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

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

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

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.

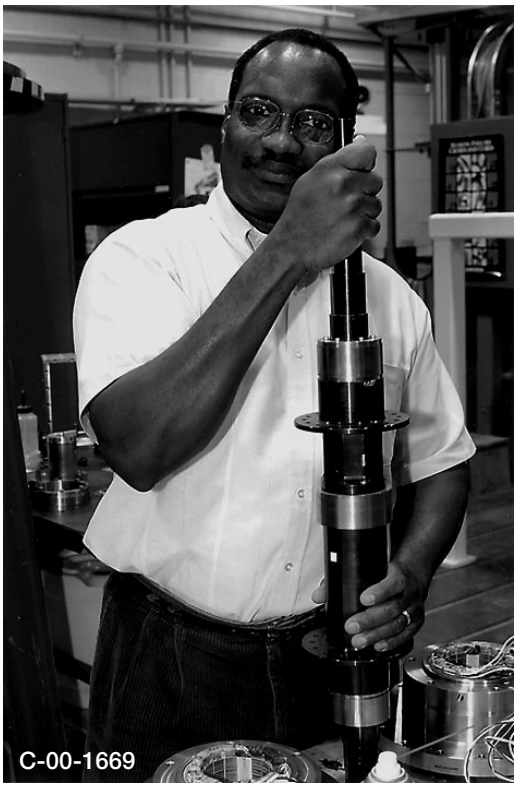


Figure 2.—Rotor juxtapositition for size comparison.

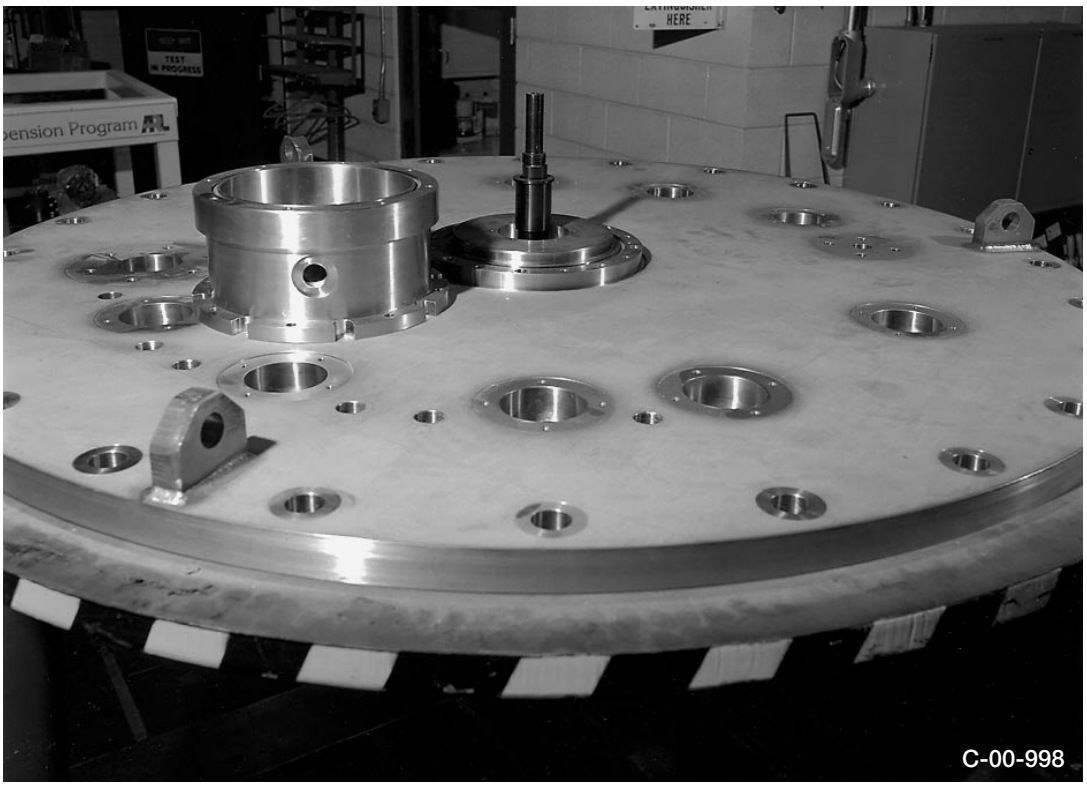


Figure 3.—Top view of rotor without thrust bearing assembly.

Figure 4.—Thrust plate and thrust coils.

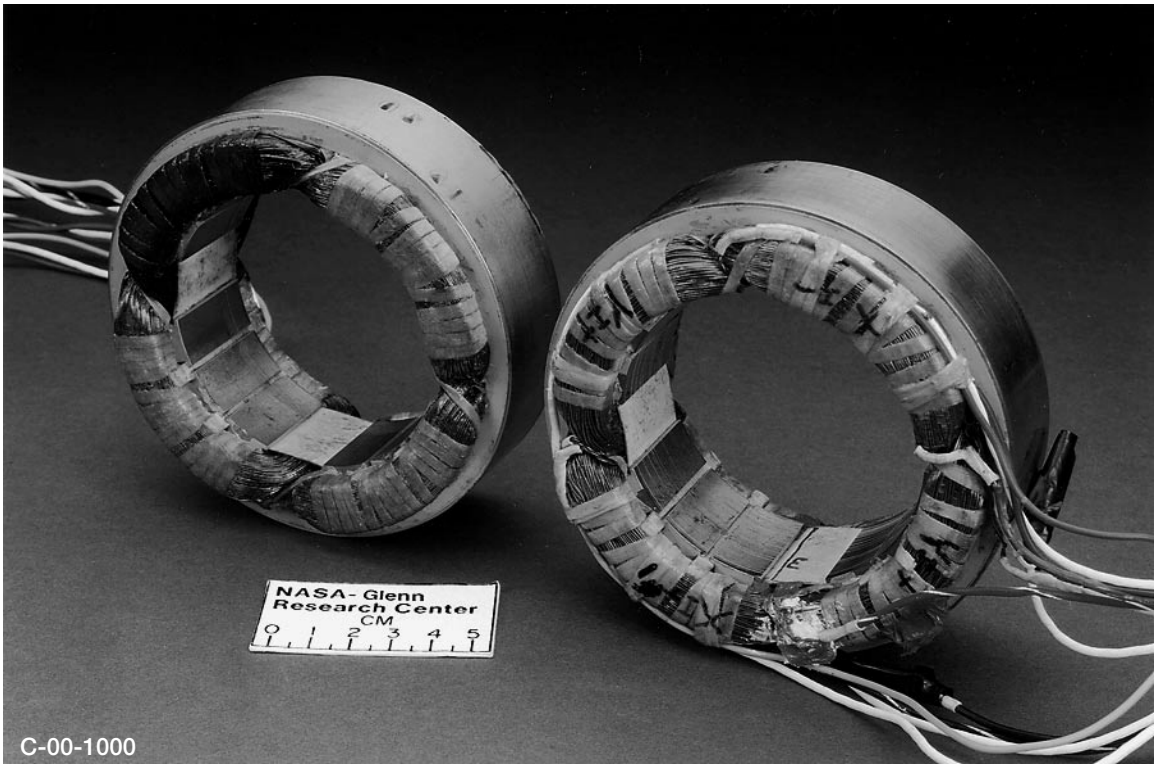


Figure 5.—Upper and lower radial stators.

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

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

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 $-(K_p x + K_d x) + i_{ex}$ 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

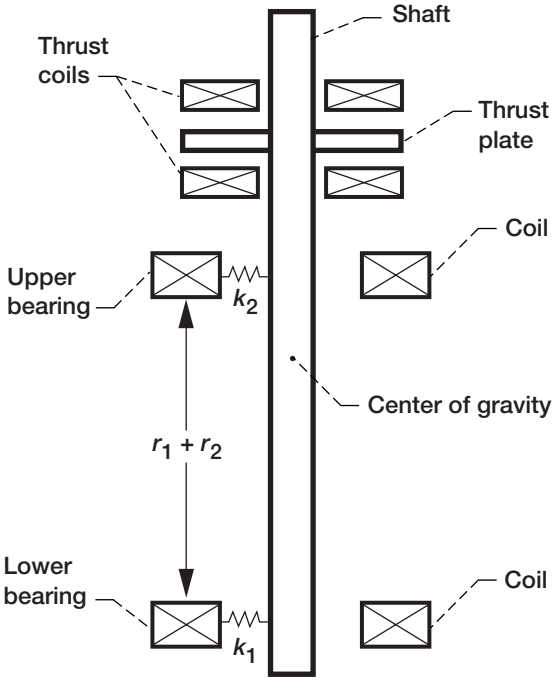


Figure 6.—Five-axis ensemble (not to scale).

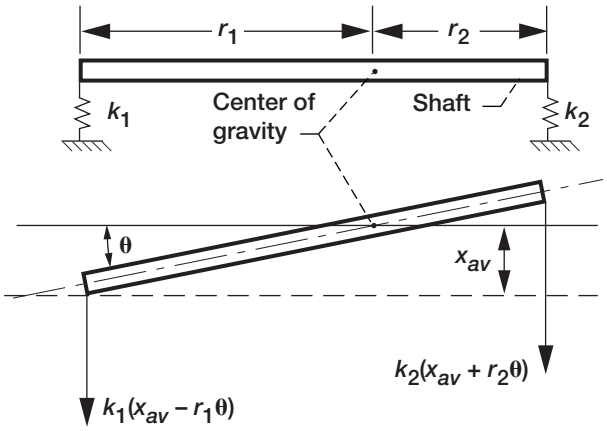


Figure 7.—Free-body diagram for modal control mathematical derivation.

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

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

$$m\ddot{x} = -\left(k_1 + k_2\right)x_{av} - \left(k_2 r_2 - k_1 r_1\right)\theta - \left(c_1 + c_2\right)\dot{x}_{av} - \left(c_2 r_2 - c_1 r_1\right)\theta \tag{8}$$

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 $c_1 = c_{\text{eq1}}$ and $c_2 = c_{\text{eq2}}$.

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

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

$$I_G\ddot{\theta} = \left(k_1 r_1 - k_2 r_2\right)x_{av} - \left(k_2 r_2^2 + k_1 r_1^2\right)\theta \tag{10}$$

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

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

$$\text{Force (rotation)} = -\left(k_2 r_2^2 + k_1 r_1^2\right)\theta - \left(c_2 r_2^2 + c_1 r_1^2\right)\dot{\theta} \tag{12}$$

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.

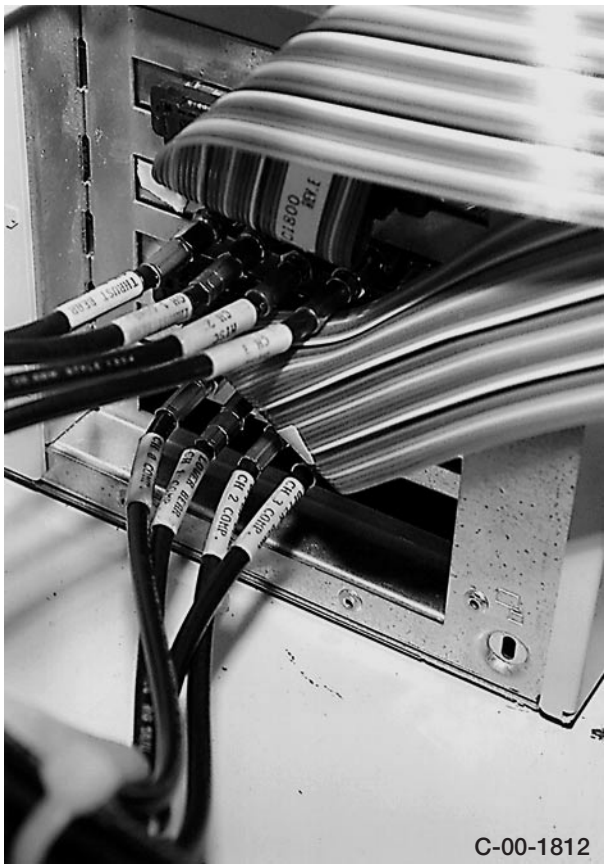


Figure 8.—Input and output board configuration in central processing unit.

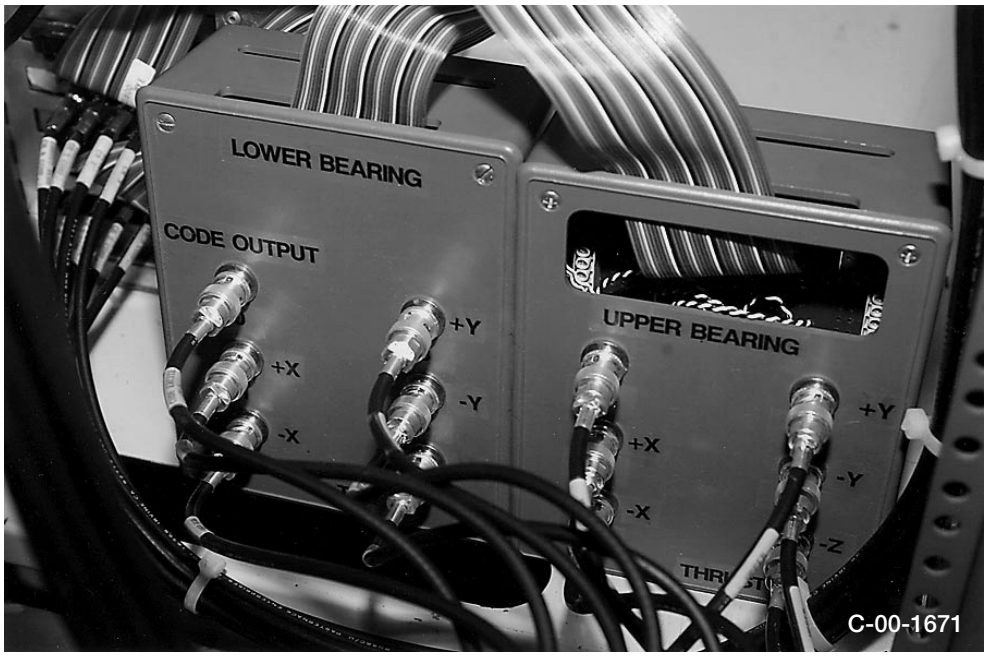


Figure 9.—Twelve-channel output box from central processing unit.

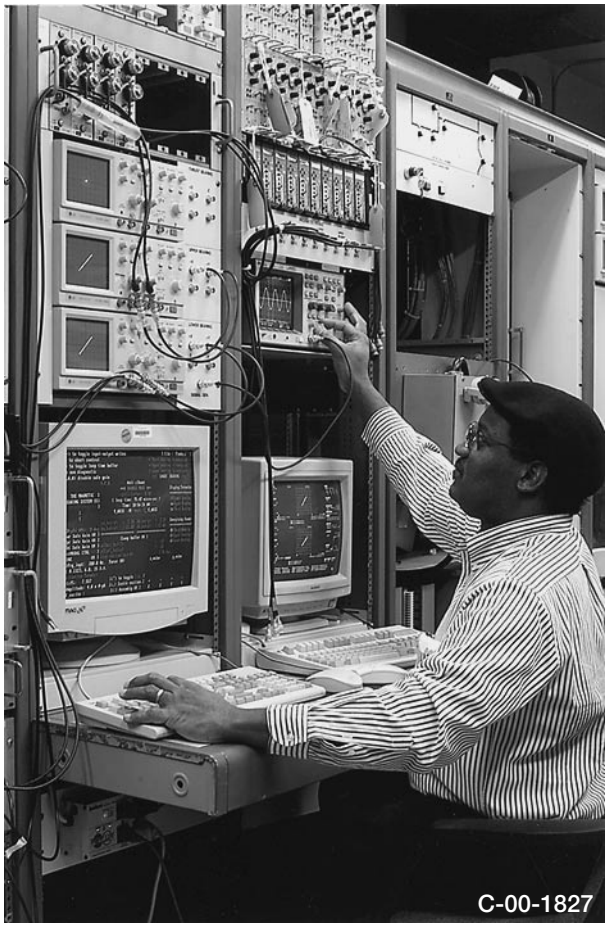


Figure 10.—Operations center for Five-Axis, Three-Magnetic-Bearing Dynamic Spin Rig.

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

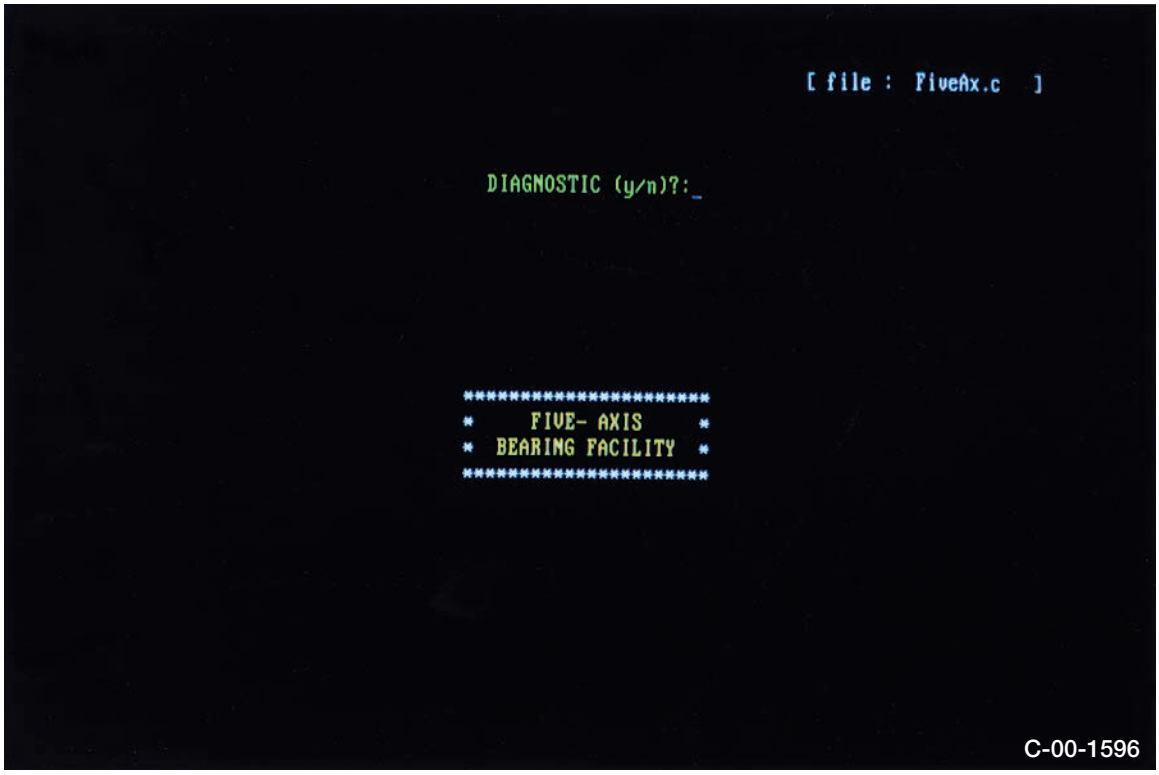


Figure 11.—Initial screen display.

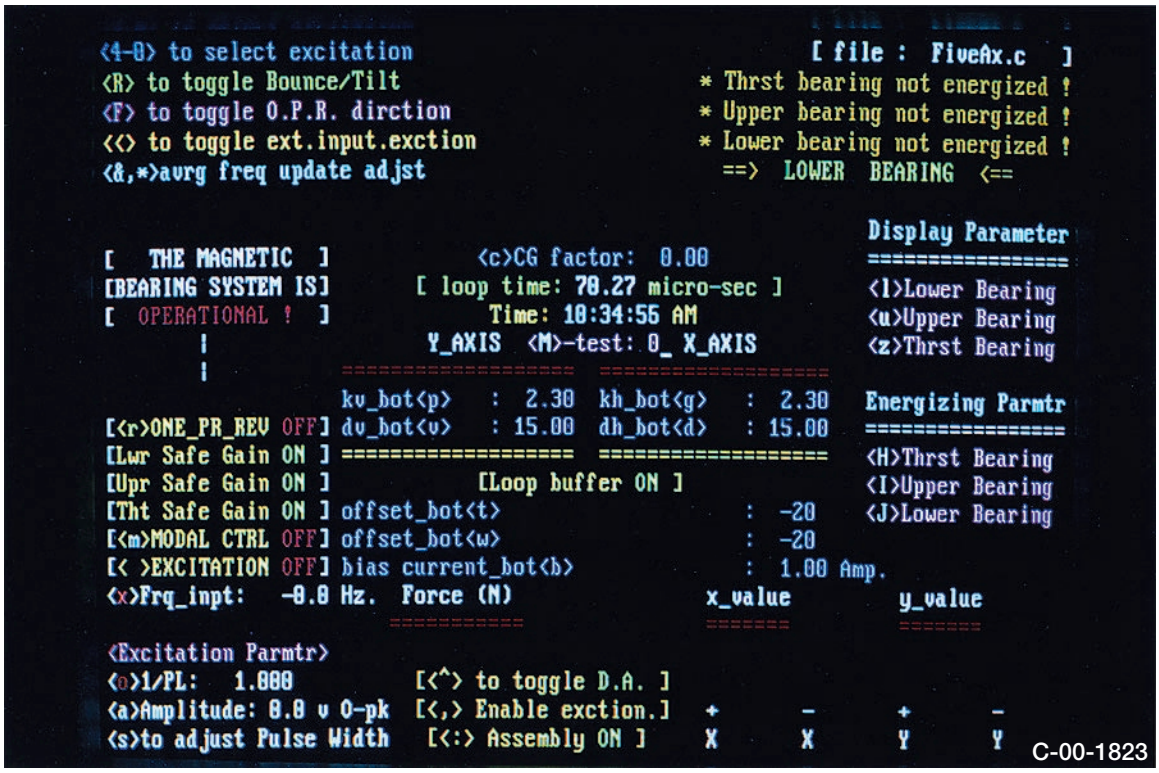


Figure 12.—Initial diagnostic mode screen display.

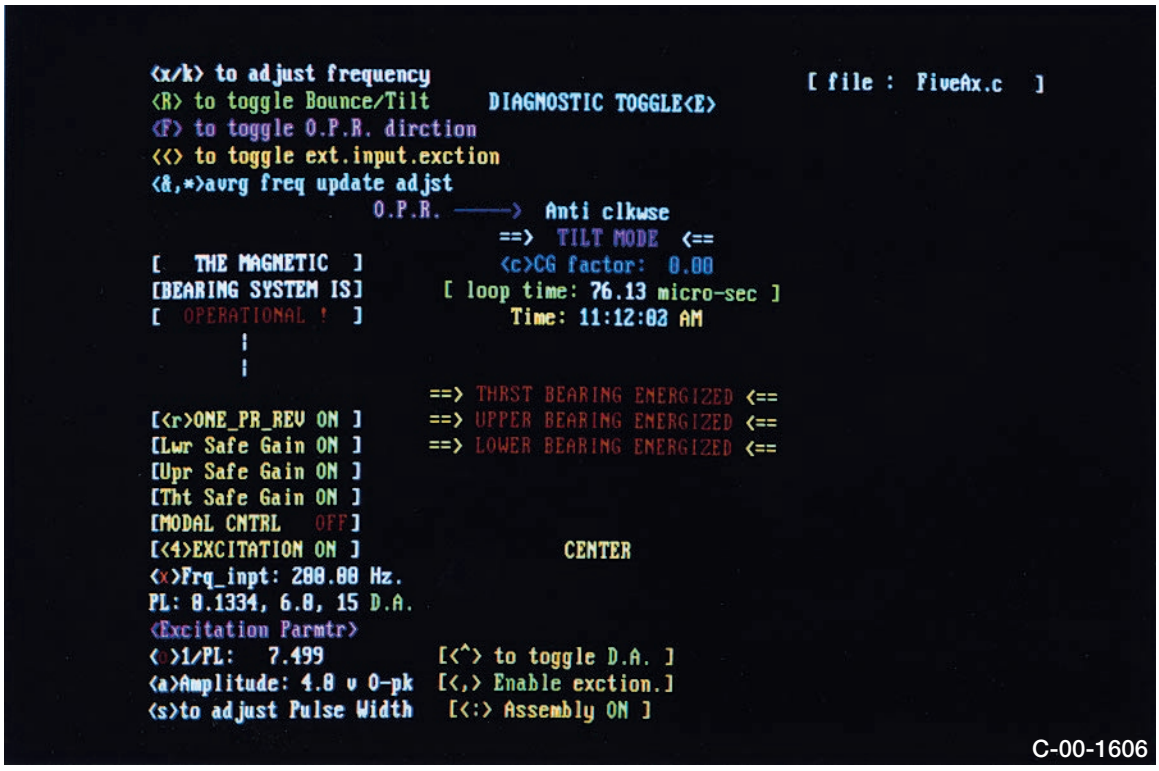
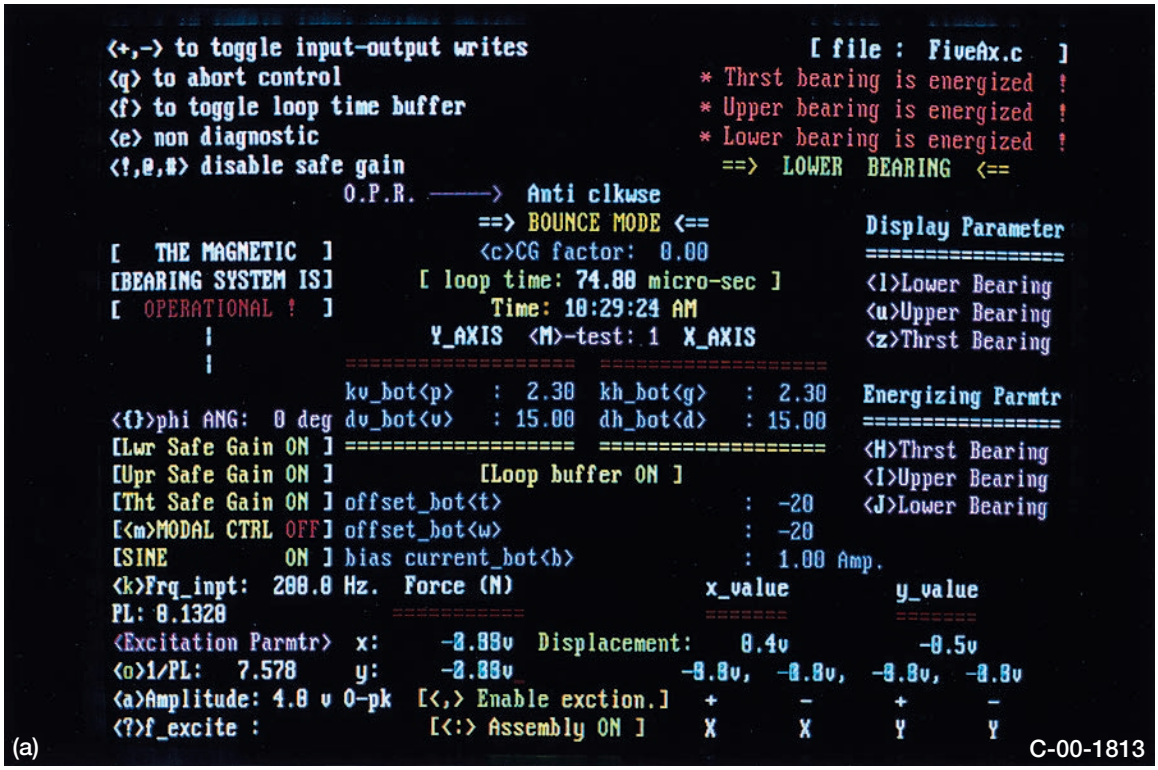


Figure 13.—Nondiagnostic screen display.



