15+128);
          cprintf("U");
        }
        flag11 = 0;// Lower bearing write out block deactivated
        flag22 = 1;// Upper bearing write out activated
        flag33 = 0;// Thrust bearing write out block deactivated
      }// End of if(flag10 == 0 || flag10 == 1)
      flag23 = 1;// Enable key press "9 & 0"
      goto loop;
     }
z_on:{
       if(flag10 == 0)// Disable this block when in modal mode
       {
         gotoxy(31,8);textcolor(9);
         cprintf("<(,)>igainth:%7.4f", igainth);
        gotoxy(52,5);textcolor(14+128);
        cprintf("==>                  <==");
        gotoxy(57,5);textcolor(10);
        cprintf("THRUST BEARING");
        gotoxy(21,13);textcolor(9);
        cprintf("kv_th<p>    :%6.2f",kv_th);
        gotoxy(42,13);textcolor(9);
        cprintf("                   ");// Erase top right half
        gotoxy(21,14);textcolor(9);
        cprintf("dv_th<v>    :%6.2f",dv_th);
        gotoxy(42,14);textcolor(9);
        cprintf("                   ");// Erase bottom right half
        gotoxy(21,17);textcolor(9);
        cprintf("offset_th<t>                       :");
        gotoxy(55,17);textcolor(9);
        cprintf("%5d",tBias_th);
        gotoxy(21,18);textcolor(9);// Erase wBias_th
        cprintf("                                       ");
        gotoxy(21,19);textcolor(9);
        cprintf("bias current_th<b>               :");
        gotoxy(55,19);textcolor(9);
        cprintf("%6.2f Amp.", ibias_th);
        gotoxy(50,20);textcolor(15);
        cprintf(" z_value                 ");
        gotoxy(50,21);textcolor(4);
        cprintf(" =======                 ");
        gotoxy(26,20);textcolor(15);
        cprintf("Force (N)");
        gotoxy(25,21);textcolor( 4);
        cprintf("==========");
        gotoxy(24,22);// Erase x:
        printf("                                                  ");
        gotoxy(22,23);// Erase y:
        printf
        ("                                                            ");
        gotoxy(49,24);textcolor(15);
        cprintf("  +       -                  ");
        gotoxy(49,25);textcolor(15);
        cprintf("  Z       Z                  ");
        gotoxy(19,11);textcolor(15);
        cprintf("         Z_AXIS                      ");
        gotoxy(36,11);textcolor(13);
        cprintf("< >-test: %1u",test_signal);
        gotoxy(37,11);textcolor(15);
        cprintf("M");
        gotoxy(21,12);textcolor(4);
        cprintf("==================                     ");
        gotoxy(21,15);textcolor(14);
        cprintf("==================                     ");
        gotoxy(22,22);textcolor(15);
        cprintf("  ");// Erase "x:"
        if(nw_th == 1)
        {
          gotoxy(37,22);textcolor(10);
          cprintf("Displacement:");
          gotoxy(22,22);textcolor(15);
          cprintf("z:");
        }
```

```
         gotoxy(65,9);textcolor(15);
         cprintf("l");
         gotoxy(65,10);textcolor(15);
         cprintf("u");
         gotoxy(65,11);textcolor(15+128);
         cprintf("z");

         flag15 = 0;
         flag11 = 0;// Lower bearing write out block deactivated
         flag22 = 0;// Upper bearing write out block deactivated
         flag33 = 1;// Thrust bearing write out block activated
         flag23 = 0;// Disable key press "9 & 0"
         flag_GG = 1;

         goto loop;
       }// End of if(flag10 == 0)
       goto loop;
     }
kv_up:{
        if(flag10 == 0)
        {
          if(flag11 == 1)
          {
            kv_bot = kv_bot + 0.1;
            gotoxy(34,13);textcolor(15);
            cprintf("%6.2f", kv_bot);
            goto loop;
          }
          else
          if(flag22 == 1)
          {
            kv_top = kv_top + 0.1;
            gotoxy(34,13);textcolor(15);
            cprintf("%6.2f", kv_top);
            goto loop;
          }
          else
          if(flag33 == 1)
          {
            kv_th = kv_th + 0.1;
            gotoxy(34,13);textcolor(15);
            cprintf("%6.2f", kv_th);
            goto loop;
          }
        }// End of if(flag10 == 0)
        goto loop;
      }
kv_down:{
        if(flag10 == 0)
        {
          if(flag11 == 1)
          {
            kv_bot = kv_bot - 0.1;
            gotoxy(34,13); printf("%6.2f", kv_bot);
            goto loop;
          }
          else
          if(flag22 == 1)
```