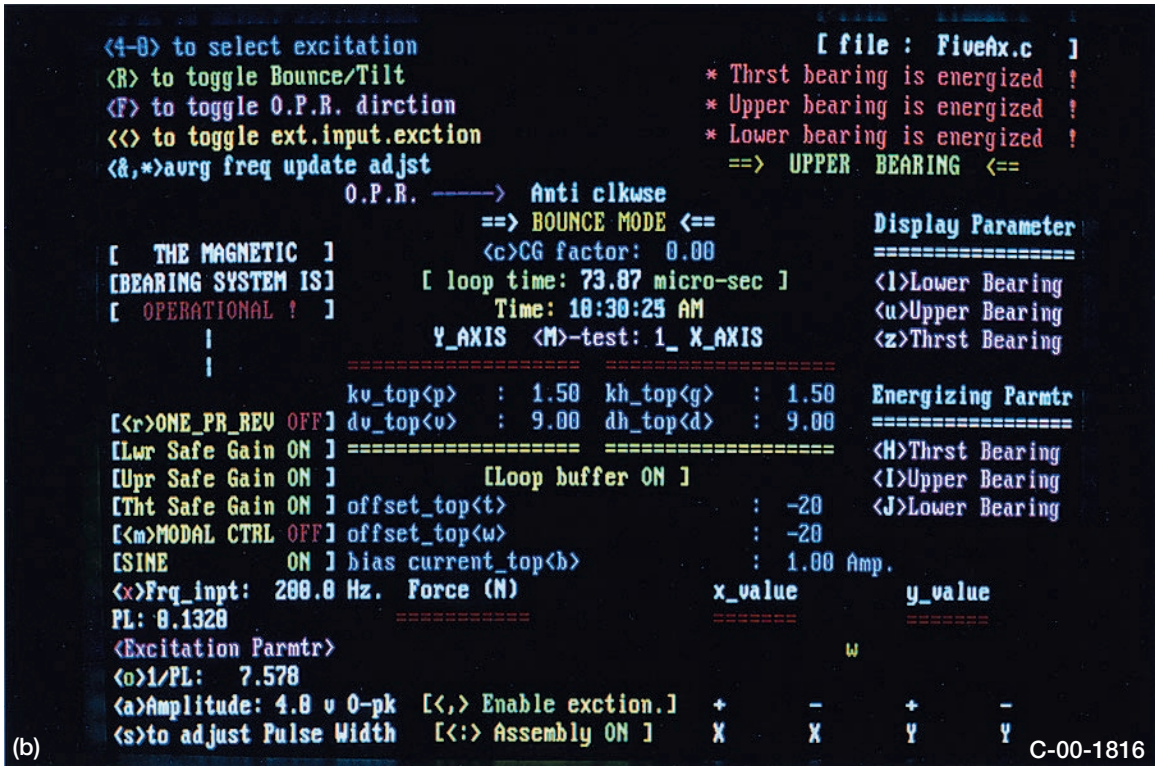


Figure 14.—Diagnostic mode screen displays for upper and lower bearings. (a) Lower bearing. (b) Upper bearing.

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 $\pm z$, 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).

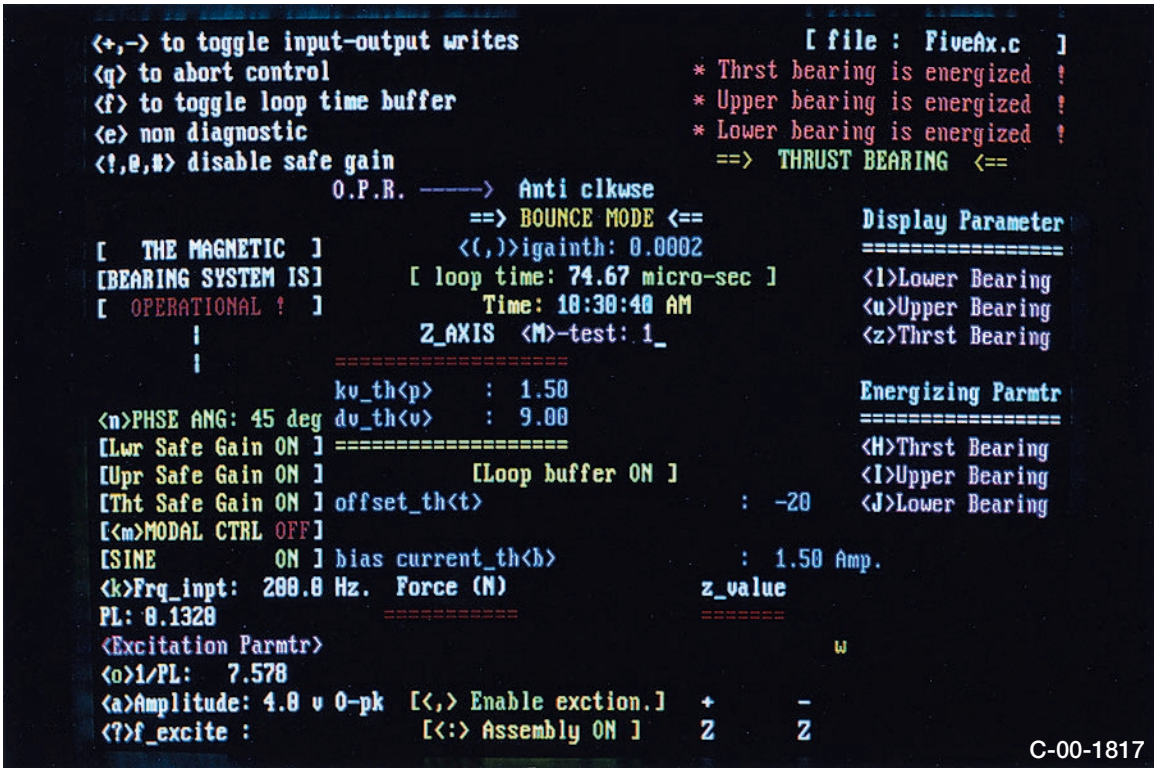


Figure 15.—Diagnostic mode screen display for thrust bearing.

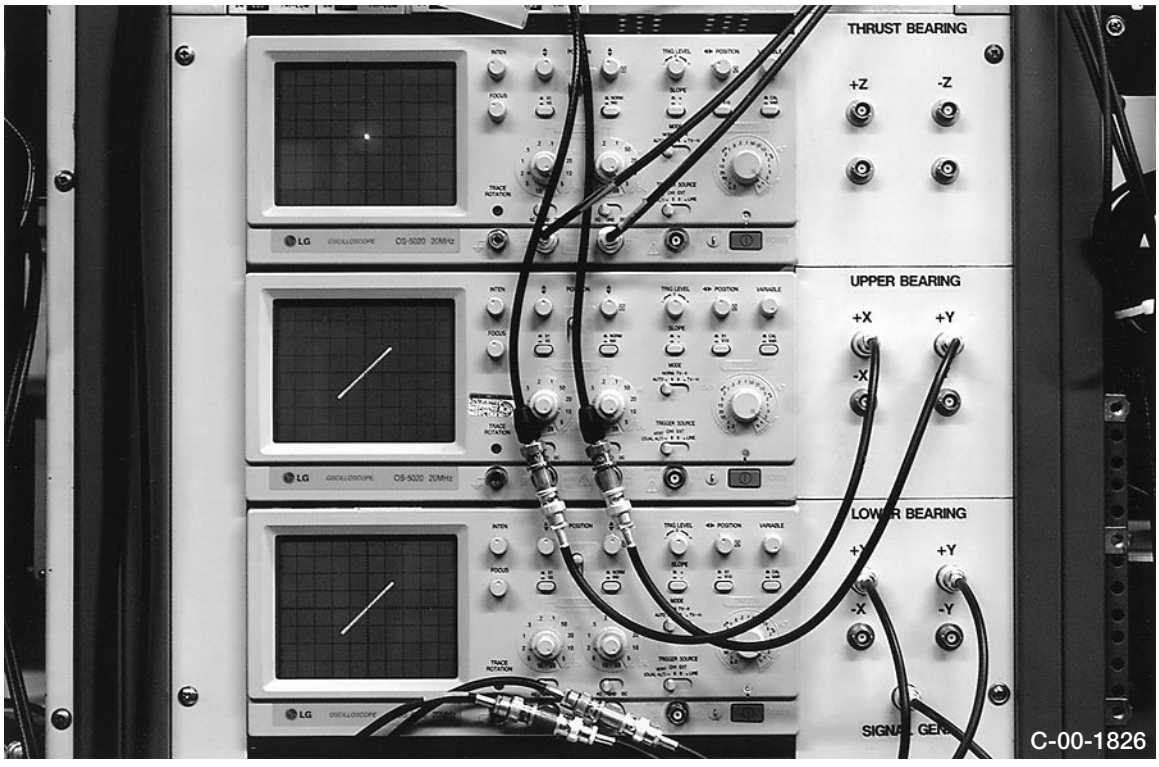


Figure 16.—Position display screen for thrust, upper, and lower bearings.

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., $\pm z$ 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

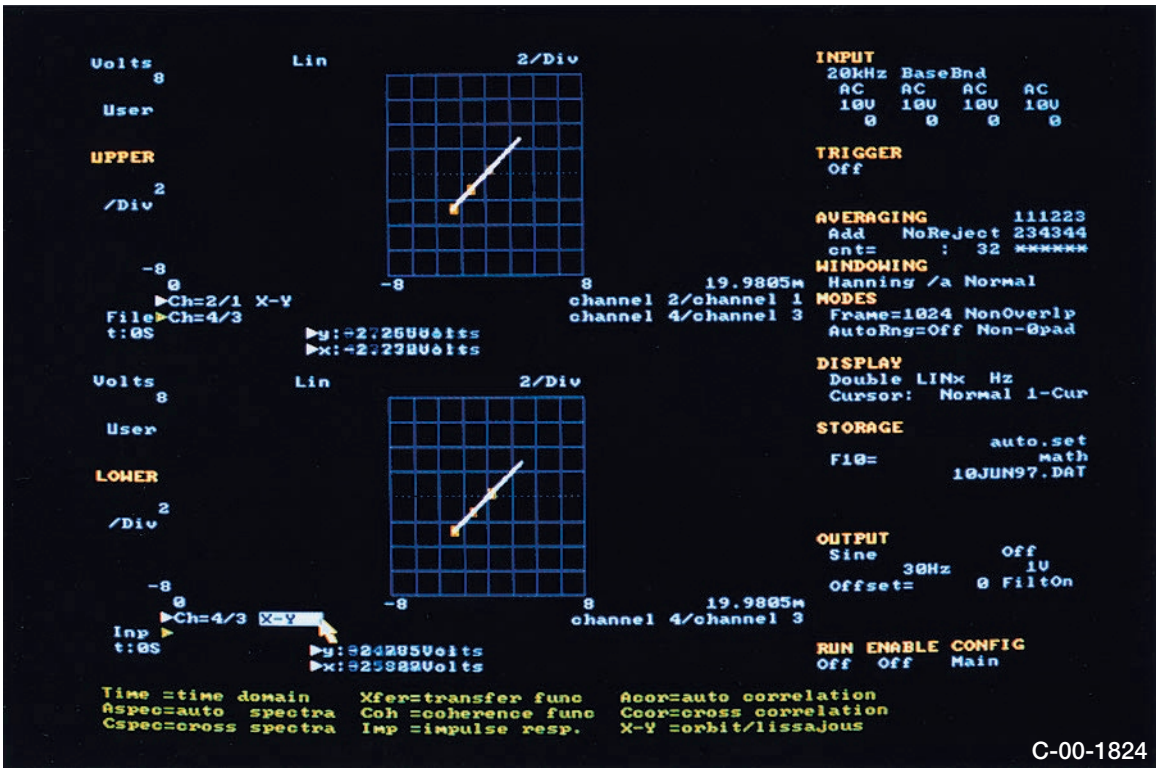


Figure 17.—Tektronix *x*,*y*-screen display of upper and lower bearing rotor displacement.

("<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.

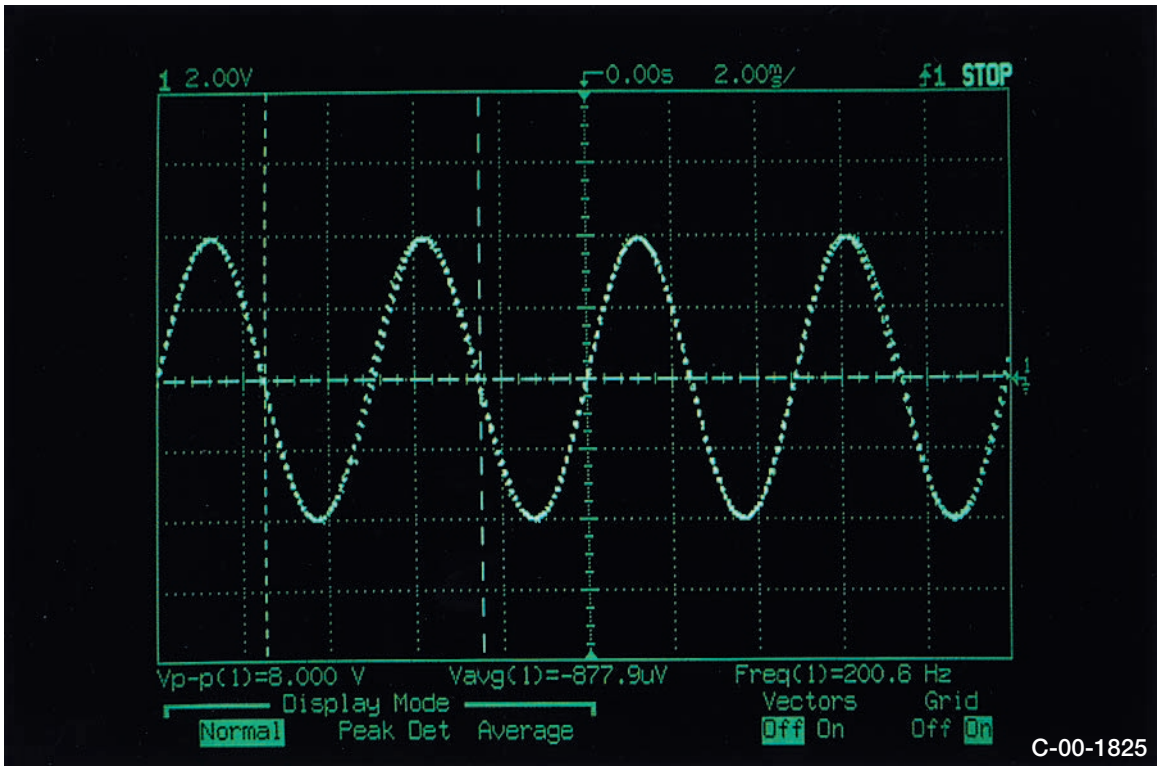


Figure 18.—Hewlett Packard digital scope display of sine curve excitation signal.

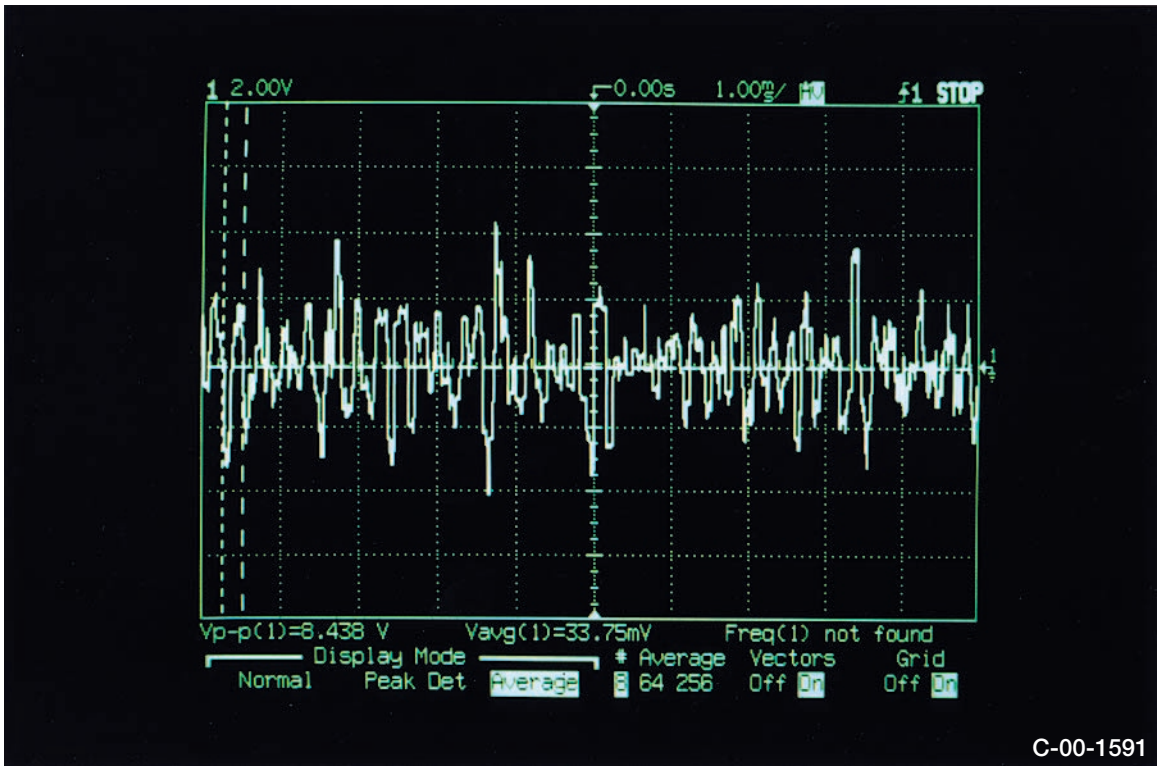


Figure 19.—Hewlett Packard digital scope display of random curve excitation signal.

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:."

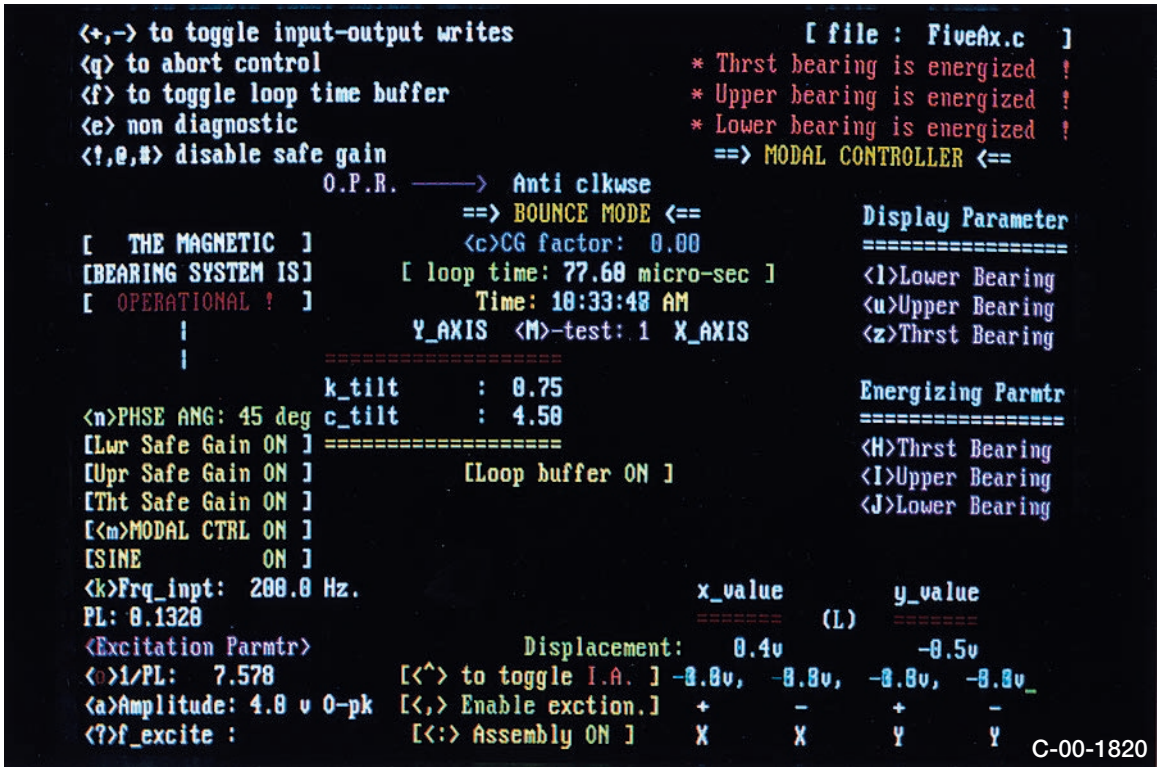


Figure 20.—Modal control display screen.

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.

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

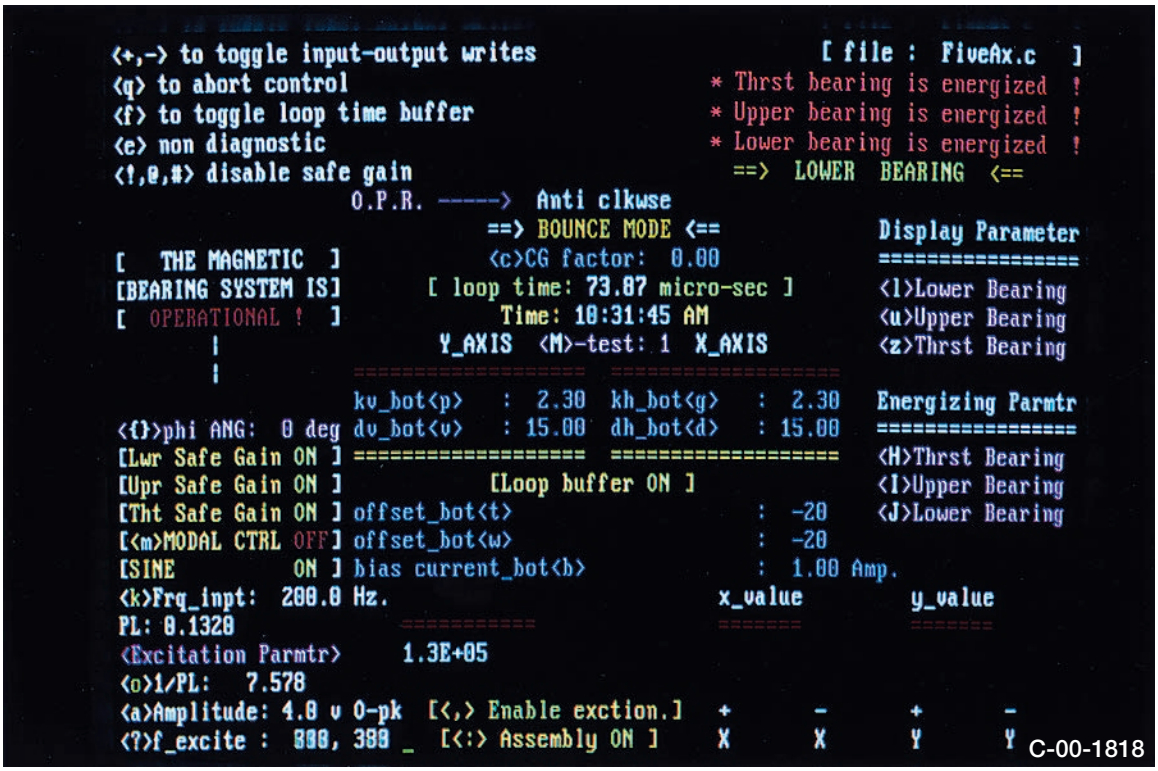


Figure 22.—Lower bearing display screen of a selected internal signal function (note outputs 200.0 Hz, 1.3E+05, 7.578).

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 “Force (N)”) 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:

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

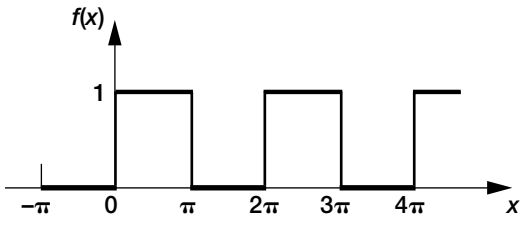
Random:

$$f(x) = A \times \sin(2.0 \times \pi \times \text{f\_excite}\,3) \times \left[\sin(2.0 \times \pi \times O \times x) + \sin(2.0 \times \pi \times \text{f\_excite}\,4 \times O)\right] \tag{18}$$

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.

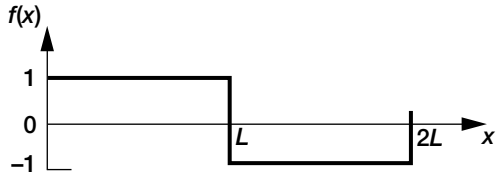
Squared pulse train:

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



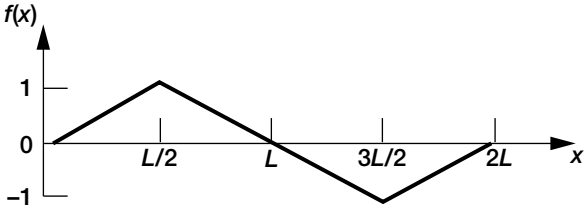
Square wave:

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



Triangular wave:

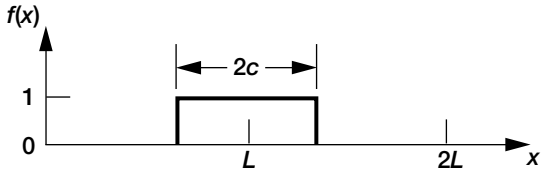
$$f(x) = A \times \left(\frac{8}{\pi^2} \left\{ \sum_{k=0}^{40} \left[\frac{(-1)^k}{(2k+1)^2} \right] \times \sin\left[2.0 \times (2k+1) \times \pi \times O \times x \right] \right\} \right) \tag{21}$$



Single square pulse:

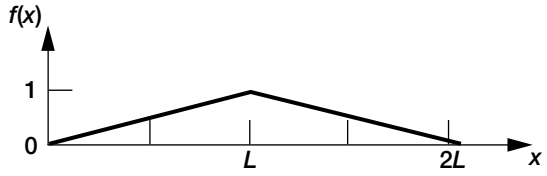
$$f(x) = A \times \left[O \times C + \frac{2}{\pi} \left(\sum_{k1=1}^{40} \left\{ \left[\frac{(-1)^{k1}}{k1} \right] \times \sin(k1 \times \pi \times O \times C) \times \cos(2.0 \times k1 \times \pi \times O \times x) \right\} \right) \right] \tag{22}$$

where C is the pulse width PW.



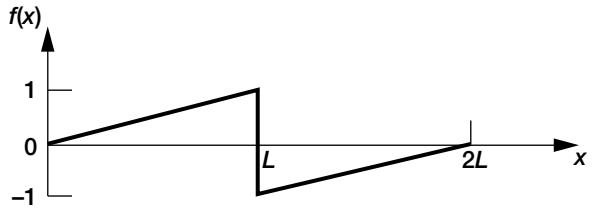
Single triangular pulse:

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



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.

```
1
2
3
4
5
6
7
8
9
10
11  #include<stdio.h>
12  #include<dos.h>
13  #include<conio.h>
14  #include<math.h>
15  #include<time.h>
16  #include<stdlib.h>
17
18  /*------------------------VARIABLE DECLARATION-----------------------------*/
19
20  int board,lchan1,lchan2,lchan3,pchan1,pchan2,pchan3,erstat,xbot,ybot,xtop,
21      ytop,zth,zth1,zth2,x_bot_old1,x_bot_old2,x_bot_old3,x_bot_old4,
22      x_bot_old5,y_old_bot,y_old_top,y_old_th,y_bot_old1,y_bot_old2,
23      y_bot_old3,y_bot_old4,y_bot_old5,x_top_old1,x_top_old2,x_top_old3,
24      x_top_old4,x_top_old5,y_top_old1,y_top_old2,y_top_old3,y_top_old4,
25      y_top_old5,z_th_old1,z_th_old2,z_th_old3,z_th_old4,z_th_old5,Base1,
26      Base2,out_chan1_0,out_chan1_1,out_chan1_2,out_chan1_3,out_chan1_4,
27      out_chan1_5,out_chan2_0,out_chan2_1,out_chan2_2,out_chan2_3,out_chan2_4,
28      out_chan2_5,i_bot,i_top,i_th,j,tBias_bot,tBias_top,tBias_th,wBias_bot,
29      wBias_top,wBias_th,nw_bot,nw_top,nw_th,fig,out_min,out_max,n,jjj,
30      bias_current_bot,bias_current_top,bias_current_th,nmax,lmax,l,
31      PD_tBias_bot,PD_tBias_top,PD_tBias_th,PD_wbias_bot,PD_wbias_top,
32      PD_wbias_th,valuenoise,FIFO1,FIFO2,zero,one,two,hh,g,vv=15,k,k1,m,m1,
33      m2,m3,m4,p,x0,d_max_th,d,v,ROUND,flag1,flag2,flag3,flag4,flag5,
34      flag6,flag7,flag8,flag9,flag10,flag12,flag13,flag15,flag11,flag22,
35      flag33,flag44,flag16,flag18,flag19,flag20,flag21,flag23,flag24,flag25,
36      flag4a,flag4b,flag4c,flag4d,flag_A,flag_B,flag_C,flag_D,flag_E,flag_F,
37      flag_G,flag_H,flag_I,flag_J,flag_K,flag_L,flag_M,flag_N,flag_AA,flag_BB,
38      flag_CC,flag_DD,flag_EE,flag_FF,flag_GG,flag_HH,flag_II,flag_JJ,flagJJ,
39      thp,flag_jj,flagKK,flagLL,flagMM,flagNN,out_bot,out_top,out_th,diag,t48,
40      round2,cir,cir2,sg1,sg2,sg3,excite,f_excite,f_excite2,num,n_x,SSS,th,
41      i_rev,one_per_rev,trigger=21,rise,N_ticks,j_rev,X_P_O_B,X_N_O_B,Y_P_O_B,
42      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,
43      excite_sin,maxv,set=1,rr=0,qq=0,i;
44
45  double I_lim,loop_time,last_time,micro,junk,ibias_bot,ibias_top,ibias_th,
46         dh_bot,kh_top,dh_top,dh_th,dv_bot,kh_bot,kh_th,kv_bot,kv_top,kv_th,
47         dv_top,dv_th,x_force_bot,y_force_bot,x_force_top,y_force_top,
48         z_force_th,xbotderiv,ybotderiv,xtopderiv,ytopderiv,zthderiv,
49         x_pos_output_bot,x_neg_output_bot,x_pos_output_top,x_neg_output_top,
50         up_output_th,down_output_th,y_pos_output_bot,y_pos_output_top,
51         y_neg_output_bot,y_neg_output_top,z,xbotsum,ybotsum,xtopsum,zthsum,
52         ytopsum,igainbot,igaintop,igainth,igainmod,safe,zsafe,x,O,frequency,
53         period,PL,ex,f_ex,volt,C,PW,PWW,freq,t04,THETA,f_excite_cos,
54         f_excite_sin,PI2_o_Nticks,PI2,phi,i_rev1,pp=0.0,Yav,Xav,xbot_force_tr,
55         xtop_force_tr,ybot_force_tr,ytop_force_tr,dotXav,dotYav,oldoldXav,
56         oldXav,oldoldYav,oldYav,ThetaX,ThetaY,L,xbot_force_rot,k_tilt,c_tilt,
57         dotThetaX,xtop_force_rot,ybot_force_rot,dotThetaY,ytop_force_rot,
58         oldoldThetaX,oldoldThetaY,oldThetaY,oldThetaX,xbot_force_modal_pos,
```

```
59          xbot_force_modal_neg,xtop_force_modal_pos,xtop_force_modal_neg,
60          ybot_force_modal_pos,ybot_force_modal_neg,ytop_force_modal_pos,II,JJ,
61          ytop_force_modal_neg,F_XB_tr,F_XT_tr,F_YB_tr,F_YT_tr,excitef,
62          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,
63          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,
64          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,
65          B10=0.0,B11=0.0,B12=0.0,B13=0.0,B14=0.0,B15=0.0,f_excite3,f_excite4,
66          xy=0.0,COUNTMAX=15.0,MCG,PCG,cos(double x),sin(double x),ns;
67
68    struct time now,tt;
69
70    unsigned int ti_min,ti_second,ti_hund;
71
72    float round1(float u),randvalue,time1;
73
74    char resp,lu,respp,ig;
75
76    const int NUMBERS = 1;
77
78    int main(void)
79
80    /*------------------------------INITIALIZE --------------------------------*/
81
82    {
83       clrscr();
84
85    // ******************* Datel Input Board (1) setup *******************
86              // Board address: 0x300
87       outportb(0x30e, 0x3a);          j = 1; while ( j<5000 ) j++;
88       outportb(0x308,    2);          j = 1; while ( j<5000 ) j++;
89       outportb(0x308,    0);          j = 1; while ( j<5000 ) j++;
90
91       outportb(0x30e, 0x7a);          j = 1; while ( j<5000 ) j++;
92       outportb(0x30a,    1);          j = 1; while ( j<5000 ) j++;
93       outportb(0x30a,    0);          j = 1; while ( j<5000 ) j++;
94
95       outportb(0x30e, 0xba);          j = 1; while ( j<5000 ) j++;
96       outportb(0x30c,    1);          j = 1; while ( j<5000 ) j++;
97       outportb(0x30c,    0);          j = 1; while ( j<5000 ) j++;
98
99       outport (0x302, 0x40);          j = 1; while ( j<5000 ) j++;
100      outport (0x306,    1);          j = 1; while ( j<5000 ) j++;
101      outport (0x300,  0xe);          j = 1; while ( j<5000 ) j++;
102
103   // ******************* Datel Input Board (2) setup *******************
104             // Board address: 0x360
105      outportb(0x36e, 0x3a);          j = 1; while ( j<5000 ) j++;
106      outportb(0x368,    2);          j = 1; while ( j<5000 ) j++;
107      outportb(0x368,    0);          j = 1; while ( j<5000 ) j++;
108
109      outportb(0x36e, 0x7a);          j = 1; while ( j<5000 ) j++;
110      outportb(0x36a,    1);          j = 1; while ( j<5000 ) j++;
111      outportb(0x36a,    0);          j = 1; while ( j<5000 ) j++;
112
113      outportb(0x36e, 0xba);          j = 1; while ( j<5000 ) j++;
114      outportb(0x36c,    1);          j = 1; while ( j<5000 ) j++;
115      outportb(0x36c,    0);          j = 1; while ( j<5000 ) j++;
116
```

```
117     outport (0x362, 0x40);          j = 1; while ( j<5000 ) j++;
118     outport (0x366,    1);          j = 1; while ( j<5000 ) j++;
119     outport (0x360,  0xe);          j = 1; while ( j<5000 ) j++;
120
121     FIFO1 = 0x306;// Base = 300, FIFO1 = base + 6;
122     FIFO2 = 0x366;// Base = 360, FIFO2 = base + 6;
123
124  // ****************  Metrabyte Output Board (1) setup ****************
125
126     Base1 = 0x330;// Board address: 0x330 Lower Bearing + Thrust up (Z+)
127     out_chan1_0 = Base1 +  0;
128     out_chan1_1 = Base1 +  2;
129     out_chan1_2 = Base1 +  4;
130     out_chan1_3 = Base1 +  6;
131     out_chan1_4 = Base1 +  8;
132     out_chan1_5 = Base1 + 10;
133
134     t48 = 2048;// 2048 => Ten volts
135
136     outport(out_chan1_0, t48);// Code's signal output
137     outport(out_chan1_1, t48);// +X_L
138     outport(out_chan1_2, t48);// -X_L
139     outport(out_chan1_3, t48);// +Y_L
140     outport(out_chan1_4, t48);// -Y_L
141     outport(out_chan1_5, t48);// +Z_TH
142
143  // ****************  Metrabyte Output Board (2) setup ****************
144
145     Base2 = 0x390;// Board address: 0x390 Upper Bearing + Thrust down (Z-)
146  // out_chan2_0 = Base2 +  0;
147     out_chan2_1 = Base2 +  2;
148     out_chan2_2 = Base2 +  4;
149     out_chan2_3 = Base2 +  6;
150     out_chan2_4 = Base2 +  8;
151     out_chan2_5 = Base2 + 10;
152
153  // outport(out_chan2_0, t48);
154     outport(out_chan2_1, t48);// +X_U
155     outport(out_chan2_2, t48);// -X_U
156     outport(out_chan2_3, t48);// +Y_U
157     outport(out_chan2_4, t48);// -Y_U
158     outport(out_chan2_5, t48);// -Z_TH
159
160
161  // ********************** GENERAL VARIABLE INITIALIZATION ******************
162
163     safe  = 32600;
164     zsafe = 16300;
165     nmax = 500; lmax = 150; l = 0;
166     micro = (1000000.0 / nmax / lmax);
167     I_lim = 4.0;
168     out_min = -round1(2.0 * I_lim * 204.8) + t48;
169     out_max =  round1(2.0 * I_lim * 204.8) + t48;
170     loop_time = 0.78; hh = 0;
171     zero = 0; one = 1; two = 2;
172     LIM = 1.75 * pow(10,308);// max # of period lengths (upper limit)
173     x0 = 21;/*(0.1)*//*103(.5v)*//*205(1v)*//*1435(7v)*/
174     k = 0;
```

```
175     k1 = 1;
176     x = 0.0;
177     f_excite = 0.0;
178     excite = 0.0;
179     f_excite_sin = 0.0;
180     f_excite_cos = 0.0;
181     JJ = 1.0;
182     II = 1.0;
183     ex = 0.0;
184     O = 1.0;
185     frequency = 0.0;
186     PWW = 0.0;
187     PW = 0.0;
188     i_rev = 0;
189     j_rev = 0;
190     THETA = 0.0;
191     th = 0;
192     PI2 = 2 * M_PI;
193     phi = 0.0;
194     L = 1.0;
195     TC = 9;
196     test_signal = 0;
197     t04 = 0.0;
198     freq = 0.0;
199     PL = 1.0;
200     CG = 0.0;
201     MCG = 0.5 - CG;
202     PCG = 0.5 + CG;
203     cir = 23;
204     cir2 = 55;
205     flag5 = 0;
206     flag6 = 0;
207     flag7 = 0;
208     flag8 = 0;
209     flag9 = 0;
210     flag12 = 0;
211     flag13 = 0;
212     flag10 = 0;// Disable modal block
213     flag15 = 1;
214     flag16 = 1;// Assembly condition (on)
215     flag18 = 0;
216     flag19 = 0;
217     flag20 = 1;
218     flag21 = 0;// Disable excitation block
219     flag23 = 1;
220     flag24 = 1;// Enable loop_time averaging
221     flag25 = 1;// Toggle loop_time averaging
222     flag_A = 1;// Assembly toggle set to off
223     flag_B = 1;
224     flag_C = 1;
225     flag_D = 1;
226     flag_E = 1;
227     flag_F = 1;
228     flag_G = 1;
229     flag_H = 1;
230     flag_I = 1;
231     flag_J = 1;
232     flag_K = 0;
```

```
233     flag_L = 0;
234     flag_M = 1;
235     flag_N = 1;
236     flag_AA = 0;
237     flag_BB = 1;
238     flag_GG = 1;
239     flag_HH = 0;
240     flag_II = 0;
241     flagJJ = 1;
242     flag_JJ = 1;
243     flag_jj = 1;
244     flagKK = 1;
245     flagLL = 1;
246     flagMM = 0;
247     flagNN = 1;
248     switch1 = 0;
249
250  // ******************* BOTTOM BEARING VARIABLE INITIALIZATION **************
251
252     kv_bot = 2.3;
253     kh_bot = kv_bot;
254     dh_bot = 15.0;
255     dv_bot = dh_bot;
256     ibias_bot = 1.0;// Amperes
257     bias_current_bot = round1(ibias_bot * 2.0 * 204.8);// two Volts => one Amp.
258
259     // Remember amplifier gain is 0.5A/V
260
261     PD_tBias_bot = -20; PD_wbias_bot = -20;// Initial differential biases
262     tBias_bot = PD_tBias_bot; wBias_bot = PD_wbias_bot;
263     nw_bot = 0;// For writeout, set nw_bot = 1
264     sg1 = 1;// Lower Bearing safe gain set
265
266  // ****************** TOP BEARING VARIABLE INITIALIZATION *******************
267
268     kv_top = 2.3;
269     kh_top = kv_top;
270     dh_top = 15.0;
271     dv_top = dh_top;
272     ibias_top = 1.0;// Amperes
273     bias_current_top = round1(ibias_top * 2.0 * 204.8);// Two Volts => one Amp
274
275     // Remember amplifier gain is 0.5A/V
276
277     PD_tBias_top = -20; PD_wbias_top = -20;// Initial differential biases
278     tBias_top = PD_tBias_top; wBias_top = PD_wbias_top;
279     nw_top = 0;
280     sg2 = 1;// Upper Bearing safe gain set
281
282  // ***************** THRUST BEARING VARIABLE INITIALIZATION *****************
283
284     kv_th = 2.3;
285     dv_th = 15.0;
286     ibias_th = 1.5;// Amperes multiplication factor
287     igainth   = 0.0002;
288     bias_current_th = round1(ibias_th * 2.0 * 204.8);// Two Volts => one Amp
289
290     // Remember amplifier gain is 0.5A/V
```

```
291
292     PD_tBias_th = -20;                      // Initial differential biases
293     tBias_th = PD_tBias_th;
294     nw_th = 0;
295     sg3 = 1;// Thrust Bearing safe gain set
296
297  // ***************************************************************************
298
299      flag1  = 0;
300      flag2  = 0;
301      flag3  = 0;
302      flag4  = 1;
303
304      flag4a = 0;
305      flag4b = 0;
306      flag4c = 0;
307      flag4d = 1;
308
309      flag11 = 1;// Enable lower bearing write out block
310      flag22 = 0;// Disable upper bearing write out block
311      flag33 = 0;// Disable thrust bearing write out block
312      flag44 = 0;// Enable D.A/I.A. display
313
314  // -------------------------------SHOW MENU---------------------------------
315
316      clrscr();
317
318      gotoxy(45,6);textcolor(4);
319      gotoxy(59,1);textcolor(15);
320      cprintf("[ file :  FiveAx.c   ]");
321      gotoxy(29,13);textcolor(15);
322      cprintf("**********************");
323      gotoxy(29,14);textcolor(15);
324      cprintf("*                    *");
325      gotoxy(29,15);textcolor(15);
326      cprintf("*                    *");
327      gotoxy(29,16);textcolor(15);
328      cprintf("**********************");
329      gotoxy(35,14);textcolor(14);
330      cprintf("FIVE- AXIS");
331      gotoxy(32,15);textcolor(14);
332      cprintf("BEARING FACILITY");
333
334  G:  gotoxy(31,5);textcolor(10);
335      cprintf("DIAGNOSTIC (y/n)?:");
336      respp = getch();
337      gotoxy(31,5);
338      printf("                  ");// Erase "DIAGNOSTIC (y/n)?:"
339      if (respp == 'y' || resp == 'Y')
340      {
341        SSS = 1;
342        diag  = 1;
343
344        clrscr();
345
346        goto H;
347      }
348
```

```
349       else
350
351       if (respp == 'n' || resp == 'N')
352       {
353         clrscr();
354
355         SSS = 0;
356         gotoxy(1,1);textcolor(15);
357         cprintf("<x/k> to adjust frequency");
358         gotoxy(1,2);textcolor(15);
359         cprintf("<q> to abort control");
360         gotoxy(1,3);textcolor(15);
361         cprintf("<m> to toggle modal cntrl");
362         gotoxy(1,4);textcolor(15);
363         cprintf("<?> to toggle f_excite");
364         gotoxy(1,5);textcolor(15);
365         cprintf("<4-0> to select excitation");
366         gotoxy(59,1);textcolor(15);
367         cprintf("[ file :  FiveAx.c   ]");
368         gotoxy(31,2);textcolor(11);
369         cprintf("DIAGNOSTIC TOGGLE<E>");
370         gotoxy(1,22);textcolor(13);
371         cprintf("<Excitation Parmtr>");
372         gotoxy(1,14);textcolor(10);
373         cprintf("< >PHSE ANG:%3u deg",th);
374         gotoxy(2,14);textcolor(15);
375         cprintf("n");
376         gotoxy(1,23);textcolor(15);
377         cprintf("<o>freq:%8.2f  Hz",frequency);
378         gotoxy(1,20);textcolor(15);
379         cprintf("<x>Frq_inpt:%7.1f Hz.",freq);
380         gotoxy(1,25);textcolor(15);
381         cprintf("<s>to adjust Pulse Width");
382         gotoxy(1,24);textcolor(15);
383         cprintf("<a>Amplitude:%4.1f v O-pk",volt);
384         gotoxy(27,23);textcolor(14);
385         cprintf("[<^> to toggle D.A. ]");
386         gotoxy(27,24);textcolor(14);
387         cprintf("[<,> Enable exction.]");
388         gotoxy(28,25);textcolor(14);
389         cprintf("[<:> Assembly    ]");
390         gotoxy(42,25);textcolor(10);
391         cprintf("ON");
392
393         nw_bot = 0;
394         nw_top = 0;
395         nw_th  = 0;
396
397         diag  = 0;
398         flag1 = 1;// Lower bearing block activated
399         flag2 = 1;// Upper bearing block activated
400         flag3 = 1;// Thrust bearing block activated
401
402         goto H;
403       }
404         goto G;
405
406  H:   if (diag == 1)
```

```
407     {
408       gotoxy(59,1);textcolor(15);
409       cprintf("[ file :  FiveAx.c   ]");
410       gotoxy(1,1);textcolor(15);
411       cprintf("<+,-> to toggle input-output writes");
412       gotoxy(1,2);textcolor(15);
413       cprintf("<q> to abort control");
414       gotoxy(1,3);textcolor(15);
415       cprintf("<f> to toggle loop time buffer");
416       gotoxy(1,4);textcolor(15);
417       cprintf("<e> non diagnostic");
418       gotoxy(1,5);textcolor(15);
419       cprintf("<!,@,#> disable safe gain");
420       gotoxy(19,11);textcolor(15);
421       cprintf("         Y_AXIS               X_AXIS");
422       gotoxy(36,11);textcolor(13);
423       cprintf("< >-test: %1u",test_signal);
424       gotoxy(37,11);textcolor(15);
425       cprintf("M");
426       gotoxy(21,12);textcolor(4);
427       cprintf("==================  ==================");
428       gotoxy(21,15);textcolor(14);
429       cprintf("==================  ==================");
430       gotoxy(52,5);textcolor(14+128);
431       cprintf("==>                 <==");
432       gotoxy(57,5);textcolor(10);
433       cprintf("LOWER  BEARING");
434       gotoxy(31,8);textcolor(9);
435       cprintf(" <c>CG factor: %5.2f",CG);
436       gotoxy(32,16);textcolor(14);
437       cprintf("[loop buffer    ]");
438       gotoxy(45,16);textcolor(10);
439       cprintf("ON ");
440       gotoxy(21,13);textcolor(9);
441       cprintf("kv_bot<p>   :%6.2f", kv_bot);
442       gotoxy(42,13);textcolor(9);
443       cprintf("kh_bot<g>   :%6.2f", kh_bot);
444       gotoxy(21,14);textcolor(9);
445       cprintf("dv_bot<v>   :%6.2f", dv_bot);
446       gotoxy(42,14);textcolor(9);
447       cprintf("dh_bot<d>   :%6.2f", dh_bot);
448       gotoxy(21,17);textcolor(9);
449       cprintf("offset_bot<t>                        :");
450       gotoxy(55,17);textcolor(9);
451       cprintf("%5d", tBias_bot);
452       gotoxy(21,18);textcolor(9);
453       cprintf("offset_bot<w>                        :");
454       gotoxy(55,18);textcolor(9);
455       cprintf("%5d", wBias_bot);
456       gotoxy(21,19);textcolor(9);
457       cprintf("offset current_bot<b>                :");
458       gotoxy(55,19);textcolor(9);
459       cprintf("%6.2f Amp.", ibias_bot);
460       gotoxy(51,20);textcolor(15);
461       cprintf("x_value         y_value");
462       gotoxy(51,21);textcolor(4);
463       cprintf("=======         =======");
464       gotoxy(49,24);textcolor(15);
```

```
465         cprintf("  +       -       +       -  ");
466         gotoxy(49,25);textcolor(15);
467         cprintf("  X       X       Y       Y  ");
468         gotoxy(64, 7);textcolor(11);cprintf("Display Parameter");
469         gotoxy(64, 8);textcolor(15);cprintf("================");
470         gotoxy(64, 9);textcolor(13);cprintf("< >Lower Bearing");
471         gotoxy(65, 9);textcolor(15);cprintf("l");
472         gotoxy(64,10);textcolor(13);cprintf("< >Upper Bearing");
473         gotoxy(65,10);textcolor(15);cprintf("u");
474         gotoxy(64,11);textcolor(13);cprintf("< >Thrst Bearing");
475         gotoxy(65,11);textcolor(15);cprintf("z");
476         gotoxy(64,13);textcolor(11);cprintf("Energizing Parmtr");
477         gotoxy(64,14);textcolor(15);cprintf("================");
478         gotoxy(64,15);textcolor(13);cprintf("< >Thrst Bearing");
479         gotoxy(65,15);textcolor(15+128);cprintf("H");
480         gotoxy(64,16);textcolor(13);cprintf("< >Upper Bearing");
481         gotoxy(65,16);textcolor(15+128);cprintf("I");
482         gotoxy(64,17);textcolor(13);cprintf("< >Lower Bearing");
483         gotoxy(65,17);textcolor(15+128);cprintf("J");
484         gotoxy(26,20);textcolor(15);
485         cprintf("Force (N)");
486         gotoxy(25,21);textcolor( 4);
487         cprintf("==========");
488         gotoxy(1,20);textcolor(15);
489         cprintf("<x>Frq_inpt:%7.1f Hz.");
490         gotoxy(1,25);textcolor(15);
491         cprintf("<s>to adjust Pulse Width");
492         gotoxy(1,15);textcolor(15);
493         cprintf("[                ]");
494         gotoxy(1,14);textcolor(10);
495         cprintf("< >PHSE ANG:%3u deg",th);
496         gotoxy(2,14);textcolor(15);
497         cprintf("n");
498         gotoxy(2,15);textcolor(14);
499         cprintf("Lwr Safe Gain   ");
500         gotoxy(16,15);textcolor(10);
501         cprintf("ON ");
502         gotoxy(1,16);textcolor(15);
503         cprintf("[                ]");
504         gotoxy(2,16);textcolor(14);
505         cprintf("Upr Safe Gain   ");
506         gotoxy(16,16);textcolor(10);
507         cprintf("ON ");
508         gotoxy(1,17);textcolor(15);
509         cprintf("[                ]");
510         gotoxy(2,17);textcolor(14);
511         cprintf("Tht Safe Gain   ");
512         gotoxy(16,17);textcolor(10);
513         cprintf("ON ");
514         gotoxy(1,18);textcolor(15);
515         cprintf("[                ]");
516         gotoxy(2,18);textcolor(14);
517         cprintf("< >MODAL CTRL   ");
518         gotoxy(3,18);textcolor(15+128);
519         cprintf("m");
520         gotoxy(16,18);textcolor(12+128);
521         cprintf("OFF");
522         gotoxy(1,19);textcolor(15);
```

```
523         cprintf("[                  ]");
524         gotoxy(2,19);textcolor(14);
525         cprintf("< >EXCITATION    ");
526         gotoxy(16,19);textcolor(12+128);
527         cprintf("OFF");
528         gotoxy(1,22);textcolor(13);
529         cprintf("<Excitation Parmtr>");
530         gotoxy(1,23);textcolor(15);
531         cprintf("<o>freq:%8.2f Hz",frequency);
532         gotoxy(1,24);textcolor(15);
533         cprintf("<a>Amplitude:%4.1f v O-pk",volt);
534         gotoxy(27,23);textcolor(14);
535         cprintf("[<^> to toggle D.A. ]");
536         gotoxy(27,24);textcolor(14);
537         cprintf("[<,> Enable exction.]");
538         gotoxy(28,25);textcolor(14);
539         cprintf("[<:> Assembly    ]");
540         gotoxy(42,25);textcolor(10);
541         cprintf("ON");
542       }// End if (diag == 1)
543         gotoxy(27, 9);textcolor(10);
544         cprintf("[ loop time:       micro-sec ]");
545         gotoxy(1, 8);textcolor(15);cprintf("[   THE MAGNETIC  ]");
546         gotoxy(1, 9);textcolor(15);cprintf("[BEARING SYSTEM IS]");
547         gotoxy(1,10);textcolor(15);cprintf("[                 ]");
548         gotoxy(9,11);textcolor(15);cprintf("|");
549         gotoxy(9,12);textcolor(15);cprintf("|");
550
551         if(flag4 == 0)
552         {
553          gotoxy(4,10);textcolor(12+128);
554          cprintf("OPERATIONAL !  ");
555         }
556         else
557         {
558           gotoxy(4,10);textcolor(12+128);
559           cprintf("OPERATIONAL !\a  ");
560         }
561
562         if(diag == 1)
563         {
564           flag_CC = 1;
565           flag1  = 0;
566           flag4a = 1;// Turn on Lower Bearing buffer
567           gotoxy(48,4);textcolor(14+128);
568           cprintf("  * Lower bearing not energized !");
569
570           flag_DD = 1;
571           flag2  = 0;
572           flag4b = 1;// Turn on Upper Bearing buffer
573           gotoxy(48,3);textcolor(14+128);
574           cprintf("  * Upper bearing not energized !");
575
576           flag_EE = 1;
577           flag3  = 0;
578           flag4c = 1;// Turn on Thrust Bearing buffer
579           gotoxy(48,2);textcolor(14+128);
580           cprintf("  * Thrst bearing not energized !");
```

```
581         }
582         else
583
584         if (diag == 0)
585         {
586           gotoxy(31,8);textcolor(9);
587           cprintf(" <c>CG factor: %5.2f",CG);
588           gotoxy(26,13);textcolor(14);
589           cprintf("==>                          <==");
590           gotoxy(30,13);textcolor(12+128);
591           cprintf("THRST BEARING ENERGIZED");
592           gotoxy(26,14);textcolor(14);
593           cprintf("==>                          <==");
594           gotoxy(30,14);textcolor(12+128);
595           cprintf("UPPER BEARING ENERGIZED");
596           gotoxy(26,15);textcolor(14);
597           cprintf("==>                          <==");
598           gotoxy(30,15);textcolor(12+128);
599           cprintf("LOWER BEARING ENERGIZED");
600         }
601           gotoxy(1,15);textcolor(15);
602           cprintf("[                  ]");
603           gotoxy(2,15);textcolor(14);
604           cprintf("Lwr Safe Gain   ");
605           gotoxy(16,15);textcolor(10);
606           cprintf("ON ");
607           gotoxy(1,16);textcolor(15);
608           cprintf("[                  ]");
609           gotoxy(2,16);textcolor(14);
610           cprintf("Upr Safe Gain   ");
611           gotoxy(16,16);textcolor(10);
612           cprintf("ON ");
613           gotoxy(1,17);textcolor(15);
614           cprintf("[                  ]");
615           gotoxy(2,17);textcolor(14);
616           cprintf("Tht Safe Gain   ");
617           gotoxy(16,17);textcolor(10);
618           cprintf("ON ");
619           gotoxy(1,18);textcolor(15);
620           cprintf("[                  ]");
621           gotoxy(2,18);textcolor(14);
622           cprintf("< >MODAL CTRL   ");
623           gotoxy(3,18);textcolor(15+128);
624           cprintf("m");
625           gotoxy(16,18);textcolor(12+128);
626           cprintf("OFF");
627           gotoxy(1,19);textcolor(15);
628           cprintf("[                  ]");
629           gotoxy(2,19);textcolor(14);
630           cprintf("< >EXCITATION    ");
631           gotoxy(16,19);textcolor(12+128);
632           cprintf("OFF");
633
634   C:
635
636   // --------------------------- CONTROL LOOP -----------------------------
637
638   loop:
```

```
639                 i_bot=1;i_top=1;i_th=1; n=0;
640   while (n <= nmax)
641   {
642     if(diag == 0)
643     {
644       if (n == 1)
645       {
646         gotoxy(cir-1, 21);
647         textcolor(9); cprintf(" >>> ");
648         if(cir == 52)
649         {
650           gotoxy(cir-1, 21);
651           cir = 25;
652         }// End of if(cir == 52)
653         cir++;
654   //    ******************************
655         gotoxy(cir2, 21);
656         textcolor(9); cprintf(" <<< ");
657         if(cir2 == 25)
658         {
659           gotoxy(cir2, 21);
660           cir2 = 52;
661         }// End of if(cir2 == 25)
662         cir2--;
663       }// End of if(n == 1)
664     }// End of if(diag == 0)
665
666   // ******************** Datel Board data input block ************************
667
668         if(flag16 == 0)// Non assembly condition
669         {
670           xbot          =  - inport(FIFO1);// - x0;// Channel 0
671           ybot          =    inport(FIFO1);// + x0;// Channel 1
672
673           xtop          =  - inport(FIFO1);// - x0;// Channel 2
674           ytop          =    inport(FIFO1);// + x0;// Channel 3
675
676   //      ================================================
677
678           zth1          =  - inport(FIFO2);// - x0;// Channel 0
679           zth2          =    inport(FIFO2);// + x0;// Channel 1
680
681           one_per_rev =      inport(FIFO2);// + x0;// Channel 2
682           f_excite2   =      inport(FIFO2);// + x0;// Channel 3
683         }
684
685         else
686
687         if(flag16 == 1)// Activates assembly block
688         {
689           asm{
690                mov dx, [FIFO1]// Channel 0
691                in  ax, dx
692                neg ax
693                sub ax, [x0]
694                mov [xbot], ax
695              }
696           asm{
```

```
697                mov dx, [FIFO1]// Channel 1
698                in  ax, dx
699                add ax, [x0]
700                mov [ybot], ax
701             }
702          asm{
703                mov dx, [FIFO1]// Channel 2
704                in  ax, dx
705                neg ax
706                sub ax, [x0]
707                mov [xtop], ax
708             }
709          asm{
710                mov dx, [FIFO1]// Channel 3
711                in  ax, dx
712                add ax, [x0]
713                mov [ytop], ax
714             }
715          asm{
716                mov dx, [FIFO2]// Channel 0
717                in  ax, dx
718                neg ax
719                sub ax, [x0]
720                mov [zth1], ax
721             }
722          asm{
723                mov dx, [FIFO2]// Channel 1
724                in  ax, dx
725                add ax, [x0]
726                mov [zth2], ax
727             }
728          asm{
729                mov dx, [FIFO2]// Channel 2
730                in  ax, dx
731                mov [one_per_rev], ax
732             }
733          asm{
734                mov dx, [FIFO2]// Channel 3
735                in  ax, dx
736                mov [f_excite2], ax
737             }
738        }
739
740  // ***************** End Datel Board data input block **********************
741
742  // ********************* Signal generation block ***************************
743
744      if(switch1 == 0)// Shuts down excitation function block when an
745      {                 // external excitation (switch=1) source is used
746        if(flag5 == 1)// <4>
747        {
748          if(flag_AA == 1)
749          {
750            f_ex = t04 * sin(O*x*2.0*M_PI);// Sine
751          }
752
753          else
754
```

```
755         if(flag_AA == 2)
756         {