```
           {
             kv_top = kv_top - 0.1;
             gotoxy(34,13); printf("%6.2f", kv_top);
             goto loop;
           }
           else
           if(flag33 == 1)
           {
             kv_th = kv_th - 0.1;
             gotoxy(34,13); printf("%6.2f", kv_th);
             goto loop;
           }
         }
         goto loop;
        }
dh_up:{
       if(flag10 == 0)
       {
         if(flag11 == 1)
         {
           dh_bot = dh_bot + 0.5;
           gotoxy(55,14);textcolor(15);
           cprintf("%6.2f", dh_bot);
           goto loop;
         }
         else
         if(flag22 == 1)
         {
           dh_top = dh_top + 0.5;
           gotoxy(55,14);textcolor(15);
           cprintf("%6.2f", dh_top);
           goto loop;
         }
       }
       goto loop;
      }
dh_down:{
          if(flag10 == 0)
          {
            if(flag11 == 1)
            {
              dh_bot = dh_bot - 0.5;
              gotoxy(55,14); printf("%6.2f", dh_bot);
              goto loop;
            }
            else
            if(flag22 == 1)
            {
              dh_top = dh_top - 0.5;
              gotoxy(55,14); printf("%6.2f", dh_top);
              goto loop;
            }
          }
          goto loop;
        }
kh_up:{
        if(flag10 == 0)
        {
          if(flag11 == 1)
          {
            kh_bot = kh_bot + 0.1;
            gotoxy(55,13);textcolor(15);
            cprintf("%6.2f", kh_bot);
            goto loop;
          }
          else
          if(flag22 == 1)
          {
            kh_top = kh_top + 0.1;
            gotoxy(55,13);textcolor(15);
            cprintf("%6.2f", kh_top);
            goto loop;
          }
        }
        goto loop;
      }
kh_down:{
          if(flag10 == 0)
          {
            if(flag11 == 1)
            {
              kh_bot = kh_bot - 0.1;
              gotoxy(55,13); printf("%6.2f", kh_bot);
              goto loop;
            }
            else
            if(flag22 == 1)
            {
              kh_top = kh_top - 0.1;
              gotoxy(55,13); printf("%6.2f", kh_top);
              goto loop;
            }
          }
          goto loop;
        }
dv_up:{
        if(flag10 == 0)
        {
          if(flag11 == 1)
          {
            dv_bot = dv_bot + 0.5;
            gotoxy(34,14);textcolor(15);
            cprintf("%6.2f", dv_bot);
            goto loop;
          }
          else
          if(flag22 == 1)
          {
            dv_top = dv_top + 0.5;
            gotoxy(34,14);textcolor(15);
            cprintf("%6.2f", dv_top);
            goto loop;
          }
          else
          if(flag33 == 1)
          {
            dv_th = dv_th + 0.5;
            gotoxy(34,14);textcolor(15);
            cprintf("%6.2f", dv_th);
            goto loop;
          }
        }
        goto loop;
      }
dv_down:{
          if(flag10 == 0)
          {
            if(flag11 == 1)
            {
              dv_bot = dv_bot - 0.5;
              gotoxy(34,14); printf("%6.2f", dv_bot);
              goto loop;
            }
            else
            if(flag22 == 1)
            {
              dv_top = dv_top - 0.5;
              gotoxy(34,14); printf("%6.2f", dv_top);
              goto loop;
            }
            else
            if(flag33 == 1)
            {
              dv_th = dv_th - 0.5;
              gotoxy(34,14); printf("%6.2f", dv_th);
              goto loop;
            }
          }
          goto loop;
        }
wBias_up:{
           if(flag10 == 0)
           {
             if(flag11 == 1)
             {
               wBias_bot = wBias_bot + 5;
               gotoxy(55,18);textcolor(15);
               cprintf("%5d", wBias_bot);
               goto loop;
             }
             else
             if(flag22 == 1)
             {
               wBias_top = wBias_top + 5;
               gotoxy(55,18);textcolor(15);
               cprintf("%5d", wBias_top);
               goto loop;
             }
           }
           goto loop;
         }
wBias_down:{
             if(flag10 == 0)
             {
               if(flag11 == 1)
               {
                 wBias_bot = wBias_bot - 5;
                 gotoxy(55,18);printf("%5d", wBias_bot);
                 goto loop;
               }
               else
               if(flag22 == 1)
               {
                 wBias_top = wBias_top - 5;
                 gotoxy(55,18);printf("%5d", wBias_top);
                 goto loop;
               }
             }
             goto loop;
           }
tBias_up:{
           if(flag10 == 0)
           {
             if(flag11 == 1)
             {
               tBias_bot = tBias_bot + 5;
               gotoxy(55,17);textcolor(15);
               cprintf("%5d", tBias_bot);
               goto loop;
             }
             else
             if(flag22 == 1)
             {
               tBias_top = tBias_top + 5;
               gotoxy(55,17);textcolor(15);
               cprintf("%5d", tBias_top);
               goto loop;
             }
             else
             if(flag33 == 1)
             {
               tBias_th = tBias_th + 5;
               gotoxy(55,17);textcolor(15);
               cprintf("%5d", tBias_th);
               goto loop;
             }
           }
           goto loop;
         }
tBias_down:{
             if(flag10 == 0)
             {
               if(flag11 == 1)
               {
                 tBias_bot = tBias_bot - 5;
                 gotoxy(55,17); printf("%5d", tBias_bot);