757           f_ex = t04 * pow(sin(O*x/*2.0*/*M_PI),2);// Sine squared
758         }
759
760         else
761
762         if(flag_AA == 3)
763         {
764           f_ex = t04 * cos(O*x*2.0*M_PI);// Cosine
765         }
766
767         else
768
769         if(flag_AA == 4)
770         {
771           f_ex = t04 * pow(cos(O*x/*2.0*/*M_PI),2);// Cosine squared
772         }
773
774         else
775
776         if(flag_AA == 5)
777         {
778           xy = xy + 1.0;
779           srand(xy);
780
781           if(flag21 == 1)// Excitation switch
782           {
783             for(i = 1; i <= 2; i++)
784             {
785               f_excite4  = (float(rand())/RAND_MAX);
786             }
787             for(i = 1; i <= 2/*NUMBERS*/; i++)
788             {
789               f_excite3 = float(rand())/RAND_MAX;
790             }
791           }// End of if(flag21 == 1)
792
793           if(xy >= LIM)
794              xy = 0.0;
795
796              f_ex = t04 * sin(2.0*M_PI*f_excite3)*(sin(O*x*2.0*M_PI) +
797                           sin(O*2.0*M_PI*f_excite4));// Random sine
798         }// End of if(flag_AA == 5)
799         {
800           g = ceil(f_ex);
801           z = f_ex + 0.5;
802
803           if(g >= z)
804              v = floor(f_ex);
805           else
806              v = g;
807
808           if(flag21 == 1)// Excitation On/Off switch
809           {
810             f_excite = v;
811           }
812         }
```

```
813             x = x + 0.002;
814         }// End of if(flag5 == 1)// <4>
815
816         else
817
818         if(flag6 == 1)// <5>
819         {
820           while (k <= 40)// Forty terms in series
821           {
822             ex = ex + (1.0/(2.0*k+1.0))*sin(2.0*(2.0*k+1)*O*M_PI*x);
823             k++;                              // Square wave pulse train
824           }
825             f_ex = t04 + t04 * (4.0/M_PI) * ex;
826           {
827             g = ceil(f_ex);
828             z = f_ex + 0.5;
829
830             if(g >= z)
831                v = floor(f_ex);
832             else
833                v = g;
834
835             if(flag21 == 1)// Excitation On/Off switch
836             {
837               f_excite = v / 2;
838             }
839           }
840               x = x + 0.002;
841               k = 0;
842         }// End of if(flag6 == 1)// <5>
843
844         else
845
846         if(flag7 == 1)// <6>
847         {
848           while (k <= 40)// Forty terms in series
849           {
850             ex = ex + (1.0/(2.0*k+1.0))*sin(2.0*(2.0*k+1)*O*M_PI*x);
851             k++;                                          // Square wave
852           }
853             f_ex = t04 * (4.0/M_PI) * ex;
854           {
855             g = ceil(f_ex);
856             z = f_ex + 0.5;
857
858             if(g >= z)
859                v = floor(f_ex);
860             else
861                v = g;
862
863             if(flag21 == 1)// Excitation On/Off switch
864             {
865               f_excite = v;
866             }
867           }
868               x = x + 0.002;
869               k = 0;
870         }// End of if(flag7 == 1)// <6>
```

```
871
872         else
873
874         if(flag8 == 1)// <7>
875         {
876           while (k <= 40)// Forty terms in series
877           {
878             ex = ex + (pow(-1,k)/pow((2.0*k+1.0),2))*sin(2.0*(2*k+1)*O*M_PI*x);
879             k++;                                                    // Saw tooth
880           }
881             f_ex = t04 * (8.0/pow(M_PI,2)) * ex;
882           {
883             g = ceil(f_ex);
884             z = f_ex + 0.5;
885
886             if(g >= z)
887                v = floor(f_ex);
888             else
889                v = g;
890
891             if(flag21 == 1)// Excitation On/Off switch
892             {
893               f_excite = v;
894             }
895           }
896               x = x + 0.002;
897               k = 0;
898         }// End of if(flag8 == 1)// <7>
899
900         else
901
902         if(flag9 == 1)// <8>
903         {
904           C = PW;
905           while (k1 <= 40)// Forty terms in series
906           {
907             ex = ex + (pow(-1,k1)/k1)*sin(k1*O*M_PI*C)*cos(2.0*k1*O*M_PI*x);
908             k1++;                                          // Single square pulse
909           }
910             f_ex = t04 * (O * C + (2.0/M_PI) * ex);
911           {
912             g = ceil(f_ex);
913             z = f_ex + 0.5;
914
915             if(g >= z)
916                v = floor(f_ex);
917             else
918                v = g;
919
920             if(flag21 == 1)// Excitation switch
921             {
922               f_excite = v;
923             }
924           }
925               x = x + 0.002;
926               k1 = 1;
927         }// End of if(flag9 == 1)// <8>
928
```

```
929         else
930
931         if(flag12 == 1)// <9>
932         {
933           while (k <= 40)// Forty terms in series
934           {
935             ex = ex + (1.0/pow((2.0*k+1.0),2))*cos(2.0*(2.0*k+1.0)*O*M_PI*x);
936             k++;                                          // Single triangular pulse
937           }
938             f_ex = t04 * (0.5 - (4.0/pow(M_PI,2)) * ex);
939           {
940             g = ceil(f_ex);
941             z = f_ex + 0.5;
942
943             if(g >= z)
944                v = floor(f_ex);
945             else
946                v = g;
947
948             if(flag21 == 1)// Excitation On/Off switch
949             {
950               f_excite = v;
951             }
952           }
953               x = x + 0.002;
954               k = 0;
955         }// End of if(flag12 == 1)// <9>
956
957         else
958
959         if(flag13 == 1)// <0>
960         {
961           while (k1 <= 40)// Forty terms in series
962           {
963             ex = ex + (pow(-1,(k1+1))/(k1*1.0))*sin(2.0*k1*O*M_PI*x);// Saw tooth
964 //          ex = ex + (1/k1)*sin(k1*O*M_PI*x);
965             k1++;
966           }
967             f_ex = t04 * (2.0/M_PI) * ex;
968 //          f_ex = t04 * (0.5 - 1.0/M_PI * ex);
969           {
970             g = ceil(f_ex);
971             z = f_ex + 0.5;
972
973             if(g >= z)
974                v = floor(f_ex);
975             else
976                v = g;
977
978             if(flag21 == 1)// Excitation On/Off switch
979             {
980               f_excite = v;
981             }
982           }
983               x = x + 0.002;
984               k1 = 1;
985       }// End of if(flag13 == 1)// <0>
986     }// End of if(switch1 == 0)
```

```
987
988  // ********************* End of signal generation block ********************
989
990  // ********************* External Excitation input Block ********************
991
992        if(switch1 == 1)// External excitation flag
993        {
994          if(flag21 == 1)// Excitation On/Off switch
995          {
996            f_excite = f_excite2;// Datel input channel #3 on board #2
997          }
998        }
999
1000 //    ***************** End of External Excitation Block ***************
1001
1002 //           ******************************************************
1003 //           * This block is used to output the excitation signal *
1004 //           ******************************************************
1005
1006       if(test_signal == 1)
1007       {
1008         if(flag16 == 0)
1009            outport(out_chan1_0,(f_excite + t48));// Board 1
1010
1011         else
1012
1013         if(flag16 == 1)
1014         {
1015           asm{
1016                mov dx, [out_chan1_0]
1017                mov ax, [f_excite]
1018                add ax, [t48]
1019                out dx, ax
1020         }
1021       }
1022 /*
1023         if(flag16 == 0)
1024            outport(out_chan2_0, (f_excite + t48);// Board 2
1025
1026         else
1027
1028         if(flag16 == 1)
1029         {
1030         asm{
1031              mov dx, [out_chan2_0]
1032              mov ax, [f_excite]
1033              out dx, ax
1034            }
1035         }
1036 */
1037       }// End of if(test_signal == 1)
1038
1039 //    ************** End of signal generator block **************
1040
1041 //                  ***********************************
1042 //                  * This block is used to generate *
1043 //                  * the One - Per - Rev  signal    *
1044 //                  ***********************************
```

```
1045
1046        if(flag_II == 1)// one_per_rev set to on
1047        {
1048          THETA = II * (PI2_o_Nticks * i_rev);
1049        }
1050
1051        {
1052          f_excite_cos =  f_excite * cos(THETA);// X - AXIS
1053          f_excite_sin =  f_excite * sin(ns*THETA);// Y - AXIS
1054        }
1055
1056        if(flag18 == 1)
1057        {
1058          delay(60);// Delay 60 milli sec. - used for diagnostic purposes
1059
1060          if(diag == 1)// Display # of period length(s) only in diagnostic mode
1061          {
1062            gotoxy(25,22);textcolor(11);
1063            cprintf("%5.1E     ",x - 0.002);
1064            gotoxy(25,23);
1065            printf("           ");// Erase y: display
1066          }
1067            gotoxy(14,25);textcolor(15);
1068            cprintf("%5d,%4u ",f_excite,n);
1069        }// End of if(flag18 == 1)
1070
1071        if(n == 500)// Test for maximum # of loops in one period length
1072        {
1073          x = x + 0.002;
1074
1075          if(x > LIM)
1076          {
1077            x = 0.0;// Resets x to zero
1078          }
1079        }// Ene of if(n == 500)
1080            ex = 0;// Summed ex values zeroed
1081
1082        if(flag_II == 1)// one_per_rev set to on
1083        {
1084          if(one_per_rev < trigger)// No pulse condition, One_per_rev is < 0.1v
1085             rise = 1;
1086          if(rise == 1)
1087          if(one_per_rev >= trigger)// --> A pulse
1088          {
1089            rise = 0;
1090            N_ticks = j_rev;// # of loops in one revolution of the shaft
1091            if(N_ticks == 0)
1092               N_ticks = 1;
1093
1094            PI2_o_Nticks = PI2/N_ticks;// Shaft radians per loop
1095            i_rev1 = (phi/360.0) * N_ticks;// phi: (0.0 --> 360.0) deg.
1096            {
1097              g = ceil(i_rev1);
1098              z = i_rev1 + 0.5;
1099
1100              if(g >= z)
1101                 v = floor(i_rev1);
1102              else
```

```
1103                v = g;
1104
1105                i_rev = v;// After one shaft rotation i_rev = 0 if phi = 0
1106             }
1107                j_rev = 0;// After one revolution of shaft.
1108          }// End of if (one_per_rev >= trigger).
1109
1110                i_rev++;  // Loop counter for one shaft rotation
1111                          // used to calculate (THETA).
1112
1113                j_rev++;  // Loop counter for one shaft rotation
1114                          // used to calculate (PI2_o_Nticks).
1115                if(i_rev > N_ticks)
1116                   i_rev = i_rev - N_ticks;
1117       }// End of if(flag_II == 1)
1118
1119 //    ************** End of One - Per - Rev block **************
1120
1121   if(flag16 == 0)// Non assembly condition.
1122   {
1123     // Commands board (1) to read next input value
1124     outport(0x300, one);
1125     outport(FIFO1, two);
1126     outport(0x300, 0xe);
1127
1128     // Commands board (2) to read next input value
1129     outport(0x360, one);
1130     outport(FIFO2, two);
1131     outport(0x360, 0xe);
1132   }
1133
1134   else
1135
1136   if(flag16 == 1)// Assembly condition
1137   {              // Commands board (1) to read next input value
1138     asm{
1139          mov dx, 0x300
1140          mov ax, [one]
1141          out dx, ax
1142        }
1143     asm{
1144          mov dx, [FIFO1]
1145          mov ax, [two]
1146          out dx, ax
1147        }
1148     asm{
1149          mov dx, 0x300
1150          mov ax, 0xe
1151          out dx, ax
1152        }
1153 // *************************
1154        // Commands board (2) to read next input value
1155     asm{
1156          mov dx, 0x360
1157          mov ax, [one]
1158          out dx, ax
1159        }
1160     asm{
```

```
1161          mov dx, [FIFO2]
1162          mov ax, [two]
1163          out dx, ax
1164        }
1165     asm{
1166          mov dx, 0x360
1167          mov ax, 0xe
1168          out dx, ax
1169        }
1170   }// End of if(flag16 == 1)
1171
1172 if(flag10 == 0)// Non modal condition
1173 {
1174 // ***************************** LOWER BEARING ****************************
1175
1176 if(flag1 == 1)
1177 {
1178
1179 //       * * * Begin x_force_bot calc * * *
1180
1181   xbotderiv   = xbot - x_bot_old3;
1182
1183 //       * * * Calculate x_force_bot * * *
1184
1185   x_force_bot = (((kh_bot * xbot + dh_bot * xbotderiv) * MCG)
1186                                    - tBias_bot) + f_excite_cos;
1187   x_pos_output_bot = - x_force_bot - bias_current_bot;
1188   x_neg_output_bot = - x_force_bot + bias_current_bot;
1189
1190 //       * * *  OUTPUTS FOR x_direction_bot  * * *
1191
1192 // **********ROUNDING BLOCK**********
1193        g = ceil(x_pos_output_bot);
1194        z = x_pos_output_bot + 0.5;
1195
1196        if(g >= z)
1197           v = floor(x_pos_output_bot);
1198        else
1199           v = g;
1200
1201        round2 = v + t48;
1202 // ***********************************
1203
1204      if(round2 < out_min)
1205      {
1206        if(flag16 == 0)
1207           outport(out_chan1_1, out_min);
1208
1209        else
1210
1211        if(flag16 == 1)
1212        {
1213          asm{
1214               mov dx, [out_chan1_1]
1215               mov ax, [out_min]
1216               out dx, ax
1217             }
1218        }
```

```
1219      }// End of if(round2 < out_min)
1220
1221      else
1222
1223      if(round2 > out_max)
1224      {
1225        if(flag16 == 0)
1226           outport(out_chan1_1, out_max);
1227
1228        else
1229
1230        if(flag16 == 1)
1231        {
1232          asm{
1233               mov dx, [out_chan1_1]
1234               mov ax, [out_max]
1235               out dx, ax
1236             }
1237        }
1238      }// End of if(round2 > out_max)
1239
1240      else
1241
1242      {
1243        if(flag16 == 0)
1244           outport(out_chan1_1, round2);// HORIZ.(RIGHT)
1245
1246        else
1247
1248        if(flag16 == 1)
1249        {
1250          asm{
1251               mov dx, [out_chan1_1]
1252               mov ax, [round2]
1253               out dx, ax
1254             }
1255        }
1256      }
1257 // **********ROUNDING BLOCK***********
1258        g = ceil(x_neg_output_bot);
1259        z = x_neg_output_bot + 0.5;
1260
1261        if(g >= z)
1262           v = floor(x_neg_output_bot);
1263        else
1264           v = g;
1265
1266        round2 = v + t48;
1267 // *********************************
1268
1269      if(round2 < out_min)
1270      {
1271        if(flag16 == 0)
1272           outport(out_chan1_2, out_min);
1273
1274        else
1275
1276        if(flag16 == 1)
```

```
1277        {
1278         asm{
1279              mov dx, [out_chan1_2]
1280              mov ax, [out_min]
1281              out dx, ax
1282             }
1283        }
1284      }// End of if(round2 < out_min)
1285
1286      else
1287
1288      if(round2 > out_max)
1289      {
1290        if(flag16 == 0)
1291           outport(out_chan1_2, out_max);
1292
1293        else
1294
1295        if(flag16 == 1)
1296        {
1297          asm{
1298               mov dx, [out_chan1_2]
1299               mov ax, [out_max]
1300               out dx, ax
1301             }
1302        }
1303      }// End of if(round2 > out_max)
1304
1305      else
1306
1307      {
1308        if(flag16 == 0)
1309           outport(out_chan1_2, round2);// HORIZ.(LEFT)
1310
1311        else
1312
1313        if(flag16 == 1)
1314        {
1315          asm{
1316               mov dx, [out_chan1_2]
1317               mov ax, [round2]
1318               out dx, ax
1319             }
1320        }
1321      }
1322
1323 // x_bot_old5 = x_bot_old4;
1324 // x_bot_old4 = x_bot_old3;
1325    x_bot_old3 = x_bot_old2;
1326    x_bot_old2 = x_bot_old1;
1327    x_bot_old1 = xbot;
1328
1329 //              * * * End x_force_bot * * *
1330
1331 //       * * * Begin y_force_bot calc * * *
1332
1333   ybotderiv   = ybot - y_bot_old3;
1334
```

```
1335 //       * * * Calculate y_force_bot * * *
1336
1337   y_force_bot = (((kv_bot * ybot + dv_bot * ybotderiv) * MCG)
1338                                     - wBias_bot) + f_excite_sin;
1339   y_pos_output_bot =  y_force_bot - bias_current_bot;
1340   y_neg_output_bot =  y_force_bot + bias_current_bot;
1341
1342 //                 * * * OUTPUTS FOR y_direction_bot  * * *
1343
1344 // **********ROUNDING BLOCK**********
1345        g = ceil(y_pos_output_bot);
1346        z = y_pos_output_bot + 0.5;
1347
1348        if(g >= z)
1349           v = floor(y_pos_output_bot);
1350        else
1351           v = g;
1352
1353        round2 = v + t48;
1354 // ***********************************
1355
1356      if(round2 < out_min)
1357      {
1358        if(flag16 == 0)
1359           outport(out_chan1_3, out_min);
1360
1361        else
1362
1363        if(flag16 == 1)
1364        {
1365          asm{
1366               mov dx, [out_chan1_3]
1367               mov ax, [out_min]
1368               out dx, ax
1369             }
1370        }
1371      }// End of if(round2 < out_min)
1372
1373      else
1374
1375      if(round2 > out_max)
1376      {
1377        if(flag16 == 0)
1378           outport(out_chan1_3, out_max);
1379
1380        else
1381
1382        if(flag16 == 1)
1383        {
1384          asm{
1385               mov dx, [out_chan1_3]
1386               mov ax, [out_max]
1387               out dx, ax
1388             }
1389        }
1390      }// End of if(round2 > out_max)
1391
1392      else
```

```
1393
1394      {
1395        if(flag16 == 0)
1396           outport(out_chan1_3, round2);// VERT.(TOP)
1397
1398        else
1399
1400        if(flag16 == 1)
1401        {
1402          asm{
1403               mov dx, [out_chan1_3]
1404               mov ax, [round2]
1405               out dx, ax
1406             }
1407        }
1408      }
1409 // **********ROUNDING BLOCK**********
1410        g = ceil(y_neg_output_bot);
1411        z = y_neg_output_bot + 0.5;
1412
1413        if(g >= z)
1414           v = floor(y_neg_output_bot);
1415        else
1416           v = g;
1417
1418        round2 = v + t48;
1419 // ************************************
1420
1421      if(round2 < out_min)
1422      {
1423        if(flag16 == 0)
1424           outport(out_chan1_4, out_min);
1425
1426        else
1427
1428        if(flag16 == 1)
1429        {
1430          asm{
1431               mov dx, [out_chan1_4]
1432               mov ax, [out_min]
1433               out dx, ax
1434             }
1435        }
1436      }// End of if(round2 < out_min)
1437
1438      else
1439
1440      if(round2 > out_max)
1441      {
1442        if(flag16 == 0)
1443           outport(out_chan1_4, out_max);
1444
1445        else
1446
1447        if(flag16 == 1)
1448        {
1449          asm{
1450               mov dx, [out_chan1_4]
```

```
1451                mov ax, [out_max]
1452                out dx, ax
1453              }
1454        }
1455      }// End of if(round2 > out_max)
1456
1457      else
1458
1459      {
1460        if(flag16 == 0)
1461           outport(out_chan1_4, round2);// VERT.(BOTTOM)
1462
1463        else
1464
1465        if(flag16 == 1)
1466        {
1467          asm{
1468                mov dx, [out_chan1_4]
1469                mov ax, [round2]
1470                out dx, ax
1471              }
1472        }
1473      }
1474
1475 // y_bot_old5 = y_bot_old4;
1476 // y_bot_old4 = y_bot_old3;
1477    y_bot_old3 = y_bot_old2;
1478    y_bot_old2 = y_bot_old1;
1479    y_bot_old1 = ybot;
1480
1481 //         * * * End y_force_bot * * *
1482
1483 //                        * * * Safe Gain * * *
1484      if (sg1 == 1)
1485      goto L1;
1486
1487      else
1488
1489      goto L2;
1490
1491 L1:  {
1492        if ((xbot * xbot + ybot * ybot) > safe)
1493        {
1494          kh_bot = 1.5; kv_bot = kh_bot;
1495          dh_bot = 9.0; dv_bot = dh_bot;
1496        }
1497        goto L2;
1498      }
1499 //                        * * * End Safe Gain * * *
1500
1501 }// End of if(flag1 == 1)
1502
1503 L2:
1504
1505 if(diag == 1)
1506 {
1507   if(flag4d == 1)
1508   {
```

```
1509      if(flag4a == 1)
1510      {
1511 //     junk = exp(1.34567);
1512 //     junk = exp(1.34567);
1513        junk = exp(1.34567);
1514        junk = exp(1.34567);
1515        junk = exp(1.34567);
1516        junk = exp(1.34567);
1517        junk = cos(1.34567);
1518        junk = cos(1.34567);
1519 //     junk = cos(1.34567);
1520 //     junk = cos(1.34567);
1521 //     junk = cos(1.34567);
1522      }// End of if(flag4a == 1)
1523    }// End of if(flag4d == 1)
1524 }// End of if(diag == 1)
1525
1526 // *************************** UPPER BEARING *******************************
1527
1528 if(flag2 == 1)
1529 {
1530 //       * * * Begin x_force_top calc * * *
1531
1532   xtopderiv = xtop - x_top_old3;
1533
1534 //       * * * Calculate x_force_top * * *
1535
1536   x_force_top = (((kh_top * xtop + dh_top * xtopderiv) * PCG)
1537                                - tBias_top) + JJ * f_excite_cos;
1538   x_pos_output_top = - x_force_top - bias_current_top;
1539   x_neg_output_top = - x_force_top + bias_current_top;
1540
1541 //       * * *  OUTPUTS FOR x_direction_top  * * *
1542
1543 // **********ROUNDING BLOCK**********
1544        g = ceil(x_pos_output_top);
1545        z = x_pos_output_top + 0.5;
1546
1547        if(g >= z)
1548           v = floor(x_pos_output_top);
1549        else
1550           v = g;
1551
1552        round2 = v + t48;
1553 // **********************************
1554
1555      if(round2 < out_min)
1556      {
1557        if(flag16 == 0)
1558           outport(out_chan2_1, out_min);
1559
1560        else
1561
1562        if(flag16 == 1)
1563        {
1564          asm{
1565               mov dx, [out_chan2_1]
1566               mov ax, [out_min]
```

```
1567                out dx, ax
1568              }
1569          }
1570       }// End of if(round2 < out_min)
1571
1572       else
1573
1574       if(round2 > out_max)
1575       {
1576         if(flag16 == 0)
1577            outport(out_chan2_1, out_max);
1578
1579         else
1580
1581         if(flag16 == 1)
1582         {
1583          asm{
1584               mov dx, [out_chan2_1]
1585               mov ax, [out_max]
1586               out dx, ax
1587             }
1588         }
1589       }// End of if(round2 > out_max)
1590
1591       else
1592
1593       {
1594         if(flag16 == 0)
1595            outport(out_chan2_1, round2);// HORIZ.(RIGHT)
1596
1597         else
1598
1599         if(flag16 == 1)
1600         {
1601           asm{
1602                mov dx, [out_chan2_1]
1603                mov ax, [round2]
1604                out dx, ax
1605              }
1606         }
1607       }
1608 // **********ROUNDING BLOCK**********
1609         g = ceil(x_neg_output_top);
1610         z = x_neg_output_top + 0.5;
1611
1612         if(g >= z)
1613            v = floor(x_neg_output_top);
1614         else
1615            v = g;
1616
1617         round2 = v + t48;
1618 // ***********************************
1619
1620       if(round2 < out_min)
1621       {
1622         if(flag16 == 0)
1623            outport(out_chan2_2, out_min);
1624
```

```
1625        else
1626
1627        if(flag16 == 1)
1628        {
1629          asm{
1630               mov dx, [out_chan2_2]
1631               mov ax, [out_min]
1632               out dx, ax
1633             }
1634        }
1635      }// End of if(round2 < out_min)
1636
1637      else
1638
1639      if(round2 > out_max)
1640      {
1641        if(flag16 == 0)
1642           outport(out_chan2_2, out_max);
1643
1644        else
1645
1646        if(flag16 == 1)
1647        {
1648          asm{
1649               mov dx, [out_chan2_2]
1650               mov ax, [out_max]
1651               out dx, ax
1652             }
1653        }
1654      }// End of if(round2 > out_max)
1655
1656      else
1657
1658      {
1659        if(flag16 == 0)
1660           outport(out_chan2_2, round2);// HORIZ.(LEFT)
1661
1662        else
1663
1664        if(flag16 == 1)
1665        {
1666          asm{
1667               mov dx, [out_chan2_2]
1668               mov ax, [round2]
1669               out dx, ax
1670             }
1671        }
1672      }
1673
1674 // x_top_old5 = x_top_old4;
1675 // x_top_old4 = x_top_old3;
1676    x_top_old3 = x_top_old2;
1677    x_top_old2 = x_top_old1;
1678    x_top_old1 = xtop;
1679
1680 //         * * * End x_force_top * * *
1681
1682 //       * * * Begin y_force_top calc * * *
```

```
1683
1684   ytopderiv = ytop - y_top_old3;
1685
1686 //       * * * Calculate y_force_top * * *
1687
1688   y_force_top = (((kv_top * ytop + dv_top * ytopderiv) * PCG)
1689                                    - wBias_top) + f_excite_sin;
1690   y_pos_output_top =  y_force_top - bias_current_top;
1691   y_neg_output_top =  y_force_top + bias_current_top;
1692
1693 //       * * *  OUTPUTS FOR y_direction_top  * * *
1694
1695 // **********ROUNDING BLOCK**********
1696        g = ceil(y_pos_output_top);
1697        z = y_pos_output_top + 0.5;
1698
1699        if(g >= z)
1700           v = floor(y_pos_output_top);
1701        else
1702           v = g;
1703
1704        round2 = v + t48;
1705 // ************************************
1706
1707      if(round2 < out_min)
1708      {
1709        if(flag16 == 0)
1710           outport(out_chan2_3, out_min);
1711
1712        else
1713
1714        if(flag16 == 1)
1715        {
1716          asm{
1717               mov dx, [out_chan2_3]
1718               mov ax, [out_min]
1719               out dx, ax
1720             }
1721        }
1722      }
1723
1724      else
1725
1726      if(round2 > out_max)
1727      {
1728        if(flag16 == 0)
1729           outport(out_chan2_3, out_max);
1730
1731        else
1732
1733        if(flag16 == 1)
1734        {
1735          asm{
1736               mov dx, [out_chan2_3]
1737               mov ax, [out_max]
1738               out dx, ax
1739             }
1740        }
```

```
1741      }
1742
1743      else
1744
1745      {
1746        if(flag16 == 0)
1747           outport(out_chan2_3, round2);// VERT.(TOP)
1748
1749        else
1750
1751        if(flag16 == 1)
1752        {
1753          asm{
1754               mov dx, [out_chan2_3]
1755               mov ax, [round2]
1756               out dx, ax
1757             }
1758        }
1759      }
1760 // **********ROUNDING BLOCK***********
1761        g = ceil(y_neg_output_top);
1762        z = y_neg_output_top + 0.5;
1763
1764        if(g >= z)
1765           v = floor(y_neg_output_top);
1766        else
1767           v = g;
1768
1769        round2 = v + t48;
1770 // ***********************************
1771
1772      if(round2 < out_min)
1773      {
1774        if(flag16 == 0)
1775           outport(out_chan2_4, out_min);
1776
1777        else
1778
1779        if(flag16 == 1)
1780        {
1781          asm{
1782               mov dx, [out_chan2_4]
1783               mov ax, [out_min]
1784               out dx, ax
1785             }
1786        }
1787      }
1788
1789      else
1790
1791      if(round2 > out_max)
1792      {
1793        if(flag16 == 0)
1794           outport(out_chan2_4, out_max);
1795
1796        else
1797
1798        if(flag16 == 1)
```

```
1799        {
1800          asm{
1801               mov dx, [out_chan2_4]
1802               mov ax, [out_max]
1803               out dx, ax
1804             }
1805        }
1806      }
1807
1808      else
1809
1810      {
1811        if(flag16 == 0)
1812           outport(out_chan2_4, round2);// VERT.(BOTTOM)
1813
1814        else
1815
1816        if(flag16 == 1)
1817        {
1818          asm{
1819               mov dx, [out_chan2_4]
1820               mov ax, [round2]
1821               out dx, ax
1822             }
1823        }
1824      }
1825
1826 // y_top_old5 = y_top_old4;
1827 // y_top_old4 = y_top_old3;
1828    y_top_old3 = y_top_old2;
1829    y_top_old2 = y_top_old1;
1830    y_top_old1 = ytop;
1831
1832 //          * * * End y_force_top * * *
1833
1834 //                        * * * Safe Gain * * *
1835      if (sg2 == 1)
1836      goto U1;
1837
1838      else
1839
1840      goto U2;
1841
1842 U1:  {
1843        if ((xtop * xtop + ytop * ytop) > safe)
1844        {
1845          kh_top = 1.5; kv_top = kh_top;
1846          dh_top = 9.0; dv_top = dh_top;
1847        }
1848        goto U2;
1849      }
1850 //                        * * * End Safe Gain * * *
1851
1852 }// End of if(flag2 == 1)
1853
1854 U2:
1855
1856 if(diag == 1)
```

```
1857 {
1858   if(flag4d == 1)
1859   {
1860     if(flag4b == 1)
1861     {
1862 //    junk = exp(1.34567);
1863 //    junk = exp(1.34567);
1864       junk = exp(1.34567);
1865       junk = exp(1.34567);
1866       junk = exp(1.34567);
1867       junk = exp(1.34567);
1868       junk = cos(1.34567);
1869       junk = cos(1.34567);
1870 //    junk = cos(1.34567);
1871 //    junk = cos(1.34567);
1872 //    junk = cos(1.34567);
1873     }// End of if(diag == 1)
1874   }// End of if(flag4d == 1)
1875 }// End of if(flag4b == 1)
1876 }// End of if(flag10 == 0)
1877
1878 // ************************* MODAL CONTROL BLOCK **************************
1879
1880 if(flag10 == 1)// Modal condition
1881 {
1882 // ******************  Centralized Rigid Body Translation  ******************
1883
1884   Xav = xbot * MCG + xtop * PCG;
1885   Yav = ybot * MCG + ytop * PCG;
1886
1887   xbot_force_tr = (-(kh_bot + kh_top) * Xav - (dh_bot + dh_top) * dotXav);
1888   xtop_force_tr = (-(kh_bot + kh_top) * Xav - (dh_bot + dh_top) * dotXav);
1889
1890   ybot_force_tr = (-(kv_bot + kv_top) * Yav - (dv_bot + dv_top) * dotYav);
1891   ytop_force_tr = (-(kv_bot + kv_top) * Yav - (dv_bot + dv_top) * dotYav);
1892
1893   F_XB_tr = xbot_force_tr * MCG;// F1_X
1894   F_XT_tr = xtop_force_tr * PCG;// F2_X
1895
1896   F_YB_tr = ybot_force_tr * MCG;// F1_Y
1897   F_YT_tr = ytop_force_tr * PCG;// F2_Y
1898
1899 // ******************  Centralized Rigid Body Rotation  *******************
1900
1901   ThetaX = xbot - xtop;
1902   ThetaY = ybot - ytop;
1903
1904   k_tilt = kh_top * MCG * MCG + kh_bot * PCG * PCG;
1905   c_tilt = dh_top * MCG * MCG + dh_bot * PCG * PCG;
1906
1907   xtop_force_rot =  k_tilt * ThetaX + c_tilt * dotThetaX;
1908   xbot_force_rot = -k_tilt * ThetaX - c_tilt * dotThetaX;
1909
1910   ytop_force_rot =  k_tilt * ThetaY + c_tilt * dotThetaY;
1911   ybot_force_rot = -k_tilt * ThetaY - c_tilt * dotThetaY;
1912
1913 // *********************  Centralized force summed  ***********************
1914
```

```
1915   xbot_force_modal_pos =   F_XB_tr + xbot_force_rot  + bias_current_bot;
1916   xbot_force_modal_neg = -(F_XB_tr + xbot_force_rot) + bias_current_bot;
1917
1918   ybot_force_modal_pos =   F_YB_tr + ybot_force_rot  - bias_current_bot;
1919   ybot_force_modal_neg = -(F_YB_tr + ybot_force_rot) - bias_current_bot;
1920 //-------------------------------------------------------------------------
1921   xtop_force_modal_pos =   F_XT_tr + xtop_force_rot  + bias_current_top;
1922   xtop_force_modal_neg = -(F_XT_tr + xtop_force_rot) + bias_current_top;
1923
1924   ytop_force_modal_pos =   F_YT_tr + xtop_force_rot  - bias_current_top;
1925   ytop_force_modal_neg = -(F_YT_tr + xtop_force_rot) - bias_current_top;
1926 //-------------------------------------------------------------------------
1927   x_pos_output_bot = xbot_force_modal_pos + f_excite_cos * -1;
1928   x_neg_output_bot = xbot_force_modal_neg + f_excite_cos * -1;
1929
1930   y_pos_output_bot = ybot_force_modal_pos + f_excite_sin;
1931   y_neg_output_bot = ybot_force_modal_neg + f_excite_sin;
1932 //-------------------------------------------------------------------------
1933   x_pos_output_top = xtop_force_modal_pos + f_excite_cos * -1;
1934   x_neg_output_top = xtop_force_modal_neg + f_excite_cos * -1;
1935
1936   y_pos_output_top = ytop_force_modal_pos + JJ * f_excite_sin;
1937   y_neg_output_top = ytop_force_modal_neg + JJ * f_excite_sin;
1938
1939 // Note that f_excite_cos is multiplied by -1 to give
1940 // the correct One - Per - Rev vector rotation direction.
1941
1942 // ********** ROUNDING BLOCK - x_pos_output_bot **********
1943   g = ceil(x_pos_output_bot);
1944   z = x_pos_output_bot + 0.5;
1945
1946   if(g >= z)
1947      v = floor(x_pos_output_bot);
1948   else
1949      v = g;
1950
1951      X_P_O_B = v + t48;
1952 //    ************
1953   if(X_P_O_B < out_min)
1954   {
1955     if(flag16 == 0)
1956        outport(out_chan1_1, out_min);
1957
1958     else
1959
1960     if(flag16 == 1)
1961     {
1962       asm{
1963            mov dx, [out_chan1_1]
1964            mov ax, [out_min]
1965            out dx, ax
1966          }
1967     }
1968   }// End of if(X_P_O_B < out_min)
1969
1970   else
1971
1972   if(X_P_O_B > out_max)
```

```
1973    {
1974      if(flag16 == 0)
1975         outport(out_chan1_1, out_max);
1976
1977      else
1978
1979      if(flag16 == 1)
1980      {
1981        asm{
1982             mov dx, [out_chan1_1]
1983             mov ax, [out_max]
1984             out dx, ax
1985           }
1986      }
1987    }// End of if(X_P_O_B > out_max)
1988
1989    else
1990
1991    {
1992      if(flag16 == 0)
1993         outport(out_chan1_1, X_P_O_B);
1994
1995      else
1996
1997      if(flag16 == 1)
1998      {
1999        asm{
2000             mov dx, [out_chan1_1]
2001             mov ax, [X_P_O_B]
2002             out dx, ax
2003           }
2004      }
2005    }
2006  // ********** ROUNDING BLOCK - x_neg_output_bot **********
2007    g = ceil(x_neg_output_bot);
2008    z = x_neg_output_bot + 0.5;
2009
2010    if(g >= z)
2011       v = floor(x_neg_output_bot);
2012    else
2013       v = g;
2014
2015       X_N_O_B = v + t48;
2016  //     ************
2017    if(X_N_O_B < out_min)
2018    {
2019      if(flag16 == 0)
2020         outport(out_chan1_2, out_min);
2021
2022      else
2023
2024      if(flag16 == 1)
2025      {
2026        asm{
2027             mov dx, [out_chan1_2]
2028             mov ax, [out_min]
2029             out dx, ax
2030           }
```

```
2031      }
2032    }// End of if(X_N_O_B < out_min)
2033
2034    else
2035
2036    if(X_N_O_B > out_max)
2037    {
2038      if(flag16 == 0)
2039         outport(out_chan1_2, out_max);
2040
2041      else
2042
2043      if(flag16 == 1)
2044      {
2045        asm{
2046             mov dx, [out_chan1_2]
2047             mov ax, [out_max]
2048             out dx, ax
2049           }
2050      }
2051    }// End of if(X_N_O_B > out_max)
2052
2053    else
2054
2055    {
2056      if(flag16 == 0)
2057         outport(out_chan1_3, X_N_O_B);
2058
2059      else
2060
2061      if(flag16 == 1)
2062      {
2063        asm{
2064             mov dx, [out_chan1_2]
2065             mov ax, [X_N_O_B]
2066             out dx, ax
2067           }
2068      }
2069    }
2070 // ********** ROUNDING BLOCK - y_pos_output_bot **********
2071    g = ceil(y_pos_output_bot);
2072    z = y_pos_output_bot + 0.5;
2073
2074    if(g >= z)
2075       v = floor(y_pos_output_bot);
2076    else
2077       v = g;
2078
2079       Y_P_O_B = v + t48;
2080 //     ************
2081    if(Y_P_O_B < out_min)
2082    {
2083      if(flag16 == 0)
2084         outport(out_chan1_3, out_min);
2085
2086      else
2087
2088      if(flag16 == 1)
```

```
2089      {
2090        asm{
2091             mov dx, [out_chan1_3]
2092             mov ax, [out_min]
2093             out dx, ax
2094           }
2095      }
2096    }// End of if(Y_P_O_B < out_min)
2097
2098    else
2099
2100    if(Y_P_O_B > out_max)
2101    {
2102      if(flag16 == 0)
2103         outport(out_chan1_3, out_max);
2104
2105      else
2106
2107      if(flag16 == 1)
2108      {
2109        asm{
2110             mov dx, [out_chan1_3]
2111             mov ax, [out_max]
2112             out dx, ax
2113           }
2114      }
2115    }// End of if(Y_P_O_B > out_max)
2116
2117    else
2118
2119    {
2120      if(flag16 == 0)
2121         outport(out_chan1_3, Y_P_O_B);
2122
2123      else
2124
2125      if(flag16 == 1)
2126      {
2127        asm{
2128             mov dx, [out_chan1_3]
2129             mov ax, [Y_P_O_B]
2130             out dx, ax
2131           }
2132      }
2133    }
2134 // ********** ROUNDING BLOCK - y_neg_output_bot **********
2135    g = ceil(y_neg_output_bot);
2136    z = y_neg_output_bot + 0.5;
2137
2138    if(g >= z)
2139       v = floor(y_neg_output_bot);
2140    else
2141       v = g;
2142
2143       Y_N_O_B = v + t48;
2144 //     ************
2145    if(Y_N_O_B < out_min)
2146    {
```

```
2147      if(flag16 == 0)
2148         outport(out_chan1_4, out_min);
2149
2150      else
2151
2152      if(flag16 == 1)
2153      {
2154        asm{
2155             mov dx, [out_chan1_4]
2156             mov ax, [out_min]
2157             out dx, ax
2158           }
2159      }
2160    }// End of if(Y_N_O_B < out_min)
2161
2162    else
2163
2164    if(Y_N_O_B > out_max)
2165    {
2166      if(flag16 == 0)
2167         outport(out_chan1_4, out_max);
2168
2169      else
2170
2171      if(flag16 == 1)
2172      {
2173        asm{
2174             mov dx, [out_chan1_4]
2175             mov ax, [out_max]
2176             out dx, ax
2177           }
2178      }
2179    }// End of if(Y_N_O_B > out_max)
2180
2181    else
2182
2183    {
2184      if(flag16 == 0)
2185         outport(out_chan1_4, Y_N_O_B);
2186
2187      else
2188
2189      if(flag16 == 1)
2190      {
2191        asm{
2192             mov dx, [out_chan1_4]
2193             mov ax, [Y_N_O_B]
2194             out dx, ax
2195           }
2196      }
2197    }
2198  // ********** ROUNDING BLOCK - x_pos_output_top **********
2199    g = ceil(x_pos_output_top);
2200    z = x_pos_output_top + 0.5;
2201
2202    if(g >= z)
2203       v = floor(x_pos_output_top);
2204    else
```

```
2205      v = g;
2206
2207      X_P_O_T = v + t48;
2208 //    ************
2209   if(X_P_O_T < out_min)
2210   {
2211     if(flag16 == 0)
2212        outport(out_chan2_1, out_min);
2213
2214     else
2215
2216     if(flag16 == 1)
2217     {
2218       asm{
2219            mov dx, [out_chan2_1]
2220            mov ax, [out_min]
2221            out dx, ax
2222          }
2223     }
2224   }// End of if(X_P_O_T < out_min)
2225
2226   else
2227
2228   if(X_P_O_T > out_max)
2229   {
2230     if(flag16 == 0)
2231        outport(out_chan2_1, out_max);
2232
2233     else
2234
2235     if(flag16 == 1)
2236     {
2237       asm{
2238            mov dx, [out_chan2_1]
2239            mov ax, [out_max]
2240            out dx, ax
2241          }
2242     }
2243   }// End of if(X_P_O_T > out_max)
2244
2245   else
2246
2247   {
2248     if(flag16 == 0)
2249        outport(out_chan2_1, X_P_O_T);
2250
2251     else
2252
2253     if(flag16 == 1)
2254     {
2255       asm{
2256            mov dx, [out_chan2_1]
2257            mov ax, [X_P_O_T]
2258            out dx, ax
2259          }
2260     }
2261   }
2262 // ********** ROUNDING BLOCK - x_neg_output_top **********
```

```
2263    g = ceil(x_neg_output_top);
2264    z = x_neg_output_top + 0.5;
2265
2266    if(g >= z)
2267       v = floor(x_neg_output_top);
2268    else
2269       v = g;
2270
2271       X_N_O_T = v + t48;
2272 //     ************
2273    if(X_N_O_T < out_min)
2274    {
2275      if(flag16 == 0)
2276         outport(out_chan2_2, out_min);
2277
2278      else
2279
2280      if(flag16 == 1)
2281      {
2282        asm{
2283             mov dx, [out_chan2_2]
2284             mov ax, [out_min]
2285             out dx, ax
2286           }
2287      }
2288    }// End of if(X_N_O_T < out_min)
2289
2290    else
2291
2292    if(X_N_O_T > out_max)
2293    {
2294      if(flag16 == 0)
2295         outport(out_chan2_2, out_max);
2296
2297      else
2298
2299      if(flag16 == 1)
2300      {
2301        asm{
2302             mov dx, [out_chan2_2]
2303             mov ax, [out_max]
2304             out dx, ax
2305           }
2306      }
2307    }// End of if(X_N_O_T > out_max)
2308
2309    else
2310
2311    {
2312      if(flag16 == 0)
2313         outport(out_chan2_2, X_N_O_T);
2314
2315      else
2316
2317      if(flag16 == 1)
2318      {
2319        asm{
2320             mov dx, [out_chan2_2]
```

```
2321            mov ax, [X_N_O_T]
2322            out dx, ax
2323          }
2324     }
2325   }
2326 // ********** ROUNDING BLOCK - y_pos_output_top **********
2327   g = ceil(y_pos_output_top);
2328   z = y_pos_output_top + 0.5;
2329
2330   if(g >= z)
2331      v = floor(y_pos_output_top);
2332   else
2333      v = g;
2334
2335      Y_P_O_T = v + t48;
2336 //     ************
2337   if(Y_P_O_T < out_min)
2338   {
2339     if(flag16 == 0)
2340        outport(out_chan2_3, out_min);
2341
2342     else
2343
2344     if(flag16 == 1)
2345     {
2346       asm{
2347            mov dx, [out_chan2_3]
2348            mov ax, [out_min]
2349            out dx, ax
2350          }
2351     }
2352   }// End of if(Y_P_O_T < out_min)
2353
2354   else
2355
2356   if(Y_P_O_T > out_max)
2357   {
2358     if(flag16 == 0)
2359        outport(out_chan2_3, out_max);
2360
2361     else
2362
2363     if(flag16 == 1)
2364     {
2365       asm{
2366            mov dx, [out_chan2_3]
2367            mov ax, [out_max]
2368            out dx, ax
2369          }
2370     }
2371   }// End of if(Y_P_O_T > out_max)
2372
2373   else
2374
2375   {
2376     if(flag16 == 0)
2377        outport(out_chan2_3, Y_P_O_T);
2378
```

```
2379      else
2380
2381      if(flag16 == 1)
2382      {
2383        asm{
2384             mov dx, [out_chan2_3]
2385             mov ax, [Y_P_O_T]
2386             out dx, ax
2387           }
2388      }
2389    }
2390  // ********** ROUNDING BLOCK - y_neg_output_top **********
2391    g = ceil(y_neg_output_top);
2392    z = y_neg_output_top + 0.5;
2393
2394    if(g >= z)
2395       v = floor(y_neg_output_top);
2396    else
2397       v = g;
2398
2399       Y_N_O_T = v + t48;
2400  //     ************
2401    if(Y_N_O_T < out_min)
2402    {
2403      if(flag16 == 0)
2404         outport(out_chan2_4, out_min);
2405
2406      else
2407
2408      if(flag16 == 1)
2409      {
2410        asm{
2411             mov dx, [out_chan2_4]
2412             mov ax, [out_min]
2413             out dx, ax
2414           }
2415      }
2416    }// End of if(Y_N_O_T < out_min)
2417
2418    else
2419
2420    if(Y_N_O_T > out_max)
2421    {
2422      if(flag16 == 0)
2423         outport(out_chan2_4, out_max);
2424
2425      else
2426
2427      if(flag16 == 1)
2428      {
2429        asm{
2430             mov dx, [out_chan2_4]
2431             mov ax, [out_max]
2432             out dx, ax
2433           }
2434      }
2435    }// End of if(Y_N_O_T > out_max)
2436
```

```
2437    else
2438
2439    {
2440      if(flag16 == 0)
2441         outport(out_chan2_4, Y_N_O_T);
2442
2443      else
2444
2445      if(flag16 == 1)
2446      {
2447        asm{
2448             mov dx, [out_chan2_4]
2449             mov ax, [Y_N_O_T]
2450             out dx, ax
2451           }
2452      }
2453    }
2454
2455    dotXav = Xav - oldoldXav;
2456    oldoldXav = oldXav;
2457    oldXav = Xav;
2458
2459    dotYav = Yav - oldoldYav;
2460    oldoldYav = oldYav;
2461    oldYav = Yav;
2462
2463    dotThetaX = ThetaX - oldoldThetaX;
2464    oldoldThetaX = oldThetaX;
2465    oldThetaX = ThetaX;
2466
2467    dotThetaY = ThetaY - oldoldThetaY;
2468    oldoldThetaY = oldThetaY;
2469    oldThetaY = ThetaY;
2470 }// End of if(flag10 == 1)
2471
2472 // ****************************END MODAL CONTROL*****************************
2473
2474 // ****************************** THRUST BEARING ****************************
2475
2476 if(flag3 == 1)
2477 {
2478 //       * * * Begin z_force_th calc * * *
2479
2480   zth = (zth1 + zth2) / 2.0;
2481   zthderiv = zth - z_th_old3;
2482   zthsum = zthsum + igainth * zth;
2483
2484 //       * * * Calculate z_force_th * * *
2485
2486   z_force_th = (kv_th * zth + dv_th * zthderiv) / 2.0 + zthsum
2487                                                       - tBias_th;
2488   up_output_th   =   z_force_th - bias_current_th;
2489   down_output_th =   z_force_th + bias_current_th;
2490
2491 //       * * *  OUTPUTS FOR z_direction_th  * * *
2492
2493 // **********ROUNDING BLOCK**********
2494        g = ceil(up_output_th);
```

```
2495         z = up_output_th + 0.5;
2496
2497         if(g >= z)
2498            v = floor(up_output_th);
2499         else
2500            v = g;
2501
2502         round2 = v + t48;
2503 // **********************************
2504
2505       if(round2 < out_min)
2506       {
2507         if(flag16 == 0)
2508            outport(out_chan1_5, out_min);
2509
2510         else
2511
2512         if(flag16 == 1)
2513         {
2514           asm{
2515                mov dx, [out_chan1_5]
2516                mov ax, [out_min]
2517                out dx, ax
2518              }
2519         }
2520       }
2521
2522       else
2523
2524       if(round2 > out_max)
2525       {
2526         if(flag16 == 0)
2527            outport(out_chan1_5, out_max);
2528
2529         else
2530
2531         if(flag16 == 1)
2532         {
2533           asm{
2534                mov dx, [out_chan1_5]
2535                mov ax, [out_max]
2536                out dx, ax
2537              }
2538         }
2539       }
2540
2541       else
2542
2543       {
2544         if(flag16 == 0)
2545            outport(out_chan1_5, round2);// VERT.(UP)
2546
2547         else
2548
2549         if(flag16 == 1)
2550         {
2551           asm{
2552                mov dx, [out_chan1_5]
```

```
2553                mov ax, [round2]
2554                out dx, ax
2555              }
2556         }
2557       }
2558 // ***********************************
2559         g = ceil(down_output_th);
2560         z = down_output_th + 0.5;
2561
2562         if(g >= z)
2563            v = floor(down_output_th);
2564         else
2565            v = g;
2566
2567         round2 = v + t48;
2568 // ***********************************
2569
2570       if(round2 < out_min)
2571       {
2572         if(flag16 == 0)
2573            outport(out_chan2_5, out_min);
2574
2575         else
2576
2577         if(flag16 == 1)
2578         {
2579           asm{
2580                mov dx, [out_chan2_5]
2581                mov ax, [out_min]
2582                out dx, ax
2583              }
2584         }
2585       }
2586
2587       else
2588
2589       if(round2 > out_max)
2590       {
2591         if(flag16 == 0)
2592            outport(out_chan2_5, out_max);
2593
2594         else
2595
2596         if(flag16 == 1)
2597         {
2598           asm{
2599                mov dx, [out_chan2_5]
2600                mov ax, [out_max]
2601                out dx, ax
2602              }
2603         }
2604       }
2605
2606       else
2607
2608       {
2609         if(flag16 == 0)
2610            outport(out_chan2_5, round2);// VERT.(DOWN)
```

```
2611
2612        else
2613
2614        if(flag16 == 1)
2615        {
2616          asm{
2617               mov dx, [out_chan2_5]
2618               mov ax, [round2]
2619               out dx, ax
2620             }
2621        }
2622      }
2623
2624 // z_th_old5 = z_th_old4;
2625 // z_th_old4 = z_th_old3;
2626    z_th_old3 = z_th_old2;
2627    z_th_old2 = z_th_old1;
2628    z_th_old1 = zth;
2629
2630 //         * * * End z_force_th * * *
2631
2632 }// End of if(flag3 == 1)
2633
2634 //                        * * * Safe Gain * * *
2635      if (sg3 == 1)
2636      goto T1;
2637
2638      else
2639
2640      goto T2;
2641
2642 T1:  {
2643        if ((zth * zth) > zsafe)
2644        {
2645          kv_th = 1.5;
2646          dv_th = 9.0;
2647        }
2648        goto T2;
2649      }
2650 //                        * * * End Safe Gain * * *
2651
2652 T2:
2653
2654 if(diag == 1)
2655 {
2656   if(flag4d == 1)
2657   {
2658     if(flag4c == 1)
2659     {
2660
2661 //    junk = exp(1.34567);
2662 //    junk = exp(1.34567);
2663       junk = exp(1.34567);
2664       junk = exp(1.34567);
2665       junk = cos(1.34567);
2666       junk = cos(1.34567);
2667 //    junk = cos(1.34567);
2668 //    junk = cos(1.34567);
```

```
2669 //    junk = cos(1.34567);
2670 //    junk = cos(1.34567);
2671     }// End of if(flag4c == 1)
2672   }// End of if(flag4d == 1)
2673 }// End of if(diag == 1)
2674   if(flag11 == 1)// Lower bearing write out activation flag
2675   {
2676     if(nw_bot == 1)
2677     {
2678       if(i_bot == 1)
2679       {
2680         gotoxy(51,22);textcolor(11);
2681         cprintf("%6.1fv         %6.1fv", xbot / 204.8, ybot / 204.8);
2682         gotoxy(49,23);textcolor(11);
2683         cprintf("%4.1fv,%6.1fv,%6.1fv,%6.1fv", x_pos_output_bot / 204.8,
2684                                                x_neg_output_bot / 204.8,
2685                                                y_pos_output_bot / 204.8,
2686                                                y_neg_output_bot / 204.8);
2687
2688         if(flag10 == 0)// Activates when modal is off
2689         {
2690           gotoxy(25,22);textcolor(11);
2691           cprintf("%9.2fv",x_force_bot / 204.8);
2692           gotoxy(25,23);
2693           cprintf("%9.2fv",y_force_bot / 204.8);
2694         }
2695       }// End of if(i_bot == 1)
2696         i_bot = i_bot + 1;
2697
2698         if(i_bot == 1025)
2699            i_bot = 1;
2700     }// End of if(nw_bot == 1)
2701   }// End of if(flag11 == 1)
2702
2703   else
2704
2705   if(flag22 == 1)// Upper bearing write out activation flag
2706   {
2707     if(nw_top == 1)
2708     {
2709       if(i_top == 1)
2710       {
2711         gotoxy(51,22);textcolor(11);
2712         cprintf("%6.1fv         %6.1fv", xtop / 204.8, ytop / 204.8);
2713         gotoxy(49,23);textcolor(11);
2714         cprintf("%4.1fv,%6.1fv,%6.1fv,%6.1fv", x_pos_output_top / 204.8,
2715                                                x_neg_output_top / 204.8,
2716                                                y_pos_output_top / 204.8,
2717                                                y_neg_output_top / 204.8);
2718
2719         if(flag10 == 0)// Activates when modal is off
2720         {
2721           gotoxy(25,22);textcolor(11);
2722           cprintf("%9.2fv",x_force_top / 204.8);
2723           gotoxy(25,23);
2724           cprintf("%9.2fv",y_force_top / 204.8);
2725         }
2726       }// End of if(i_top == 1)
```

```
2727          i_top = i_top + 1;
2728
2729          if(i_top == 1025)
2730             i_top = 1;
2731       }// End of if(nw_top == 1)
2732    }// End of if(flag22 == 1)
2733
2734    else
2735
2736    if(flag33 == 1)// Thrust bearing write out activation flag
2737    {
2738       if(nw_th == 1)
2739       {
2740          if(i_th == 1)
2741          {
2742             gotoxy(51,22);textcolor(11);
2743             cprintf("%6.1fv     ", zth / 204.8);
2744             gotoxy(49,23);textcolor(11);
2745             cprintf("%4.1fv,%6.1fv", up_output_th / 204.8,
2746                                      down_output_th / 204.8);
2747
2748             if(flag10 == 0)// Activates when modal is off
2749             {
2750                gotoxy(25,22);textcolor(11);
2751                cprintf("%9.2fv",z_force_th / 204.8);
2752             }
2753          }// End of if(i_th == 1)
2754             i_th = i_th + 1;
2755
2756             if(i_th == 1025)
2757                i_th = 1;
2758       }// End of if(nw_th == 1)
2759    }// End of if(flag33 == 1)
2760
2761 n++;
2762
2763 }// End of while (n <= nmax) loop
2764
2765 // ************** Time & Loop time update block **************
2766
2767       gettime(&tt);
2768
2769       if(tt.ti_hour == 0)
2770       {
2771          hh = -12;
2772          gotoxy(48,10);textcolor(14);
2773          cprintf("AM");
2774       }
2775
2776       else
2777
2778       if(tt.ti_hour >= 1 && tt.ti_hour < 12)
2779       {
2780          hh = 0;
2781          gotoxy(48,10);textcolor(14);
2782          cprintf("AM");
2783       }
2784
```

```
2785      else
2786
2787      if(tt.ti_hour == 12)
2788      {
2789        hh = 0;
2790        gotoxy(48,10);textcolor(14);
2791        cprintf("PM");
2792      }
2793
2794      else
2795
2796      if(tt.ti_hour > 12 && tt.ti_hour < 24)
2797      {
2798        hh = 12;
2799        gotoxy(48,10);textcolor(14);
2800        cprintf("PM");
2801      }
2802        gotoxy(33,10);textcolor(14);
2803        cprintf("Time:");
2804
2805        gotoxy(39,10);textcolor(11);
2806        cprintf("%2d:%02d:%02d\n",
2807        tt.ti_hour-hh, tt.ti_min, tt.ti_sec);
2808
2809      if(flag_L == 1)
2810      {
2811        gotoxy(1,13);textcolor(14+128);
2812        cprintf("   QUIT(y/n)?: ");
2813      }
2814
2815      if(l == lmax)// Time update block
2816      {
2817        gettime(&now);
2818        last_time = time1;
2819        time1 = now.ti_sec + 0.01 * now.ti_hund + 60.0 * now.ti_min;
2820        loop_time =((time1 - last_time) * micro);
2821
2822        if(abs(loop_time) < 800.0)
2823        {
2824          if(flag10 == 1 && diag == 1)
2825          {
2826            gotoxy(34,13);textcolor(15);
2827            cprintf("%6.2f", k_tilt);
2828            gotoxy(34,14);textcolor(15);
2829            cprintf("%6.2f",c_tilt);
2830          }
2831          if(nw_bot == 1 || nw_top == 1 || nw_th == 1)
2832          {
2833            gotoxy(62,22);textcolor(128+14);
2834            cprintf("w");
2835          }
2836
2837          if(nw_bot == 0 && nw_top == 0 &&
2838             nw_th  == 0 && flag_B == 1 &&
2839             flag10 == 0 || flag10 == 1)
2840          {
2841            gotoxy(27,23);textcolor(14);
2842            cprintf("[<^> to toggle D.A. ]");
```

```
2843          }
2844          gotoxy(39,9);textcolor(15);
2845          cprintf("%6.2f",loop_time);
2846
2847          if(flag24 == 1 && flag_K == 1)// Dynamic Averaging block
2848          {
2849            ii = ii + 1.0;
2850
2851            A1  =  A2; A2  =  A3; A3  =  A4; A4  = A5;
2852            A5  =  A6; A6  =  A7; A7  =  A8; A8  = A9;
2853            A9  = A10; A10 = A11; A11 = A12; A12 = A13;
2854            A13 = A14; A14 = A15; A15 = loop_time;
2855
2856            L_T = A1+A2+A3+A4+A5+A6+A7+A8+A9+A10+A11+A12+A13+A14+A15;
2857            LT = L_T / 15.0;// Average loop time
2858
2859            PL = 1000000.0 / (freq*LT*500);// Period length
2860            O =  1/PL;// O = (1/period length), used in signal generation
2861                      // block
2862
2863            qq = qq + 1;
2864            if(qq > vv)
2865            {
2866              qq = 0;
2867              ii = 0.0;
2868            }
2869          }// End of if(flag24 == 1 && flag_K == 1)
2870
2871          else
2872
2873          if(flag24 == 0 && flag_K == 1)// Intermittent Averaging block
2874          {
2875            if(rr == 0 && ii <= 15.0)
2876            {
2877              ii = ii + 1.0;// Counter
2878              OO = 1000000.0 / (freq*loop_time*500);// Period length
2879              OL = OL + OO;// Accumulated period length
2880              L_T = L_T + loop_time;
2881              if(ii == 15.0)
2882              {
2883                PL = OL / ii;// Average period length
2884                LT = L_T / ii;// Average loop time
2885                O  = 1.0 / PL;
2886                rr = 1;
2887                OL = 0.0;
2888                L_T = 0.0;
2889              }
2890            }// End of if(rr == 0 && ii <= 15.0)
2891            qq = qq + 1;
2892            if(qq > vv)
2893            {
2894              rr = 0;
2895              qq = 0;
2896              ii = 0.0;
2897            }
2898          }// End of if(flag24 == 0 && flag_K == 1)
2899
2900          if(flag_K == 1)
```

```
2901          {
2902            if(flag_H == 1)
2903            {
2904              gotoxy(1,21);textcolor(15);
2905              cprintf("PL: %6.4f,%4.1f,%3u ",PL,ii,vv);
2906            }
2907            else
2908            if(flag_H == 0)
2909            {
2910              gotoxy(1,21);textcolor(15);
2911              cprintf("PL: %6.4f",PL);
2912            }
2913          }// End of if(flag_K == 1)
2914
2915          if(resp == 'o' || resp == 'O')
2916          {
2917            frequency = (1000000.0/(PL*loop_time*500));
2918            gotoxy(1,21);textcolor(15);
2919            cprintf("PL: %6.4f               ",PL);
2920            gotoxy(1,23);textcolor(15);
2921            cprintf("<o>freq:%8.2f Hz.",frequency);
2922          }
2923          else
2924          {
2925            gotoxy(1,23);textcolor(15);