                 goto loop;
               }
               else
               if(flag22 == 1)
               {
                 tBias_top = tBias_top - 5;
                 gotoxy(55,17); printf("%5d", tBias_top);
                 goto loop;
               }
               else
               if(flag33 == 1)
               {
                 tBias_th = tBias_th - 5;
                 gotoxy(55,17); printf("%5d", tBias_th);
                 goto loop;
               }
             }
             goto loop;
           }
writeout:{
           if(flag_B == 1)
           {
             if(flag11 == 1)
             {
               gotoxy(37,22);textcolor(10+128);
               cprintf("Displacement:");
               if(flag10 == 0)
               {
                 gotoxy(22,22);textcolor(15+128);
                 cprintf("x:");
                 gotoxy(22,23);textcolor(15+128);
                 cprintf("y:");
               }
               nw_bot = 1;// Write out enabled
               gotoxy(27,23);
               cprintf("                     ");// Erase [<^> to toggle D.A. ]
               goto loop;
             }
             else
             if(flag22 == 1)
             {
               gotoxy(37,22);textcolor(10+128);
               cprintf("Displacement:");
               if(flag10 == 0)
               {
                 gotoxy(22,22);textcolor(15+128);
                 cprintf("x:");
                 gotoxy(22,23);textcolor(15+128);
                 cprintf("y:");
               }
               nw_top = 1;// Write out enabled
               gotoxy(27,23);
               cprintf("                     ");// Erase [<^> to toggle D.A. ]
               goto loop;
             }
             else
             if(flag33 == 1)
             {
               gotoxy(37,22);textcolor(10+128);
               cprintf("Displacement:");
               gotoxy(22,22);textcolor(15+128);
               cprintf("z:");
               nw_th = 1;// Write out enabled
               gotoxy(27,23);
               cprintf("                     ");// Erase [<^> to toggle D.A. ]
               goto loop;
             }
           }// End of if(flag_B == 1)
           goto loop;
         }
nowrite:{
          if(flag_B == 1)
          {
            if(flag11 == 1)
            {
              flag44 = 0;// Enable D.A/I.A. display
              nw_bot = 0;// Write out disabled
              gotoxy(22,22);
              printf("                                                       ");
              gotoxy(22,23);
              printf
              ("                                                        ");
              goto loop;
            }
            else
            if(flag22 == 1)
            {
              flag44 = 0;// Enable D.A/I.A. display
              nw_top = 0;// Write out disabled
              gotoxy(22,22);
              printf("                                                       ");
              gotoxy(22,23);
              printf
              ("                                                        ");
              goto loop;
            }
            else
            if(flag33 == 1)
            {
              flag44 = 0;// Enable D.A/I.A. display
              nw_th = 0;// Write out disabled
              gotoxy(22,22);
              printf("                                                       ");
              gotoxy(22,23);
              printf
              ("                                                        ");
              goto loop;
            }
          }// End of if(flag_B == 1)
          goto loop;
        }
bias_up:{
          if(flag10 == 0)
          {
            if(flag11 == 1)
            {
              ibias_bot = ibias_bot + 0.1;
              bias_current_bot = round1(ibias_bot * 2.0 * 204.8);
              gotoxy(55,19);textcolor(15);
              cprintf("%6.2f", ibias_bot);
              goto loop;
            }
            else
            if(flag22 == 1)
            {
              ibias_top = ibias_top + 0.1;
              bias_current_top = round1(ibias_top * 2.0 * 204.8);
              gotoxy(55,19);textcolor(15);
              cprintf("%6.2f", ibias_top);
              goto loop;
            }
            else
            if(flag33 == 1)
            {
              ibias_th = ibias_th + 0.1;
              bias_current_th = round1(ibias_th * 2.0 * 204.8);
              gotoxy(55,19);textcolor(15);
              cprintf("%6.2f", ibias_th);
              goto loop;
            }
          }
          goto loop;
        }
bias_down:{
            if(flag10 == 0)
            {
              if(flag11 == 1)
              {
                ibias_bot = ibias_bot - 0.1;
                bias_current_bot = round1(ibias_bot * 2.0 * 204.8);
                gotoxy(55,19);
                printf("%6.2f", ibias_bot);
                goto loop;
              }
              else
              if(flag22 == 1)
              {
                ibias_top = ibias_top - 0.1;
                bias_current_top = round1(ibias_top * 2.0 * 204.8);
                gotoxy(55,19); printf("%6.2f", ibias_top);
                goto loop;
              }
              else
              if(flag33 == 1)
              {
                ibias_th = ibias_th - 0.1;
                bias_current_th = round1(ibias_th * 2.0 * 204.8);
                gotoxy(55,19); printf("%6.2f", ibias_th);
                goto loop;
              }
            }
            goto loop;
          }