2926            cprintf("< >1/PL:            ");
2927            gotoxy(10,23);textcolor(15);
2928            cprintf("%7.3f", O);
2929
2930            if(ii < COUNTMAX)
2931            {
2932              gotoxy(2,23);textcolor(12+128);
2933              cprintf("o");
2934            }
2935            else
2936            {
2937              COUNTMAX = -1.0;
2938              gotoxy(2,23);textcolor(10);
2939              cprintf("o");
2940            }
2941          }
2942        // ************************************
2943        if(diag == 0)
2944        {
2945          flag_HH = flag_HH + 1;
2946
2947          if(flag_HH == 1)
2948          {
2949            TC = 10;
2950            gotoxy(37,19);textcolor(TC);
2951            cprintf("  NASA  ");
2952          }
2953          else
2954          if(flag_HH == 2)
2955          {
2956            TC = 11;
2957            gotoxy(37,19);textcolor(TC);
2958            cprintf(" GLENN  ");
```

```
2959          }
2960          else
2961          if(flag_HH == 3)
2962          {
2963            TC = 13;
2964            gotoxy(37,19);textcolor(TC);
2965            cprintf("RESEARCH");
2966          }
2967          else
2968          if(flag_HH == 4)
2969          {
2970            TC = 14;
2971            gotoxy(37,19);textcolor(TC);
2972            cprintf(" CENTER ");
2973          }
2974          if(flag_HH >= 4)
2975             flag_HH = 0;
2976        }// End of if(diag == 0)
2977        // ************************************
2978          if(flag_BB == 1)
2979          {
2980            gotoxy(1,2);textcolor(15);
2981            cprintf("<q> to abort control     ");
2982
2983            if(flag_B == 1 && flag44 == 0 || diag == 0)
2984            {
2985              if(flag10 == 0 && nw_bot == 0 &&
2986                 nw_top == 0 && nw_th  == 0 || flag10 == 1)
2987                {
2988                  gotoxy(42,23);textcolor(12);
2989                  cprintf("I.A.");
2990                }
2991            }
2992            if(diag == 0)
2993            {
2994              gotoxy(1,5);textcolor(15);
2995              cprintf("<4-0> to select excitation ");
2996
2997              gotoxy(1,3);textcolor(15);
2998              cprintf("<m> to toggle modal cntrl    ");
2999
3000              gotoxy(1,4);textcolor(15);
3001              cprintf("<?> to toggle f_excite         ");
3002            }
3003            if(diag == 1)
3004            {
3005              gotoxy(1,1);textcolor(15);
3006              cprintf("<+,-> to toggle input-output writes");
3007
3008              gotoxy(1,3);textcolor(15);
3009              cprintf("<f> to toggle loop time buffer");
3010
3011              gotoxy(1,4);textcolor(15);
3012              cprintf("<e> non diagnostic             ");
3013
3014              gotoxy(1,5);textcolor(15);
3015              cprintf("<!,@,#> disable safe gain   ");
3016            }
```

```
3017            if(switch1 == 1)
3018            {
3019              gotoxy(1,25);textcolor(13);
3020              cprintf("[         ]       ");
3021              gotoxy(2,25);textcolor(14);
3022              cprintf("f_excite2");
3023              gotoxy(13,25);textcolor(15+128);
3024              cprintf("<==");
3025            }
3026            else
3027            if(switch1 == 0)
3028            {
3029              gotoxy(1,25);textcolor(15);
3030              cprintf("<?>f_excite :           ");
3031            }
3032            gotoxy(2,20);textcolor(10);
3033            cprintf("k");
3034
3035            if(flag_N == 1)
3036            {
3037              if(flag_jj == 1)
3038              {
3039                gotoxy(1,14);textcolor(10);
3040                cprintf("< >PHSE ANG:%3u deg ",th);
3041                gotoxy(2,14);textcolor(15);
3042                cprintf("n");
3043                flag_jj = 0;
3044              }
3045              else
3046              if(flag_jj == 0)
3047              {
3048                gotoxy(1,14);textcolor(13);
3049                cprintf("<  >phi ANG:%3u deg",thp);
3050                gotoxy(2,14);textcolor(15);
3051                cprintf("{}");
3052                flag_jj = 1;
3053              }
3054            }
3055            flag_BB = 0;
3056          }// End of if(flag_BB == 1)
3057          else
3058          if(flag_BB == 0)
3059          {
3060            if(diag == 1)
3061            {
3062              gotoxy(1,1);textcolor(9);
3063              cprintf("<4-0> to select excitation         ");
3064            }
3065
3066            if(flag_B == 1 && flag44 == 0 || diag == 0 )
3067            {
3068              if(flag10 == 0 && nw_bot == 0 &&
3069                 nw_top == 0 && nw_th  == 0 || flag10 == 1)
3070              {
3071                gotoxy(42,23);textcolor(10);
3072                cprintf("D.A.");
3073              }
3074            }
```

```
3075            gotoxy(1,2);textcolor(10);
3076            cprintf("<R> to toggle Bounce/Tilt");
3077
3078            if(diag == 1 || diag == 0)
3079            {
3080              if(flagMM == 0 && switch1 == 0)
3081              {
3082                gotoxy(1,25);textcolor(15);
3083                cprintf("<s>to adjust Pulse Width");
3084              }
3085              gotoxy(1,3);textcolor(13);
3086              cprintf("<F> to toggle O.P.R. dirction ");
3087
3088              gotoxy(1,4);textcolor(14);
3089              cprintf("<<> to toggle ext.input.exction");
3090
3091              gotoxy(1,5);textcolor(11);
3092              cprintf("<&,*>avrg freq update adjst");
3093            }// End of if(diag == 1 || diag == 0)
3094            gotoxy(2,20);textcolor(12);
3095            cprintf("x");
3096            flag_BB = 1;
3097
3098            if(flag_N == 1)
3099            {
3100              gotoxy(1,14);textcolor(15);
3101              cprintf("[                  ]");
3102              gotoxy(2,14);textcolor(14);
3103              cprintf("< >ONE_PR_REV");
3104              gotoxy(3,14);textcolor(10);
3105              cprintf("r");
3106              gotoxy(16,14);textcolor(12+128);
3107              cprintf("OFF");
3108            }
3109          }// End of if(flag_BB == 0)
3110        // ************************************
3111        }// End of if(abs(loop_time) < 800.0)
3112          l = 0;
3113      }// End of if(l == lmax)
3114          l++;
3115
3116          hh = kbhit();
3117
3118          if(hh == 0)
3119          goto loop;
3120
3121          else
3122
3123          {
3124            resp = getch();
3125            hh = 0;
3126          }
3127
3128 if(diag == 1 && SSS == 1)
3129 {
3130     if(resp == 'q' || resp == 'Q')
3131     {
3132        flag_L = 1;
```

```
3133          goto loop;
3134       }
3135       if(flag_L == 1)
3136       {
3137         if(resp == 'y' || resp == 'Y')
3138         goto ramp_down;
3139
3140         if(resp == 'n' || resp == 'N')
3141         {
3142           gotoxy(1,13);textcolor(14);
3143           cprintf("                    ");// ERASE QUIT(y/n)?:
3144           flag_L = 0;
3145           goto loop;
3146         }
3147       }
3148
3149       if(resp == 'p') goto kv_up;          if(resp == 'P') goto kv_down;
3150       if(resp == 'd') goto dh_up;          if(resp == 'D') goto dh_down;
3151       if(resp == 'g') goto kh_up;          if(resp == 'G') goto kh_down;
3152       if(resp == 'v') goto dv_up;          if(resp == 'V') goto dv_down;
3153       if(resp == 'w') goto wBias_up;       if(resp == 'W') goto wBias_down;
3154       if(resp == 't') goto tBias_up;       if(resp == 'T') goto tBias_down;
3155       if(resp == 'b') goto bias_up;        if(resp == 'B') goto bias_down;
3156       if(resp == 'f') goto buffer;         if(resp == 'M') goto test_signal;
3157       if(resp == '+') goto writeout;       if(resp == '-') goto nowrite;
3158       if(resp == ')') goto igain_up;       if(resp == '(') goto igain_down;
3159       if(resp == '!') goto disable_safe1;if(resp == '1') goto enable_safe1;
3160       if(resp == '@') goto disable_safe2;if(resp == '2') goto enable_safe2;
3161       if(resp == '#') goto disable_safe3;if(resp == '3') goto enable_safe3;
3162       if(resp == 'e') goto non_diagnostic;
3163
3164       if(resp == 'H') goto thrust_bearing;
3165       if(resp == 'I') goto upper_bearing;
3166       if(resp == 'J') goto lower_bearing;
3167
3168       if(resp == 'l') goto l_on;
3169       if(resp == 'u') goto u_on;
3170       if(resp == 'z') goto z_on;
3171 }// End of if(diag == 1 && SSS == 1)
3172
3173 if(diag == 1 || diag == 0)
3174 {
3175       if(resp == 'q' || resp == 'Q')
3176       {
3177          flag_L = 1;
3178          goto loop;
3179       }
3180       if(flag_L == 1)
3181       {
3182         if(resp == 'y' || resp == 'Y')
3183         goto ramp_down;
3184
3185         if(resp == 'n' || resp == 'N')
3186         {
3187           gotoxy(1,13);textcolor(14);
3188           cprintf("                    ");// ERASE QUIT(y/n)?:
3189           flag_L = 0;
3190           goto loop;
```

```
3191        }
3192      }
3193
3194      if(resp == 'c') goto cg_factor_up;
3195      if(resp == 'C') goto cg_factor_down;
3196      if(resp == 'E') goto diagnostic;
3197      if(resp == 'm') goto modal;
3198      if(resp == 'o') goto frequency_up;
3199      if(resp == 'O') goto frequency_down;
3200      if(resp == 'a') goto amplitude_up;
3201      if(resp == 'A') goto amplitude_down;
3202      if(resp == ':') goto assembly;
3203      if(resp == '?') goto display;
3204      if(resp == ',') goto excitation;
3205      if(resp == '<') goto excitation_switch;
3206      if(resp == '*') goto vv_up;
3207      if(resp == '&') goto vv_down;
3208      if(resp == '$') goto excite1_toggle;
3209      if(resp == '^') goto loop_time_average_toggle;
3210      if(resp == '{') goto phi_down;
3211      if(resp == '}') goto phi_up;
3212
3213      if(resp == '4') goto excite1;
3214      if(resp == '5') goto excite2;
3215      if(resp == '6') goto excite3;
3216      if(resp == '7') goto excite4;
3217      if(resp == '8') goto excite5;
3218      if(resp == '9') goto excite6;
3219      if(resp == '0') goto excite7;
3220
3221      if(resp == 's') goto pulse_width_up;
3222      if(resp == 'S') goto pulse_width_down;
3223      if(resp == 'x') goto frequency_input_up;
3224      if(resp == 'X') goto frequency_input_down;
3225      if(resp == 'k') goto freq_fine_adjust_up;
3226      if(resp == 'K') goto freq_fine_adjust_down;
3227      if(resp == 'n') goto THETA_up;
3228      if(resp == 'N') goto THETA_down;
3229      if(resp == 'r') goto one_per_rev;
3230      if(resp == 'R') goto Tilt_Bounce_Mode;
3231      if(resp == 'F') goto one_p_rev_dir;
3232      goto loop;
3233  }// End of if(diag == 1 || diag == 0)
3234
3235  loop_time_average_toggle:{
3236                              if(flag_K == 1)
3237                              {
3238                                if(flag25 == 1)
3239                                {
3240                                  flag24 = 0;
3241                                  flag25 = 0;
3242                                  flag_H = 1;
3243                                  gotoxy(21,21);textcolor(12);
3244                                  cprintf("I.A.");
3245                                  rr = 0;
3246                                  OL = 0.0;
3247                                  L_T = 0.0;
3248                                  qq = 0;
```

```
3249                                ii = 0.0;
3250                              }
3251                              else
3252                              if(flag25 == 0)
3253                              {
3254                                flag24 = 1;
3255                                flag25 = 1;
3256                                flag_H = 1;
3257                                gotoxy(1,21);textcolor(15);
3258                                cprintf("PL: %6.4f,%4.1f,%3u",PL,ii,vv=15);
3259                                gotoxy(21,21);textcolor(10);
3260                                cprintf("D.A.");
3261                              }
3262                              goto loop;
3263                            }// End of if(flag_K = 1)
3264                           goto loop;
3265                         }
3266 vv_up:{
3267         if(flag24 == 0 && flag_K == 1)
3268         {
3269           OL = 0.0;
3270           L_T = 0.0;
3271           rr = 0;
3272           qq = 0;
3273           ii = 0.0;
3274           vv = vv + 1;
3275           if(vv >= 100)
3276              vv = 100;
3277           gotoxy(17,21);textcolor(15);
3278           cprintf("%3u",vv);
3279           goto loop;
3280         }
3281         goto loop;
3282       }
3283 vv_down:{
3284           if(flag24 == 0 && flag_K == 1)
3285           {
3286             OL = 0.0;
3287             L_T = 0.0;
3288             rr = 0;
3289             qq = 0;
3290             ii = 0.0;
3291             vv = vv - 1;
3292             if(vv <= 15)
3293                vv = 15;
3294             gotoxy(17,21);textcolor(15);
3295             cprintf("%3u",vv);
3296             goto loop;
3297           }
3298           goto loop;
3299         }
3300 excitation_switch:{
3301                      if(flagNN == 1)
3302                      {
3303                        switch1 = 1;
3304                        flagNN = 0;
3305                        gotoxy(1,25);textcolor(13);
3306                        cprintf("[         ]       ");
```

```
3307                        gotoxy(2,25);textcolor(14);
3308                        cprintf("f_excite2");
3309                        gotoxy(13,25);textcolor(15);
3310                        cprintf("<==          ");
3311                      }
3312                      else
3313                      if(flagNN == 0)
3314                      {
3315                        COUNTMAX = 15.0;
3316                        OL = 0.0;
3317                        L_T = 0.0;
3318                        rr = 0;
3319                        qq = 0;
3320                        ii = 0.0;
3321                        switch1 = 0;
3322                        flagNN = 1;
3323                        gotoxy(1,25);textcolor(15);
3324                        cprintf("<?>f_excite :%5d",f_excite);
3325                      }
3326                      goto loop;
3327                    }
3328 test_signal:{
3329                 if(flagLL == 1)
3330                 {
3331                   test_signal = 1;
3332                   flagLL = 0;
3333                   gotoxy(36,11);textcolor(13);
3334                   cprintf("<M>-test: %1u",test_signal);
3335                   gotoxy(46,11);textcolor(12);
3336                   cprintf("%1u",test_signal);
3337                 }
3338                 else
3339                 if(flagLL == 0)
3340                 {
3341                  test_signal = 0;
3342                  flagLL = 1;
3343                  gotoxy(37,11);textcolor(15);
3344                  cprintf("M");
3345                  gotoxy(46,11);textcolor(10);
3346                  cprintf("%1u",test_signal);
3347                 }
3348                 goto loop;
3349               }
3350 one_p_rev_dir:{
3351                   if(flag_N == 0)
3352                   {
3353                     gotoxy(21,6);textcolor(13);
3354                     cprintf("O.P.R.");
3355                     gotoxy(28,6);textcolor(13+128);
3356                     cprintf("----->");
3357
3358                     if(flagKK == 1)
3359                     {
3360                       II = -1.0;
3361                       gotoxy(36,6);textcolor(11);
3362                       cprintf("Anti clkwse");
3363                       flagKK = 0;
3364                     }
```

```
3365                    else
3366                    if(flagKK == 0)
3367                    {
3368                      II = 1.0;
3369                      gotoxy(36,6);textcolor(11);
3370                      cprintf("  Clckwse  ");
3371                      flagKK = 1;
3372                    }
3373                    goto loop;
3374                  }
3375                  goto loop;
3376                }
3377 Tilt_Bounce_Mode:{
3378                     if(flagJJ == 1)
3379                     {
3380                       JJ = 1.0;
3381                       flagJJ = 0;
3382                       gotoxy(32,7);textcolor(15);
3383                       cprintf("==>             <==");
3384                       gotoxy(36,7);textcolor(14+128);
3385                       cprintf("BOUNCE MODE");
3386                     }
3387                     else
3388                     if(flagJJ == 0)
3389                     {
3390                       JJ = -1.0;
3391                       flagJJ = 1;
3392                       gotoxy(32,7);textcolor(15);
3393                       cprintf("==>             <==");
3394                       gotoxy(36,7);textcolor(13+128);
3395                       cprintf(" TILT MODE ");
3396                     }
3397                     goto loop;
3398                   }
3399 one_per_rev:{
3400                 if(flag_M == 1)// Toggle on flag
3401                 {
3402                   ns = 1.0;// Condition for correct manual vector rotation
3403                   gotoxy(1,14);textcolor(15);
3404                   cprintf("[                ]");
3405                   gotoxy(2,14);textcolor(14);
3406                   cprintf("< >ONE_PR_REV");
3407                   gotoxy(3,14);textcolor(10);
3408                   cprintf("r");
3409                   gotoxy(16,14);textcolor(10);
3410                   cprintf("ON");
3411                   flag_II = 1;// one_per_rev set to on
3412                   flag_M  = 0;
3413                   flag_N  = 0;
3414                   goto loop;
3415                 }
3416                 else
3417                 if(flag_M == 0)// Toggle off flag
3418                 {
3419                   gotoxy(16,14);textcolor(12+128);
3420                   cprintf("OFF");
3421                   flag_II = 0;
3422                   flag_M  = 1;
```

```
3423                  flag_N  = 1;
3424                  THETA = 0.0;
3425                  th = 0;
3426                  goto loop;
3427                }
3428              }
3429 THETA_up:{
3430             if(flag_II == 0)// One - Per - Rev is Off
3431             {
3432               ns = -1.0;// Condition for correct manual vector rotation
3433               THETA = THETA + 5.0 * M_PI/180.0;
3434               th = th + 5;
3435               if(THETA >= 2.0 * M_PI)
3436               {
3437                 THETA = 2.0 * M_PI;
3438                 th = 360;
3439               }
3440               gotoxy(1,14);textcolor(10);
3441               cprintf("< >PHSE ANG:    deg");
3442               gotoxy(2,14);textcolor(15);
3443               cprintf("n");
3444               gotoxy(13,14);textcolor(15);
3445               cprintf("%3u",th);
3446               goto loop;
3447             }// End of if(flag_II == 0)
3448             goto loop;
3449           }
3450 THETA_down:{
3451               if(flag_II == 0)
3452               {
3453                 ns = -1.0;// Condition for correct manual vector rotation
3454                 THETA = THETA - 5.0 * M_PI/180.0;
3455                 th = th - 5;
3456                 if(THETA <= 0.0 && th <= 0)
3457                 {
3458                   THETA = 0.0;
3459                   th = 0;
3460                 }
3461                 gotoxy(1,14);textcolor(10);
3462                 cprintf("< >PHSE ANG:    deg");
3463                 gotoxy(2,14);textcolor(15);
3464                 cprintf("n");
3465                 gotoxy(13,14);textcolor(15);
3466                 cprintf("%3u",th);
3467                 goto loop;
3468               }// End of if(flag_II == 0)
3469               goto loop;
3470             }
3471 phi_up:{
3472          if(flag_II == 0)
3473          {
3474            phi = phi + 5.0 * M_PI/180.0;
3475            thp = thp + 5;
3476            if(phi >= 2.0 * M_PI)
3477            {
3478              phi = 2.0 * M_PI;
3479              thp = 360;
3480            }
```

```
3481            gotoxy(1,14);textcolor(13);
3482            cprintf("<  >phi ANG:    deg");
3483            gotoxy(2,14);textcolor(15);
3484            cprintf("{}");
3485            gotoxy(13,14);textcolor(15);
3486            cprintf("%3u",thp);
3487            goto loop;
3488          }// End of if(flag_II == 0)
3489          goto loop;
3490        }
3491 phi_down:{
3492            if(flag_II == 0)
3493            {
3494              phi = phi - 5.0 * M_PI/180.0;
3495              thp = thp - 5;
3496              if(phi <= 0.0 && thp <= 0)
3497              {
3498                phi = 0.0;
3499                thp = 0;
3500              }
3501              gotoxy(1,14);textcolor(13);
3502              cprintf("<  >phi ANG:    deg");
3503              gotoxy(2,14);textcolor(15);
3504              cprintf("{}");
3505              gotoxy(13,14);textcolor(15);
3506              cprintf("%3u",thp);
3507              goto loop;
3508            }// End of if(flag_JJ == 0)
3509            goto loop;
3510          }
3511 assembly:{
3512            if(flag_A == 0)
3513            {
3514              flag16 = 1;
3515              gotoxy(42,25);textcolor(10);
3516              cprintf("ON ");
3517              flag_A = 1;
3518              goto loop;
3519            }
3520            else
3521            if(flag_A == 1)
3522            {
3523              flag16 = 0;
3524              gotoxy(42,25);textcolor(12+128);
3525              cprintf("OFF");
3526              flag_A = 0;
3527              goto loop;
3528            }
3529          }
3530 display:{
3531           if(nw_bot == 0 && nw_top == 0 && nw_th == 0)
3532           {
3533             if(flag_B == 1)
3534             {
3535               flag18 = 1;
3536               flagMM = 1;
3537               gotoxy(26,20);textcolor(15);
3538               cprintf("         ");// Erase "Force(N)"
```

```
3539                if(diag == 1)
3540                {
3541                  gotoxy(27,23);// Erase [<^> to toggle D.A. ]
3542                  cprintf("                      ");
3543                }
3544                gotoxy(1,25);textcolor(15);
3545                cprintf("<?>f_excite :%5d",f_excite);
3546                flag_B = 0;
3547                goto loop;
3548              }
3549              else
3550              if(flag_B == 0)
3551              {
3552                flag18 = 0;
3553                flagMM = 0;
3554                if(diag == 1)
3555                {
3556                 gotoxy(26,20);textcolor(15);
3557                 cprintf("Force (N)");
3558                }
3559                gotoxy(25,22);
3560                printf("         ");// Erase period length x: values
3561                gotoxy(27,23);textcolor(14);
3562                cprintf("[<^> to toggle D.A. ]");
3563                gotoxy(1,25);textcolor(15);
3564                cprintf("<s>to adjust Pulse Width");
3565                flag_B = 1;
3566                goto loop;
3567              }
3568            }
3569            goto loop;
3570          }
3571 excitation:{
3572               if(flag_C == 1)
3573               {
3574                 flag21 = 1;
3575                 gotoxy(32,24);textcolor(10);
3576                 cprintf("Enable");
3577                 flag_C = 0;
3578                 goto loop;
3579               }
3580               else
3581               if(flag_C == 0)
3582               {
3583                 flag21 = 0;
3584                 gotoxy(32,24);textcolor(12);
3585                 cprintf("Dsable");
3586                 flag_C = 1;
3587                 goto loop;
3588               }
3589             }
3590 amplitude_up:{
3591                 t04  = t04  + 102.4*0.2;
3592                 volt = volt + 0.1;
3593                 if(t04 > 1024)
3594                 {
3595                   t04  = 1024;
3596                   volt = 5.0;
```

```
3597                }
3598                gotoxy(14,24);textcolor(15);
3599                cprintf("%4.1f",volt);
3600                goto loop;
3601              }
3602 amplitude_down:{
3603                  t04  = t04  - 102.4*0.2;
3604                  volt = volt - 0.1;
3605                  if(t04 <= 0.0)
3606                  {
3607                    t04  = 0.0;
3608                    volt = 0.0;
3609                  }
3610                  gotoxy(14,24);textcolor(15);
3611                  cprintf("%4.1f",volt);
3612                  goto loop;
3613                }
3614 frequency_input_up:{
3615                       COUNTMAX = 15.0;
3616                       flag_K = 1;
3617                       flag24 = 1;
3618                       vv = 15;// used only for default display of
3619                               // D.A. mode upper limit
3620                       if(freq == 1)
3621                          freq = 0;
3622                       freq = freq + 10.0;
3623                       if(freq > 5000.0)
3624                       freq = 5000.0;
3625
3626                       gotoxy(2,20);textcolor(12);
3627                       cprintf("x");
3628
3629                       gotoxy(13,20);textcolor(15);
3630                       cprintf("%7.1f Hz.",freq);
3631
3632                       if(flag_H == 1)
3633                       {
3634                         gotoxy(21,21);textcolor(10);
3635                         cprintf("D.A.");
3636                       }
3637
3638                       rr = 0;
3639                       OL = 0.0;
3640                       L_T = 0.0;
3641                       qq = 0;
3642                       ii = 0.0;
3643
3644                       goto loop;
3645                     }
3646 freq_fine_adjust_up:{
3647                        COUNTMAX = 15.0;
3648                        flag_K = 1;
3649                        flag24 = 1;
3650                        vv = 15;// used only for default display of
3651                                // D.A. mode upper limit
3652                        freq = freq + 0.1;
3653                        if(freq > 5000.0)
3654                           freq = 5000.0;
```

```
3655
3656                        gotoxy(2,20);textcolor(10);
3657                        cprintf("k");
3658
3659                        gotoxy(13,20);textcolor(15);
3660                        cprintf("%7.1f Hz.",freq);
3661
3662                        if(flag_H == 1)
3663                        {
3664                          gotoxy(21,21);textcolor(10);
3665                          cprintf("D.A.");
3666                        }
3667
3668                        rr = 0;
3669                        OL = 0.0;
3670                        L_T = 0.0;
3671                        qq = 0;
3672                        ii = 0.0;
3673
3674                        goto loop;
3675                      }
3676 frequency_input_down:{
3677                         COUNTMAX = 15.0;
3678                         flag_K = 1;
3679                         flag24 = 1;
3680                         vv = 15;// used only for default display of
3681                                 // D.A. mode upper limit
3682                         freq = freq - 10.0;
3683                         if(freq <= 0)
3684                            freq = 10.0;
3685
3686                         gotoxy(2,20);textcolor(12);
3687                         cprintf("x");
3688
3689                         gotoxy(13,20);textcolor(15);
3690                         cprintf("%7.1f Hz.",freq);
3691
3692                         if(flag_H == 1)
3693                         {
3694                           gotoxy(21,21);textcolor(10);
3695                           cprintf("D.A.");
3696                         }
3697
3698                         rr = 0;
3699                         OL = 0.0;
3700                         L_T = 0.0;
3701                         qq = 0;
3702                         ii = 0.0;
3703
3704                         goto loop;
3705                       }
3706 freq_fine_adjust_down:{
3707                          COUNTMAX = 15.0;
3708                          flag_K = 1;
3709                          flag24 = 1;
3710                          vv = 15;// Used only for default display of
3711                                  // D.A. mode upper limit
3712                          freq = freq - 0.1;
```

```
3713                        if(freq < 0.0)
3714                           freq = 10.0;
3715
3716                        gotoxy(2,20);textcolor(10);
3717                        cprintf("k");
3718
3719                        gotoxy(13,20);textcolor(15);
3720                        cprintf("%7.1f Hz.",freq);
3721
3722                        if(flag_H == 1)
3723                        {
3724                          gotoxy(21,21);textcolor(10);
3725                          cprintf("D.A.");
3726                        }
3727
3728                        rr = 0;
3729                        OL = 0.0;
3730                        L_T = 0.0;
3731                        qq = 0;
3732                        ii = 0.0;
3733
3734                        goto loop;
3735                      }
3736 frequency_up:{
3737                 flag_K = 0;
3738                 PL = PL - 0.002;
3739                 if(PL <= 0.0)
3740                    PL = 0.002;
3741                 O = 1.0/PL;
3742                 gotoxy(1,21);textcolor(15);
3743                 cprintf("PL: %6.4f              ",PL);
3744                 goto loop;
3745               }
3746 frequency_down:{
3747                   flag_K = 0;
3748                   PL = PL + 0.002;
3749                   if(PL > 1.0)
3750                      PL = 1.0;
3751                   O = 1.0/PL;
3752                   gotoxy(1,21);textcolor(15);
3753                   cprintf("PL: %6.4f              ",PL);
3754                   goto loop;
3755                 }
3756 pulse_width_up:{
3757                   if(flag9 == 1)
3758                   {
3759                     flag_H = 0;
3760                     PWW = PWW + 1.0;
3761                     PW = 1.0/(2.0*PWW);
3762                     gotoxy(13,21);textcolor(15);
3763                     cprintf("PW: %6.4f  ",PW);
3764                     goto loop;
3765                   }
3766                   goto loop;
3767                 }
3768 pulse_width_down:{
3769                   if(flag9 == 1)
3770                   {
```

```
3771                        flag_H = 0;
3772                        PWW = PWW - 1.0;
3773                        if(PWW <= 0.0)
3774                        PWW = 1.0;
3775                        PW = 1.0/(2.0*PWW);
3776                        gotoxy(13,21);textcolor(15);
3777                        cprintf("PW: %6.4f  ",PW);
3778                        goto loop;
3779                      }
3780                      goto loop;
3781                    }
3782 excite1:{
3783           flag6 = 0;
3784           flag7 = 0;
3785           flag8 = 0;
3786           flag9 = 0;
3787           flag12 = 0;
3788           flag13 = 0;
3789           COUNTMAX = 15.0;
3790           flag_H = 1;
3791
3792           flag_AA = flag_AA + 1;
3793
3794           if(flag_AA > 5)
3795           {
3796             flag_AA = 1;
3797           }
3798
3799           if(flag_AA == 1)
3800           {
3801             gotoxy(2,19);textcolor(14);
3802             cprintf("SINE             ");
3803
3804             if(flag5 == 1)
3805             {
3806               gotoxy(16,19);textcolor(10);
3807               cprintf("ON ");
3808             }
3809             else
3810             if(flag5 == 0)
3811             {
3812               gotoxy(16,19);textcolor(12+128);
3813               cprintf("OFF");
3814             }
3815           }
3816           else
3817           if(flag_AA == 2)
3818           {
3819             gotoxy(2,19);textcolor(14);
3820             cprintf("SINE SQUARED     ");
3821
3822             if(flag5 == 1)
3823             {
3824               gotoxy(16,19);textcolor(10);
3825               cprintf("ON ");
3826             }
3827             else
3828             if(flag5 == 0)
```

```
3829              {
3830                gotoxy(16,19);textcolor(12+128);
3831                cprintf("OFF");
3832              }
3833            }
3834            else
3835            if(flag_AA == 3)
3836            {
3837              gotoxy(2,19);textcolor(14);
3838              cprintf("COSINE           ");
3839
3840              if(flag5 == 1)
3841              {
3842                gotoxy(16,19);textcolor(10);
3843                cprintf("ON ");
3844              }
3845              else
3846              if(flag5 == 0)
3847              {
3848                gotoxy(16,19);textcolor(12+128);
3849                cprintf("OFF");
3850              }
3851            }
3852            else
3853            if(flag_AA == 4)
3854            {
3855              gotoxy(2,19);textcolor(14);
3856              cprintf("COSINE SQARED    ");
3857
3858              if(flag5 == 1)
3859              {
3860                gotoxy(16,19);textcolor(10);
3861                cprintf("ON ");
3862              }
3863              else
3864              if(flag5 == 0)
3865              {
3866                gotoxy(16,19);textcolor(12+128);
3867                cprintf("OFF");
3868              }
3869            }
3870            else
3871            if(flag_AA == 5)
3872            {
3873              gotoxy(2,19);textcolor(14);
3874              cprintf("RANDOM           ");
3875
3876              if(flag5 == 1)
3877              {
3878                gotoxy(16,19);textcolor(10);
3879                cprintf("ON ");
3880              }
3881              else
3882              if(flag5 == 0)
3883              {
3884                gotoxy(16,19);textcolor(12+128);
3885                cprintf("OFF");
3886              }
```

```
3887            }
3888            goto loop;
3889          }
3890 excite1_toggle:{
3891                    if(flag_D == 1)
3892                    {
3893                      flag5 = 1;// <4>
3894                      flag6 = 0;
3895                      flag7 = 0;
3896                      flag8 = 0;
3897                      flag9 = 0;
3898                      flag12 = 0;
3899                      flag13 = 0;
3900                      num = 4;
3901                      gotoxy(16,19);textcolor(10);
3902                      cprintf("ON ");
3903                      gotoxy(13,21);textcolor(15);
3904                      cprintf("            ");// Erase "PW: %6.4f  "
3905                      gotoxy(21,21);textcolor(10);
3906                      cprintf("D.A.");
3907                      flag_D = 0;
3908                      flag_E = 1;
3909                      flag_F = 1;
3910                      flag_G = 1;
3911                      flag_H = 1;
3912                      flag_I = 1;
3913                      flag_J = 1;
3914
3915                      rr = 0;
3916                      OL = 0.0;
3917                      L_T = 0.0;
3918                      qq = 0;
3919                      ii = 0.0;
3920
3921                      goto loop;
3922                    }
3923                    else
3924                    if(flag_D == 0)
3925                    {
3926                      if(flag5 == 1)
3927                      {
3928                        flag5 = 0;
3929                        gotoxy(16,19);textcolor(12+128);
3930                        cprintf("OFF");
3931                        flag_D = 1;
3932                        flag_E = 1;
3933                        flag_F = 1;
3934                        flag_G = 1;
3935                        flag_H = 1;
3936                        flag_I = 1;
3937                        flag_J = 1;
3938                      }
3939                      goto loop;
3940                    }
3941                  }
3942 excite2:{
3943            if(flag_E == 1)
3944            {
```

```
3945              COUNTMAX = 15.0;
3946              flag5 = 0;
3947              flag6 = 1;// <5>
3948              flag7 = 0;
3949              flag8 = 0;
3950              flag9 = 0;
3951              flag12 = 0;
3952              flag13 = 0;
3953              gotoxy(2,19);textcolor(14);
3954              cprintf("< >EXCITATION    ");
3955              gotoxy(3,19);textcolor(14);
3956              cprintf("5");
3957              num = 5;
3958              gotoxy(16,19);textcolor(10);
3959              cprintf("ON ");
3960              gotoxy(13,21);textcolor(15);
3961              cprintf("            ");
3962              gotoxy(21,21);textcolor(10);
3963              cprintf("D.A.");
3964              flag_E = 0;
3965              flag_D = 1;
3966              flag_F = 1;
3967              flag_G = 1;
3968              flag_H = 1;
3969              flag_I = 1;
3970              flag_J = 1;
3971
3972              rr = 0;
3973              OL = 0.0;
3974              L_T = 0.0;
3975              qq = 0;
3976              ii = 0.0;
3977
3978              goto loop;
3979            }
3980            else
3981            if(flag_E == 0)
3982            {
3983              if(flag6 == 1)
3984              {
3985                flag6 = 0;
3986                gotoxy(16,19);textcolor(12+128);
3987                cprintf("OFF");
3988                flag_E = 1;
3989                flag_D = 1;
3990                flag_F = 1;
3991                flag_G = 1;
3992                flag_H = 1;
3993                flag_I = 1;
3994                flag_J = 1;
3995              }
3996              goto loop;
3997            }
3998          }
3999 excite3:{
4000            if(flag_F == 1)
4001            {
4002              COUNTMAX = 15.0;
```

```
4003              k = 0;
4004              flag5 = 0;
4005              flag6 = 0;
4006              flag7 = 1;// <6>
4007              flag8 = 0;
4008              flag9 = 0;
4009              flag12 = 0;
4010              flag13 = 0;
4011              gotoxy(2,19);textcolor(14);
4012              cprintf("< >EXCITATION    ");
4013              gotoxy(3,19);textcolor(14);
4014              cprintf("6");
4015              num = 6;
4016              gotoxy(16,19);textcolor(10);
4017              cprintf("ON ");
4018              gotoxy(13,21);textcolor(15);
4019              cprintf("            ");
4020              gotoxy(21,21);textcolor(10);
4021              cprintf("D.A.");
4022              flag_F = 0;
4023              flag_D = 1;
4024              flag_E = 1;
4025              flag_G = 1;
4026              flag_H = 1;
4027              flag_I = 1;
4028              flag_J = 1;
4029
4030              rr = 0;
4031              OL = 0.0;
4032              L_T = 0.0;
4033              qq = 0;
4034              ii = 0.0;
4035
4036              goto loop;
4037            }
4038            else
4039            if(flag_F == 0)
4040            {
4041              if(flag7 == 1)
4042              {
4043                flag7 = 0;
4044                gotoxy(16,19);textcolor(12+128);
4045                cprintf("OFF");
4046                flag_F = 1;
4047                flag_D = 1;
4048                flag_E = 1;
4049                flag_G = 1;
4050                flag_H = 1;
4051                flag_I = 1;
4052                flag_J = 1;
4053              }
4054              goto loop;
4055            }
4056          }
4057 excite4:{
4058            if(flag_G == 1)
4059            {
4060              COUNTMAX = 15.0;
```

```
4061              k = 0;
4062              flag5 = 0;
4063              flag6 = 0;
4064              flag7 = 0;
4065              flag8 = 1;// <7>
4066              flag9 = 0;
4067              flag12 = 0;
4068              flag13 = 0;
4069              gotoxy(2,19);textcolor(14);
4070              cprintf("< >EXCITATION    ");
4071              gotoxy(3,19);textcolor(14);
4072              cprintf("7");
4073              num = 7;
4074              gotoxy(16,19);textcolor(10);
4075              cprintf("ON ");
4076              gotoxy(13,21);textcolor(15);
4077              cprintf("            ");
4078              gotoxy(21,21);textcolor(10);
4079              cprintf("D.A.");
4080              flag_G = 0;
4081              flag_D = 1;
4082              flag_E = 1;
4083              flag_F = 1;
4084              flag_H = 1;
4085              flag_I = 1;
4086              flag_J = 1;
4087
4088              rr = 0;
4089              OL = 0.0;
4090              L_T = 0.0;
4091              qq = 0;
4092              ii = 0.0;
4093
4094              goto loop;
4095            }
4096            else
4097            if(flag_G == 0)
4098            {
4099              if(flag8 == 1)
4100              {
4101                flag8 = 0;
4102                gotoxy(16,19);textcolor(12+128);
4103                cprintf("OFF");
4104                flag_G = 1;
4105                flag_D = 1;
4106                flag_E = 1;
4107                flag_F = 1;
4108                flag_H = 1;
4109                flag_I = 1;
4110                flag_J = 1;
4111              }
4112              goto loop;
4113            }
4114          }
4115 excite5:{
4116            if(flag_H == 1)
4117            {
4118              COUNTMAX = 15.0;
```

```
4119              k1 = 1;
4120              flag5 = 0;
4121              flag6 = 0;
4122              flag7 = 0;
4123              flag8 = 0;
4124              flag9 = 1;// <8>
4125              flag12 = 0;
4126              flag13 = 0;
4127              gotoxy(2,19);textcolor(14);
4128              cprintf("< >EXCITATION    ");
4129              gotoxy(3,19);textcolor(14);
4130              cprintf("8");
4131              num = 8;
4132              gotoxy(16,19);textcolor(10);
4133              cprintf("ON ");
4134              gotoxy(13,21);textcolor(15);
4135              cprintf("PW: %6.4f  ",PW);
4136              flag_H = 0;
4137              flag_D = 1;
4138              flag_E = 1;
4139              flag_F = 1;
4140              flag_G = 1;
4141              flag_I = 1;
4142              flag_J = 1;
4143
4144              rr = 0;
4145              OL = 0.0;
4146              L_T = 0.0;
4147              qq = 0;
4148              ii = 0.0;
4149
4150              goto loop;
4151            }
4152            else
4153            if(flag_H == 0)
4154            {
4155              if(flag9 == 1)
4156              {
4157                flag9 = 0;
4158                gotoxy(16,19);textcolor(12+128);
4159                cprintf("OFF");
4160
4161                gotoxy(13,21);textcolor(10);
4162                cprintf("        D.A.");
4163
4164                flag_H = 1;
4165                flag_D = 1;
4166                flag_E = 1;
4167                flag_F = 1;
4168                flag_G = 1;
4169                flag_I = 1;
4170                flag_J = 1;
4171              }
4172              goto loop;
4173            }
4174          }
4175 excite6:{
4176            if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
```

```
4177          {
4178            if(flag_I == 1)
4179            {
4180              COUNTMAX = 15.0;
4181              k = 0;
4182              flag5 = 0;
4183              flag6 = 0;
4184              flag7 = 0;
4185              flag8 = 0;
4186              flag9 = 0;
4187              flag12 = 1;// <9>
4188              flag13 = 0;
4189              gotoxy(2,19);textcolor(14);
4190              cprintf("< >EXCITATION    ");
4191              gotoxy(3,19);textcolor(14);
4192              cprintf("9");
4193              num = 9;
4194              gotoxy(16,19);textcolor(10);
4195              cprintf("ON ");
4196              gotoxy(13,21);textcolor(15);
4197              cprintf("            ");
4198              gotoxy(21,21);textcolor(10);
4199              cprintf("D.A.");
4200              flag_I = 0;
4201              flag_D = 1;
4202              flag_E = 1;
4203              flag_F = 1;
4204              flag_G = 1;
4205              flag_H = 1;
4206              flag_J = 1;
4207
4208              rr = 0;
4209              OL = 0.0;
4210              L_T = 0.0;
4211              qq = 0;
4212              ii = 0.0;
4213
4214              goto loop;
4215            }
4216            else
4217            if(flag_I == 0)
4218            {
4219              if(flag12 == 1)
4220              {
4221                flag12 = 0;
4222                gotoxy(16,19);textcolor(12+128);
4223                cprintf("OFF");
4224                flag_I = 1;
4225                flag_D = 1;
4226                flag_E = 1;
4227                flag_F = 1;
4228                flag_G = 1;
4229                flag_H = 1;
4230                flag_J = 1;
4231              }
4232              goto loop;
4233            }
4234          }//End of if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
```

```
4235            goto loop;
4236          }
4237 excite7:{
4238            if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
4239            {
4240              if(flag_J == 1)
4241              {
4242                COUNTMAX = 15.0;
4243                k1 = 1;
4244                flag5 = 0;
4245                flag6 = 0;
4246                flag7 = 0;
4247                flag8 = 0;
4248                flag9 = 0;
4249                flag12 = 0;
4250                flag13 = 1;// <0>
4251                gotoxy(2,19);textcolor(14);
4252                cprintf("< >EXCITATION    ");
4253                gotoxy(3,19);textcolor(14);
4254                cprintf("0");
4255                num = 0;
4256                gotoxy(16,19);textcolor(10);
4257                cprintf("ON ");
4258                gotoxy(13,21);textcolor(15);
4259                cprintf("            ");
4260                gotoxy(21,21);textcolor(10);
4261                cprintf("D.A.");
4262                flag_J = 0;
4263                flag_D = 1;
4264                flag_E = 1;
4265                flag_F = 1;
4266                flag_G = 1;
4267                flag_H = 1;
4268                flag_I = 1;
4269
4270                rr = 0;
4271                OL = 0.0;
4272                L_T = 0.0;
4273                qq = 0;
4274                ii = 0.0;
4275
4276                goto loop;
4277              }
4278
4279              else
4280
4281              if(flag_J == 0)
4282              {
4283                if(flag13 == 1)
4284                {
4285                  flag13 = 0;
4286                  gotoxy(16,19);textcolor(12+128);
4287                  cprintf("OFF");
4288                  flag_J = 1;
4289                  flag_D = 1;
4290                  flag_E = 1;
4291                  flag_F = 1;
4292                  flag_G = 1;
```

```
4293                flag_H = 1;
4294                flag_I = 1;
4295              }
4296              goto loop;
4297            }
4298          }// if((flag1 == 1 || flag2 == 1 || flag3 == 1) && flag23 == 1)
4299          goto loop;
4300        }
4301 modal:{
4302        rr = 0;
4303        OL = 0.0;
4304        L_T = 0.0;
4305        qq = 0;
4306        ii = 0.0;
4307        COUNTMAX = 15.0;
4308
4309        if(diag == 1)
4310        {
4311          flag44 = 0;
4312          gotoxy(26,20);textcolor(15);
4313          cprintf("         ");// Erase "Force (N)"
4314          gotoxy(25,21);textcolor( 4);
4315          cprintf("           ");// Erasr "==========="
4316          gotoxy(42,12);textcolor(4);
4317          cprintf("                   ");// Erase "==================="
4318          gotoxy(42,15);textcolor(14);
4319          cprintf("                   ");// Erase "==================="
4320          gotoxy(22,22);textcolor(15);
4321          cprintf("  ");// Erase "x:"
4322          gotoxy(22,23);textcolor(15);
4323          cprintf("                          ");// Erase "y:"
4324          gotoxy(42,13);
4325          cprintf("                   ");// Erase "kh_bot<g>"
4326          gotoxy(42,14);
4327          cprintf("                   ");// Erase "dh_bot<d>"
4328          gotoxy(21,17);// Erase "offset_bot<t>"
4329          cprintf("                                        ");
4330          gotoxy(21,18);// Erase "offset_bot<w>"
4331          cprintf("                                        ");
4332          gotoxy(21,19);// Erase "bias_current_bot<b>"
4333          cprintf("                                              ");
4334          gotoxy(21,13);textcolor(11);
4335          cprintf("k_tilt      :");
4336          gotoxy(34,13);textcolor(15);
4337          cprintf("%6.2f", k_tilt);
4338          gotoxy(21,14);textcolor(11);
4339          cprintf("c_tilt      :");
4340          gotoxy(34,14);textcolor(15);
4341          cprintf("%6.2f",c_tilt);
4342          gotoxy(65, 9);textcolor(15);
4343          cprintf("l");
4344          gotoxy(65,10);textcolor(15);
4345          cprintf("u");
4346          gotoxy(65,11);textcolor(15);
4347          cprintf("z");
4348          gotoxy(61,21);textcolor(15);
4349          cprintf("( )");
4350          if(flag_GG == 1)
```

```
4351          {
4352            COUNTMAX = 15.0;
4353            flag10 = 1;
4354            gotoxy(52,5);textcolor(15);
4355            cprintf("==>                    <==");
4356            gotoxy(56,5);textcolor(14+128);
4357            cprintf("MODAL CONTROLLER");
4358            if(lu == 'l')
4359            {
4360              gotoxy(62,21);textcolor(15+128);
4361              cprintf("L");
4362            }
4363            else
4364            if(lu == 'u')
4365            {
4366              gotoxy(62,21);textcolor(15+128);
4367              cprintf("U");
4368            }
4369            gotoxy(16,18);textcolor(10);
4370            cprintf("ON ");
4371            gotoxy(25,22);
4372            cprintf("           ");// Erase x: along with output value
4373            flagJJ = 0;// Initialize toggle to "TILT MODE"
4374            flag_GG = 0;// Toggle condition
4375            goto loop;
4376          }
4377          else
4378          if(flag_GG == 0)
4379          {
4380            lu = 'l';
4381            flag10 = 0;
4382            flag15 = 1;
4383            gotoxy(16,18);textcolor(12+128);
4384            cprintf("OFF");
4385            gotoxy(57,5);
4386            cprintf("                          ");// Erase "MODAL CONTROLLER"
4387            gotoxy(52,5);textcolor(14+128);
4388            cprintf("==>                    <==");
4389            gotoxy(57,5);textcolor(10);
4390            cprintf("LOWER  BEARING");
4391            gotoxy(65, 9);textcolor(15+128);
4392            cprintf("l");
4393            gotoxy(61,21);textcolor(15);
4394            cprintf("   ");// Erase (L) & (U)
4395            gotoxy(31,8);textcolor(9);
4396            cprintf(" <c>CG factor: %5.2f",CG);
4397            gotoxy(21,13);textcolor(9);
4398            cprintf("kv_bot<p>   :%6.2f",kv_bot);
4399            gotoxy(42,13);textcolor(9);
4400            cprintf("kh_bot<g>   :%6.2f",kh_bot);
4401            gotoxy(21,14);textcolor(9);
4402            cprintf("dv_bot<v>   :%6.2f",dv_bot);
4403            gotoxy(42,14);textcolor(9);
4404            cprintf("dh_bot<d>   :%6.2f",dh_bot);
4405            gotoxy(21,17);textcolor(9);
4406            cprintf("offset_bot<t>                        :");
4407            gotoxy(55,17);textcolor(9);
4408            cprintf("%5d",tBias_bot);
```

```
4409            gotoxy(21,18);textcolor(9);
4410            cprintf("offset_bot<w>                          :");
4411            gotoxy(55,18);textcolor(9);
4412            cprintf("%5d",wBias_bot);
4413            gotoxy(21,19);textcolor(9);
4414            cprintf("offset current_bot<b>                  :");
4415            gotoxy(55,19);textcolor(9);
4416            cprintf("%6.2f Amp.",        ibias_bot);
4417            gotoxy(26,20);textcolor(15);
4418            cprintf("Force (N)");
4419            gotoxy(25,21);textcolor( 4);
4420            cprintf("===========");
4421            gotoxy(51,20);textcolor(15);
4422            cprintf("x_value         y_value");
4423            gotoxy(51,21);textcolor(4);
4424            cprintf("=======         =======");
4425            if(nw_bot == 1)
4426            {
4427              gotoxy(22,22);textcolor(15);
4428              cprintf("x:");
4429              gotoxy(22,23);textcolor(15);
4430              cprintf("y:");
4431            }
4432            gotoxy(49,24);textcolor(15);
4433            cprintf("  +       -       +       -  ");
4434            gotoxy(49,25);textcolor(15);
4435            cprintf("  X       X       Y       Y  ");
4436            gotoxy(19,11);textcolor(15);
4437            cprintf("         Y_AXIS                X_AXIS");
4438            gotoxy(36,11);textcolor(13);
4439            cprintf("< >-test: %1u",test_signal);
4440            gotoxy(37,11);textcolor(15);
4441            cprintf("M");
4442            gotoxy(21,12);textcolor(4);
4443            cprintf("==================  ==================");
4444            gotoxy(21,15);textcolor(14);
4445            cprintf("==================  ==================");
4446
4447            flag15 = 1;
4448            flag11 = 1;// Lower bearing write out block activated
4449            flag22 = 0;// Upper bearing write out block deactivated
4450            flag33 = 0;// Thrust bearing write out block deactivated
4451            flag23 = 1;// Enable key press "9 & 0"
4452            flag_GG = 1;
4453
4454            goto loop;
4455          }// End of if(flag_GG == 0)
4456        }// End (diag == 1)
4457        else
4458        if(diag == 0)
4459        {
4460          if(flag_GG == 1)
4461          {
4462            flag10 = 1;
4463            gotoxy(52,5);textcolor(15+128);
4464            cprintf("==>                  <==");
4465            gotoxy(56,5);textcolor(14);
4466            cprintf("MODAL CONTROLLER");
```

```
4467              gotoxy(16,18);textcolor(10);
4468              cprintf("ON ");
4469              flag_GG = 0;// Toggle condition
4470              goto loop;
4471            }
4472            else
4473            if(flag_GG == 0)
4474            {
4475              flag10 = 0;
4476              gotoxy(52,5);// Erase("==>                    <==")
4477              cprintf("                          ");
4478              gotoxy(16,18);textcolor(12+128);
4479              cprintf("OFF");
4480              flag_GG = 1;// Toggle condition
4481              goto loop;
4482            }
4483          }
4484          goto loop;
4485        }
4486 disable_safe1:{
4487                 sg1 = 0;
4488                 gotoxy(16,15);textcolor(12+128);
4489                 cprintf("OFF");
4490                 goto loop;
4491               }
4492 enable_safe1:{
4493                sg1 = 1;
4494                gotoxy(16,15);textcolor(10);
4495                cprintf("ON ");
4496                goto loop;
4497              }
4498 disable_safe2:{
4499                 sg2 = 0;
4500                 gotoxy(16,16);textcolor(12+128);
4501                 cprintf("OFF");
4502                 goto loop;
4503               }
4504 enable_safe2:{
4505                sg2 = 1;
4506                gotoxy(16,16);textcolor(10);
4507                cprintf("ON ");
4508                goto loop;
4509              }
4510 disable_safe3:{
4511                 sg3 = 0;
4512                 gotoxy(16,17);textcolor(12+128);
4513                 cprintf("OFF");
4514                 goto loop;
4515               }
4516 enable_safe3:{
4517                sg3 = 1;
4518                gotoxy(16,17);textcolor(10);
4519                cprintf("ON ");
4520                goto loop;
4521              }
4522 cg_factor_up:{
4523                CG = CG + 0.01;
4524                if(CG > 0.5)
```

```
4525                    CG = 0.5;
4526                  MCG = 0.5 - CG;
4527                  PCG = 0.5 + CG;
4528                  gotoxy(46,8);textcolor(15);
4529                  cprintf("%5.2f", CG);
4530                  goto loop;
4531                 }
4532 cg_factor_down:{
4533                   CG = CG - 0.01;
4534                   if(CG < -0.5)
4535                      CG = -0.5;
4536                   MCG = 0.5 - CG;
4537                   PCG = 0.5 + CG;
4538                   gotoxy(46,8);textcolor(15);
4539                   cprintf("%5.2f", CG);
4540                   goto loop;
4541                  }
4542 igain_up:{
4543            if(flag3 == 1 && flag15 == 0)
4544            {
4545              igainth = igainth + 0.0001;
4546              gotoxy(44,8);textcolor(15);
4547              cprintf("%7.4f", igainth);
4548              goto loop;
4549            }
4550            goto loop;
4551          }
4552 igain_down:{
4553               if(flag3 == 1 && flag15 == 0)
4554               {
4555                 igainth = igainth - 0.0001;
4556                 gotoxy(44,8);