/*-----------------------RAMP DOWN WHILE SUPPORTED--------------------------*/

ramp_down:{
            gotoxy(10,6);textcolor(14);
            cprintf("CONTROL RAMPING DOWN........         ");

```

```
             outport(out_chan1_0, t48);
             outport(out_chan1_1, t48);
             outport(out_chan1_2, t48);
             outport(out_chan1_3, t48);
             outport(out_chan1_4, t48);
             outport(out_chan1_5, t48);

//           outport(out_chan2_0, t48);
             outport(out_chan2_1, t48);
             outport(out_chan2_2, t48);
             outport(out_chan2_3, t48);
             outport(out_chan2_4, t48);
             outport(out_chan2_5, t48);

             gotoxy(31,7);textcolor(14);
             cprintf(" ......COMPLETE !      ");

             gotoxy(1, 8);printf("[   THE MAGNETIC  ]");
             gotoxy(1, 9);printf("[BEARING SYSTEM IS]");
             gotoxy(1,10);printf("[                 ]");
             gotoxy(8,10);textcolor(10+128);
             cprintf("OFF !\a\a   ");
             if(diag == 0)
             {
               gotoxy(26,13);
               cprintf("                                 ");// ERASE LBE
               gotoxy(26,14);
               cprintf("                                 ");// ERASE UBE
               gotoxy(26,15);
               cprintf("                                 ");// ERASE TBE
             }
loop3:       gotoxy(1,13);textcolor(14);
             cprintf("                    ");// ERASE AYS

             gotoxy(3,13);textcolor(10);
             cprintf("CONTINUE(y/n)?:");

             resp = getch();

             if(resp == 'y')
             {
               if(diag == 0)
               {
                 gotoxy(26,13);textcolor(14);
                 cprintf("==>                          <==");
                 gotoxy(30,13);textcolor(12+128);
                 cprintf("LOWER BEARING ENERGIZED");
                 gotoxy(26,14);textcolor(14);
                 cprintf("==>                          <==");
                 gotoxy(30,14);textcolor(12+128);
                 cprintf("UPPER BEARING ENERGIZED");
                 gotoxy(26,15);textcolor(14);
                 cprintf("==>                          <==");
                 gotoxy(30,15);textcolor(12+128);
                 cprintf("THRST BEARING ENERGIZED");
               }
             }

```

```
            if (resp == 'y' || resp == 'Y')
            {
              gotoxy(10,6); printf("                           ");// CRD
              gotoxy(31,7); printf("                           ");// C
              gotoxy(3,13);printf("               ");// ERASE "CONTINUE?"
              gotoxy(1,8);textcolor(15);
              cprintf("[   THE MAGNETIC  ]");
              gotoxy(1,9);textcolor(15);
              cprintf("[BEARING SYSTEM IS]");
              gotoxy(1,10);textcolor(15);
              cprintf("[                 ]");
              gotoxy(4,10);textcolor(12+128);
              cprintf("OPERATIONAL !\a  ");
              flag_L = 0;
              goto loop;
            }
            else
            if (resp == 'n' || resp == 'N')
            goto loop2;
            goto loop3;
          }// End ramp_down:
loop2: textcolor(7);cprintf("\b");clrscr();
return(0);
}//                * * * End of main function * * *

float round1(float u)
{
  int g,v;
  float z;

  g = ceil(u);
  z = u + 0.5;

  if(g >= z)
     v = floor(u);
  else
     v = g;
  return (v);
}
```

REFERENCES

1. Johnson, Dexter; Brown, Gerald V.; and Mehmed, Oral: A Magnetic Suspension and Excitation System for Spin Vibration Testing of Turbomachinery Blades. NASA/TM—1998-206976 (AIAA Paper 98–1851), 1998. http://gltrs.grc.nasa.gov/GLTRS/
2. Hutton, David V.: Applied Mechanical Vibrations. McGraw-Hill, New York, NY, 1981.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	**2. REPORT DATE** April 2001	**3. REPORT TYPE AND DATES COVERED** Technical Memorandum

4. TITLE AND SUBTITLE A Comprehensive C++ Controller for a Magnetically Supported Vertical Rotor: Version 1.0	**5. FUNDING NUMBERS** WU–708–28–13–00
6. AUTHOR(S) Carlos R. Morrison	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration John H. Glenn Research Center at Lewis Field Cleveland, Ohio 44135–3191	**8. PERFORMING ORGANIZATION REPORT NUMBER** E–12632
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546–0001	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** NASA TM—2001-210701

11. SUPPLEMENTARY NOTES

Responsible person, Carlos R. Morrison, organization code 5930, 216–433–8447.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Categories: 31, 61, 62, and 63 Distribution: Standard Available electronically at http://gltrs.grc.nasa.gov/GLTRS This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 words)***

This manual describes the new FATMaCC (Five-Axis, Three-Magnetic-Bearing Control Code). The FATMaCC (pronounced “fat mak”) is a versatile control code that possesses many desirable features that were not available in previous in-house controllers. The ultimate goal in designing this code was to achieve full rotor levitation and control at a loop time of 50 µs. Using a 1-GHz processor, the code will control a five-axis system in either a decentralized or a more elegant centralized (modal control) mode at a loop time of 56 µs. In addition, it will levitate and control (with only minor modification to the input/output wiring) a two-axis and/or a four-axis system. Stable rotor levitation and control of any of the systems mentioned above are accomplished through appropriate key presses to modify parameters, such as stiffness, damping, and bias. A signal generation block provides 11 excitation signals. An excitation signal is then superimposed on the radial bearing *x*- and *y*-control signals, thus producing a resultant force vector. By modulating the signals on the bearing *x*-and *y*-axes with a cosine and a sine function, respectively, a radial excitation force vector is made to rotate 360° about the bearing geometric center. The rotation of the force vector is achieved manually by using key press or automatically by engaging the “one-per-revolution” feature. Rotor rigid body modes can be excited by using the excitation module. Depending on the polarities of the excitation signal in each radial bearing, the bounce or tilt mode will be excited.

14. SUBJECT TERMS C++ controller; Controller; Five-axis controller; Magnetic bearing controller; Thrust bearing; FATMaCC	**15. NUMBER OF PAGES** 130 **16. PRICE CODE**

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**