4557                 printf("%7.4f", igainth);
4558                 goto loop;
4559               }
4560               goto loop;
4561             }
4562 buffer:{
4563          if(flag_FF == 1)// Toggle flag
4564          {
4565            flag4d = 1;// Buffer on
4566            gotoxy(45,16);textcolor(10);
4567            cprintf("ON ");
4568            flag_FF = 0;
4569            goto loop;
4570          }
4571          else
4572          if(flag_FF == 0)// Toggle flag
4573          {
4574            flag4d = 0;// Buffer off
4575            gotoxy(45,16);textcolor(12+128);
4576            cprintf("OFF");
4577            flag_FF = 1;
4578            goto loop;
4579          }
4580        }
4581 diagnostic:{
4582               gotoxy(37,19);textcolor(14);
```

```
4583            cprintf("        ");// Erase NASA, GLENN, RESEARCH, CENTER
4584            gotoxy(10,21);
4585            cprintf
4586            ("                                                                ");
4587            O      = 1.0;
4588            flag5  = 0;// |
4589            flag6  = 0;// |
4590            flag7  = 0;// |
4591            flag8  = 0;// |  Shut down excitor functions.
4592            flag9  = 0;// |
4593            flag12 = 0;// |
4594            flag13 = 0;// |
4595            flag16 = 1;// Assembly condition (on)
4596            flag18 = 0;
4597            flag21 = 0;// Excitation switch
4598            flag10 = 0;// Turn off modal block
4599            flag44 = 0;// Enable D.A./I.A. display
4600
4601            flag_A = 1;// Assembly toggle set to on
4602            flag_B = 1;// f_excite toggle set to on
4603            flag_C = 1;// Excitation toggle set to on
4604            flag_D = 1;
4605            flag_E = 1;
4606            flag_F = 1;
4607            flag_G = 1;
4608            flag_H = 1;
4609            flag_I = 1;
4610            flag_J = 1;
4611            flag_M = 1;
4612            flag_N = 1;
4613            flagKK = 1;
4614            flag_II = 0;
4615
4616            flag4a = 0;// Shuts down Lower bearing buffer
4617            flag4b = 0;// Shuts down Upper bearing buffer
4618            flag4c = 0;// Shuts down Thrust bearing buffer
4619
4620            rr = 0;
4621            OL = 0.0;
4622            L_T = 0.0;
4623            qq = 0;
4624            ii = 0.0;
4625            diag = 1;
4626            SSS  = 1;// <---- Condition necessary to access
4627                     //       diagnostic parameter controls.
4628            flag_GG = 1;
4629
4630            COUNTMAX = 15.0;
4631            gotoxy(31,8);textcolor(9);
4632            cprintf(" <c>CG factor: %5.2f",CG);
4633            gotoxy(32,16);textcolor(14);
4634            cprintf("[loop buffer    ]");
4635
4636            if(flag4d == 1)
4637            {
4638              gotoxy(45,16);textcolor(10);
4639              cprintf("ON ");
4640            }
```

```
4641            else
4642            if(flag4d == 0)
4643            {
4644              gotoxy(45,16);textcolor(12+128);
4645              cprintf("OFF");
4646            }
4647            gotoxy(30,2);
4648            printf("                        ");// Erase DT
4649            gotoxy(23,13);
4650            cprintf("                                ");// Erase LBE
4651            gotoxy(23,14);
4652            cprintf("                                ");// Erase UBE
4653            gotoxy(23,15);
4654            cprintf("                                ");// Erase TBE
4655            gotoxy(48,2);textcolor(12);
4656            cprintf("  * Thrst bearing is energized  !");
4657            gotoxy(48,3);textcolor(12);
4658            cprintf("  * Upper bearing is energized  !");
4659            gotoxy(48,4);textcolor(12);
4660            cprintf("  * Lower bearing is energized  !");
4661            gotoxy(52,5);textcolor(14+128);
4662            cprintf("==>                    <==");
4663            gotoxy(57,5);textcolor(10);
4664            cprintf("LOWER  BEARING");
4665            gotoxy(1,1);textcolor(15);
4666            cprintf ("<+,-> to toggle input-output writes");
4667            gotoxy(1,2);textcolor(15);
4668            cprintf("<q> to abort control");
4669            gotoxy(1,3);textcolor(15);
4670            cprintf("<f> to toggle loop time buffer");
4671            gotoxy(1,4);textcolor(15);
4672            cprintf("<e> non diagnostic       ");
4673            gotoxy(1,5);textcolor(15);
4674            cprintf("<!,@,#> Disable safe gain  ");
4675            gotoxy(19,11);textcolor(15);
4676            cprintf("         Y_AXIS                X_AXIS");
4677            gotoxy(36,11);textcolor(13);
4678            cprintf("< >-test: %1u",test_signal);
4679            gotoxy(37,11);textcolor(15);
4680            cprintf("M");
4681            gotoxy(21,12);textcolor(4);
4682            cprintf("==================  ==================");
4683            gotoxy(21,15);textcolor(14);
4684            cprintf("==================  ==================");
4685            gotoxy(21,13);textcolor(9);
4686            cprintf("kv_bot<p>   :%6.2f",kv_bot);
4687            gotoxy(42,13);textcolor(9);
4688            cprintf("kh_bot<g>   :%6.2f",kh_bot);
4689            gotoxy(21,14);textcolor(9);
4690            cprintf("dv_bot<v>   :%6.2f",dv_bot);
4691            gotoxy(42,14);textcolor(9);
4692            cprintf("dh_bot<d>   :%6.2f",dh_bot);
4693            gotoxy(21,17);textcolor(9);
4694            cprintf("offset_bot<t>                        :");
4695            gotoxy(55,17);textcolor(9);
4696            cprintf("%5d",tBias_bot);
4697            gotoxy(21,18);textcolor(9);
4698            cprintf("offset_bot<w>                        :");
```

```
4699            gotoxy(55,18);textcolor(9);
4700            cprintf("%5d",wBias_bot);
4701            gotoxy(21,19);textcolor(9);
4702            cprintf("bias current_bot<b>              : ");
4703            gotoxy(55,19);textcolor(9);
4704            cprintf("%6.2f Amp.",        ibias_bot);
4705            gotoxy(51,20);textcolor(15);
4706            cprintf("x_value         y_value");
4707            gotoxy(51,21);textcolor(4);
4708            cprintf("=======         =======");
4709            gotoxy(49,24);
4710            textcolor(15);
4711            cprintf("  +       -       +       -  ");
4712            gotoxy(49,25);
4713            textcolor(15);
4714            cprintf("  X       X       Y       Y  ");
4715            gotoxy(64, 7);textcolor(11);cprintf("Display Parameter");
4716            gotoxy(64, 8);textcolor(15);cprintf("==================");
4717            gotoxy(64, 9);textcolor(13);cprintf("< >Lower Bearing");
4718            gotoxy(65, 9);textcolor(15);cprintf("l");
4719            gotoxy(64,10);textcolor(13);cprintf("< >Upper Bearing");
4720            gotoxy(65,10);textcolor(15);cprintf("u");
4721            gotoxy(64,11);textcolor(13);cprintf("< >Thrst Bearing");
4722            gotoxy(65,11);textcolor(15);cprintf("z");
4723            gotoxy(64,13);textcolor(11);cprintf("Energizing Parmtr");
4724            gotoxy(64,14);textcolor(15);cprintf("==================");
4725            gotoxy(64,15);textcolor(13);cprintf("<H>Thrst Bearing");
4726            gotoxy(65,15);textcolor(15);cprintf("H");
4727            gotoxy(64,16);textcolor(13);cprintf("< >Upper Bearing");
4728            gotoxy(65,16);textcolor(15);cprintf("I");
4729            gotoxy(64,17);textcolor(13);cprintf("< >Lower Bearing");
4730            gotoxy(65,17);textcolor(15);cprintf("J");
4731            gotoxy(2,18);textcolor(14);
4732            cprintf("< >MODAL CTRL   ");
4733            gotoxy(3,18);textcolor(15+128);
4734            cprintf("m");
4735            gotoxy(16,18);textcolor(12+128);
4736            cprintf("OFF");
4737            gotoxy(2,19);textcolor(14);
4738            cprintf("< >EXCITATION    ");
4739            gotoxy(2,19);textcolor(14);
4740            cprintf("<%u>EXCITATION    ",num);
4741            gotoxy(16,19);textcolor(12+128);
4742            cprintf("OFF");
4743            gotoxy(26,20);textcolor(15);
4744            cprintf("Force (N)");
4745            gotoxy(25,21);textcolor(4 );
4746            cprintf("==========");
4747            gotoxy(1,20);textcolor(15);
4748            cprintf("<x>Frq_inpt:%7.2f Hz.",freq);
4749            gotoxy(1,25);textcolor(15);
4750            cprintf("<s>to adjust Pulse Width");
4751            gotoxy(27,24);textcolor(14);
4752            cprintf("[<,> Enable exction.]");
4753            gotoxy(28,25);textcolor(14);
4754            cprintf("[<:> Assembly    ]");
4755            gotoxy(42,25);textcolor(10);
4756            cprintf("ON");
```

```
4757              goto loop;
4758            }
4759 non_diagnostic:{
4760                  clrscr();
4761                  O      = 1.0;
4762                  flag5  = 0;// |
4763                  flag6  = 0;// |
4764                  flag7  = 0;// |
4765                  flag8  = 0;// |  Shut down excitor functions.
4766                  flag9  = 0;// |
4767                  flag12 = 0;// |
4768                  flag13 = 0;// |
4769                  flag16 = 1;// Assembly condition (on)
4770                  flag10 = 0;// Turn off modal block
4771
4772                  flag_A = 1;// Assembly toggle set to on
4773                  flag_B = 1;// f_excite toggle set to on
4774                  flag_C = 1;// Excitation toggle set to on
4775                  flag_D = 1;
4776                  flag_E = 1;
4777                  flag_F = 1;
4778                  flag_G = 1;
4779                  flag_H = 1;
4780                  flag_I = 1;
4781                  flag_J = 1;
4782                  flag_M = 1;
4783                  flag_N = 1;
4784                  flagKK = 1;
4785
4786                  flag_II = 0;
4787                  flag_CC = 0;
4788                  flag_DD = 0;
4789                  flag_EE = 0;
4790
4791                  flag18 = 0;
4792                  flag21 = 0;// Excitation switch
4793                  flag4  = 0;
4794                  SSS    = 0;
4795                  flag_GG = 1;
4796
4797                  COUNTMAX = 15.0;
4798                  gotoxy(1,1);textcolor(15);
4799                  cprintf("<x/k> to adjust frequency");
4800                  gotoxy(1,2);textcolor(15);
4801                  cprintf("<q> to abort control");
4802                  gotoxy(1,3);textcolor(15);
4803                  cprintf("<m> to toggle modal cntrl");
4804                  gotoxy(1,4);textcolor(15);
4805                  cprintf("<?> to toggle f_excite");
4806                  gotoxy(1,5);textcolor(15);
4807                  cprintf("<4-0> to select excitation");
4808                  gotoxy(59,1);textcolor(15);
4809                  cprintf("[ file :  FiveAx.c   ]");
4810                  gotoxy(23,14);
4811                  cprintf("                                    ");// Erase TBF
4812                  gotoxy(31,2);textcolor(11);
4813                  cprintf("DIAGNOSTIC TOGGLE<E>");
4814                  gotoxy(31,8);textcolor(9);
```

```
4815                cprintf(" <c>CG factor: %5.2f",CG);
4816                gotoxy(27, 9);textcolor(10);
4817                cprintf("[ loop time:       micro-sec ]");
4818                gotoxy(1,8);textcolor(15);cprintf("[   THE MAGNETIC  ]");
4819                gotoxy(1,9);textcolor(15);cprintf("[BEARING SYSTEM IS]");
4820                gotoxy(1,10);textcolor(15);cprintf("[                 ]");
4821                gotoxy(9,11);textcolor(15);cprintf("|");
4822                gotoxy(9,12);textcolor(15);cprintf("|");
4823                gotoxy(4,10);textcolor(12+128);
4824                cprintf("OPERATIONAL !  ");
4825                gotoxy(26,13);textcolor(14);
4826                cprintf("==>                          <==");
4827                gotoxy(30,13);textcolor(12+128);
4828                cprintf("THRST BEARING ENERGIZED");
4829                gotoxy(26,14);textcolor(14);
4830                cprintf("==>                          <==");
4831                gotoxy(30,14);textcolor(12+128);
4832                cprintf("UPPER BEARING ENERGIZED");
4833                gotoxy(26,15);textcolor(14);
4834                cprintf("==>                          <==");
4835                gotoxy(30,15);textcolor(12+128);
4836                cprintf("LOWER BEARING ENERGIZED");
4837                nw_bot = 0;
4838                nw_top = 0;
4839                nw_th  = 0;
4840                gotoxy(1,22);textcolor(13);
4841                cprintf("<Excitation Parmtr>");
4842                gotoxy(1,14);textcolor(10);
4843                cprintf("< >PHSE ANG:%3u deg",th);
4844                gotoxy(2,14);textcolor(15);
4845                cprintf("n");
4846                gotoxy(1,24);textcolor(15);
4847                cprintf("<a>Amplitude:%4.1f v O-pk",volt);
4848                gotoxy(1,20);textcolor(15);
4849                cprintf("<x>Frq_inpt:%7.2f Hz.",freq);
4850                gotoxy(1,25);textcolor(15);
4851                cprintf("<s>to adjust Pulse Width");
4852                gotoxy(48,22);
4853                printf("                                  ");
4854                gotoxy(46,23);
4855                printf("                                     ");
4856
4857                diag  = 0;
4858
4859                flag1 = 1;// Lower bearing block activated
4860                flag2 = 1;// Upper bearing block activated
4861                flag3 = 1;// Thrust bearing block activated
4862
4863                flag11 = 1;
4864                flag22 = 1;
4865                flag33 = 1;
4866                flag44 = 0;// Enable D.A./I.A. display
4867
4868                sg1 = 1;
4869                sg2 = 1;
4870                sg3 = 1;
4871
4872                gotoxy(1,15);textcolor(15);
```

```
4873                    cprintf("[                  ]");
4874                    gotoxy(2,15);textcolor(14);
4875                    cprintf("Lwr Safe Gain    ");
4876                    gotoxy(16,15);textcolor(10);
4877                    cprintf("ON ");
4878                    gotoxy(1,16);textcolor(15);
4879                    cprintf("[                  ]");
4880                    gotoxy(2,16);textcolor(14);
4881                    cprintf("Upr Safe Gain    ");
4882                    gotoxy(16,16);textcolor(10);
4883                    cprintf("ON ");
4884                    gotoxy(1,17);textcolor(15);
4885                    cprintf("[                  ]");
4886                    gotoxy(2,17);textcolor(14);
4887                    cprintf("Tht Safe Gain    ");
4888                    gotoxy(16,17);textcolor(10);
4889                    cprintf("ON ");
4890                    gotoxy(1,18);textcolor(15);
4891                    cprintf("[                  ]");
4892                    gotoxy(2,18);textcolor(14);
4893                    cprintf("MODAL CNTRL      ");
4894                    gotoxy(16,18);textcolor(12+128);
4895                    cprintf("OFF");
4896                    gotoxy(1,19);textcolor(15);
4897                    cprintf("[                  ]");
4898                    gotoxy(2,19);textcolor(14);
4899                    cprintf("<%u>EXCITATION     ",num);
4900                    gotoxy(16,19);textcolor(12+128);
4901                    cprintf("OFF");
4902                    gotoxy(27,24);textcolor(14);
4903                    cprintf("[<,> Enable exction.]");
4904                    gotoxy(28,25);textcolor(14);
4905                    cprintf("[<:> Assembly    ]");
4906                    gotoxy(42,25);textcolor(10);
4907                    cprintf("ON");
4908                    goto loop;
4909                  }
4910 lower_bearing:{
4911                  if(flag_CC == 1)
4912                  {
4913                    gotoxy(48,4);textcolor(12);
4914                    cprintf("  * Lower bearing is energized  !");
4915                    gotoxy(65,17);textcolor(15);cprintf("J");
4916                    flag1  = 1;
4917                    flag4a = 0;// Shuts down Lower bearing buffer
4918                    flag_CC = 0;
4919                    goto loop;
4920                  }
4921                  else
4922                  if(flag_CC == 0)
4923                  {
4924                    gotoxy(52,4);textcolor(14+128);
4925                    cprintf("Lower bearing not energized !");
4926                    gotoxy(65,17);textcolor(15+128);cprintf("J");
4927                    flag1 = 0;
4928                    flag4a = 1;// Turn on Lower bearing buffer
4929                    flag_CC = 1;
4930                    goto loop;
```

```
4931                }
4932              }
4933 upper_bearing:{
4934                if(flag_DD == 1)
4935                {
4936                  gotoxy(48,3);textcolor(12);
4937                  cprintf("  * Upper bearing is energized  !");
4938                  gotoxy(65,16);textcolor(15);cprintf("I");
4939                  flag2  = 1;
4940                  flag4b = 0;// Shuts down Upper bearing buffer
4941                  flag_DD = 0;
4942                  goto loop;
4943                }
4944                else
4945                if(flag_DD == 0)
4946                {
4947                  gotoxy(52,3);textcolor(14+128);
4948                  cprintf("Upper bearing not energized !");
4949                  gotoxy(65,16);textcolor(15+128);cprintf("I");
4950                  flag2  = 0;
4951                  flag4b = 1;// Turn on Upper bearing buffer
4952                  flag_DD = 1;
4953                  goto loop;
4954                }
4955              }
4956 thrust_bearing:{
4957                 if(flag_EE == 1)
4958                 {
4959                   gotoxy(48,2);textcolor(12);
4960                   cprintf("  * Thrst bearing is energized  !");
4961                   gotoxy(65,15);textcolor(15);cprintf("H");
4962                   flag3  = 1;
4963                   flag4c = 0;// Shuts down Thrust bearing buffer
4964                   flag_EE = 0;
4965                   goto loop;
4966                 }
4967                 else
4968                 if(flag_EE == 0)
4969                 {
4970                   gotoxy(52,2);textcolor(14+128);
4971                   cprintf("Thrst bearing not energized !");
4972                   gotoxy(65,15);textcolor(15+128);cprintf("H");
4973                   flag3  = 0;
4974                   flag4c = 1;// Surn on Thrust bearing buffer
4975                   flag_EE = 1;
4976                   goto loop;
4977                 }
4978               }
4979 l_on:{
4980        if(flag10 == 0)// Disable this block when in modal mode
4981        {
4982          gotoxy(31,8);textcolor(9);
4983          cprintf(" <c>CG factor: %5.2f",CG);
4984          gotoxy(52,5);textcolor(14+128);
4985          cprintf("==>                  <==");
4986          gotoxy(57,5);textcolor(10);
4987          cprintf("LOWER  BEARING");
4988          gotoxy(21,13);textcolor(9);
```

```
4989          cprintf("kv_bot<p>   :%6.2f",kv_bot);
4990          gotoxy(42,13);textcolor(9);
4991          cprintf("kh_bot<g>   :%6.2f",kh_bot);
4992          gotoxy(21,14);textcolor(9);
4993          cprintf("dv_bot<v>   :%6.2f",dv_bot);
4994          gotoxy(42,14);textcolor(9);
4995          cprintf("dh_bot<d>   :%6.2f",dh_bot);
4996          gotoxy(21,17);textcolor(9);
4997          cprintf("offset_bot<t>                        :");
4998          gotoxy(55,17);textcolor(9);
4999          cprintf("%5d",tBias_bot);
5000          gotoxy(21,18);textcolor(9);
5001          cprintf("offset_bot<w>                        :");
5002          gotoxy(55,18);textcolor(9);
5003          cprintf("%5d",wBias_bot);
5004          gotoxy(21,19);textcolor(9);
5005          cprintf("bias current_bot<b>                :");
5006          gotoxy(55,19);textcolor(9);
5007          cprintf("%6.2f Amp.",        ibias_bot);
5008          gotoxy(26,20);textcolor(15);
5009          cprintf("Force (N)");
5010          gotoxy(25,21);textcolor( 4);
5011          cprintf("===========");
5012          if(nw_bot == 1)
5013          {
5014            gotoxy(22,22);textcolor(15);
5015            cprintf("x:");
5016            gotoxy(22,23);textcolor(15);
5017            cprintf("y:");
5018          }
5019          gotoxy(51,20);textcolor(15);
5020          cprintf("x_value         y_value");
5021          gotoxy(51,21);textcolor(4);
5022          cprintf("=======         =======");
5023          gotoxy(27,23);
5024          cprintf("                      ");// Erase [<^> to toggle D.A. ]
5025          gotoxy(49,24);textcolor(15);
5026          cprintf("  +       -       +       -  ");
5027          gotoxy(49,25);textcolor(15);
5028          cprintf("  X       X       Y       Y  ");
5029          gotoxy(19,11);textcolor(15);
5030          cprintf("         Y_AXIS                X_AXIS");
5031          gotoxy(36,11);textcolor(13);
5032          cprintf("< >-test: %1u",test_signal);
5033          gotoxy(37,11);textcolor(15);
5034          cprintf("M");
5035          gotoxy(21,12);textcolor(4);
5036          cprintf("==================  ==================");
5037          gotoxy(21,15);textcolor(14);
5038          cprintf("==================  ==================");
5039          gotoxy(65,9);textcolor(15+128);
5040          cprintf("l");
5041          gotoxy(65,10);textcolor(15);
5042          cprintf("u");
5043          gotoxy(65,11);textcolor(15);
5044          cprintf("z");
5045
5046          flag15 = 1;
```

```
5047        }// End of if(flag10 == 0)
5048        if(flag10 == 0 || flag10 == 1)
5049        {
5050          lu = 'l';
5051          if(nw_bot == 1)
5052          {
5053            gotoxy(37,22);textcolor(10);
5054            cprintf("Displacement:");
5055          }
5056          if(flag10 == 1)
5057          {
5058            gotoxy(62,21);textcolor(15+128);
5059            cprintf("L");
5060          }
5061
5062          flag11 = 1;// Lower bearing write out block activated
5063          flag22 = 0;// Upper bearing write out block deactivated
5064          flag33 = 0;// Thrust bearing write out block deactivated
5065        }// End of if(flag10 == 0 || flag10 == 1)
5066        flag23 = 1;// Enable key press "9 & 0"
5067        goto loop;
5068      }
5069 u_on:{
5070       if(flag10 == 0)// Disable this block when in modal mode
5071       {
5072         gotoxy(31,8);textcolor(9);
5073         cprintf(" <c>CG factor: %5.2f",CG);
5074         gotoxy(52,5);textcolor(14+128);
5075         cprintf("==>                    <==");
5076         gotoxy(57,5);textcolor(10);
5077         cprintf("UPPER  BEARING");
5078         gotoxy(21,13);textcolor(9);
5079         cprintf("kv_top<p>   :%6.2f",kv_top);
5080         gotoxy(42,13);textcolor(9);
5081         cprintf("kh_top<g>   :%6.2f",kh_top);
5082         gotoxy(21,14);textcolor(9);
5083         cprintf("dv_top<v>   :%6.2f",dv_top);
5084         gotoxy(42,14);textcolor(9);
5085         cprintf("dh_top<d>   :%6.2f",dh_top);
5086         gotoxy(21,17);textcolor(9);
5087         cprintf("offset_top<t>                        :");
5088         gotoxy(55,17);textcolor(9);
5089         cprintf("%5d",tBias_top);
5090         gotoxy(21,18);textcolor(9);
5091         cprintf("offset_top<w>                        :");
5092         gotoxy(55,18);textcolor(9);
5093         cprintf("%5d",wBias_top);
5094         gotoxy(21,19);textcolor(9);
5095         cprintf("bias current_top<b>                :");
5096         gotoxy(55,19);textcolor(9);
5097         cprintf("%6.2f Amp.",        ibias_top);
5098         gotoxy(26,20);textcolor(15);
5099         cprintf("Force (N)");
5100         gotoxy(25,21);textcolor( 4);
5101         cprintf("===========");
5102         if(nw_top == 1)
5103         {
5104           gotoxy(22,22);textcolor(15);
```

```
5105            cprintf("x:");
5106            gotoxy(22,23);textcolor(15);
5107            cprintf("y:");
5108          }
5109          gotoxy(51,20);textcolor(15);
5110          cprintf("x_value         y_value");
5111          gotoxy(51,21);textcolor(4);
5112          cprintf("=======         =======");
5113          gotoxy(27,23);
5114          cprintf("                      ");// Erase [<^> to toggle D.A. ]
5115          gotoxy(49,24);textcolor(15);
5116          cprintf("  +       -       +       -  ");
5117          gotoxy(49,25);textcolor(15);
5118          cprintf("  X       X       Y       Y  ");
5119          gotoxy(19,11);textcolor(15);
5120          cprintf("         Y_AXIS               X_AXIS");
5121          gotoxy(36,11);textcolor(13);
5122          cprintf("< >-test: %1u",test_signal);
5123          gotoxy(37,11);textcolor(15);
5124          cprintf("M");
5125          gotoxy(21,12);textcolor(4);
5126          cprintf("==================  ==================");
5127          gotoxy(21,15);textcolor(14);
5128          cprintf("==================  ==================");
5129          gotoxy(65,9);textcolor(15);
5130          cprintf("l");
5131          gotoxy(65,10);textcolor(15+128);
5132          cprintf("u");
5133          gotoxy(65,11);textcolor(15);
5134          cprintf("z");
5135
5136          flag15 = 1;
5137        }// End of if(flag10 == 0)
5138        if(flag10 == 0 || flag10 == 1)
5139        {
5140          lu = 'u';
5141          if(nw_top == 1)
5142          {
5143            gotoxy(37,22);textcolor(10);
5144            cprintf("Displacement:");
5145          }
5146          if(flag10 == 1)
5147          {
5148            gotoxy(62,21);textcolor(15+128);
5149            cprintf("U");
5150          }
5151          flag11 = 0;// Lower bearing write out block deactivated
5152          flag22 = 1;// Upper bearing write out activated
5153          flag33 = 0;// Thrust bearing write out block deactivated
5154        }// End of if(flag10 == 0 || flag10 == 1)
5155        flag23 = 1;// Enable key press "9 & 0"
5156        goto loop;
5157      }
5158 z_on:{
5159         if(flag10 == 0)// Disable this block when in modal mode
5160         {
5161           gotoxy(31,8);textcolor(9);
5162           cprintf("<(,)>igainth:%7.4f", igainth);
```

```
5163          gotoxy(52,5);textcolor(14+128);
5164          cprintf("==>                    <==");
5165          gotoxy(57,5);textcolor(10);
5166          cprintf("THRUST BEARING");
5167          gotoxy(21,13);textcolor(9);
5168          cprintf("kv_th<p>    :%6.2f",kv_th);
5169          gotoxy(42,13);textcolor(9);
5170          cprintf("                    ");// Erase top right half
5171          gotoxy(21,14);textcolor(9);
5172          cprintf("dv_th<v>    :%6.2f",dv_th);
5173          gotoxy(42,14);textcolor(9);
5174          cprintf("                    ");// Erase bottom right half
5175          gotoxy(21,17);textcolor(9);
5176          cprintf("offset_th<t>                          :");
5177          gotoxy(55,17);textcolor(9);
5178          cprintf("%5d",tBias_th);
5179          gotoxy(21,18);textcolor(9);// Erase wBias_th
5180          cprintf("                                          ");
5181          gotoxy(21,19);textcolor(9);
5182          cprintf("bias current_th<b>                :");
5183          gotoxy(55,19);textcolor(9);
5184          cprintf("%6.2f Amp.", ibias_th);
5185          gotoxy(50,20);textcolor(15);
5186          cprintf(" z_value                  ");
5187          gotoxy(50,21);textcolor(4);
5188          cprintf(" =======                  ");
5189          gotoxy(26,20);textcolor(15);
5190          cprintf("Force (N)");
5191          gotoxy(25,21);textcolor( 4);
5192          cprintf("===========");
5193          gotoxy(24,22);// Erase x:
5194          printf("                                                    ");
5195          gotoxy(22,23);// Erase y:
5196          printf
5197          ("                                                        ");
5198          gotoxy(49,24);textcolor(15);
5199          cprintf("  +       -                   ");
5200          gotoxy(49,25);textcolor(15);
5201          cprintf("  Z       Z                   ");
5202          gotoxy(19,11);textcolor(15);
5203          cprintf("         Z_AXIS                        ");
5204          gotoxy(36,11);textcolor(13);
5205          cprintf("< >-test: %lu",test_signal);
5206          gotoxy(37,11);textcolor(15);
5207          cprintf("M");
5208          gotoxy(21,12);textcolor(4);
5209          cprintf("==================                    ");
5210          gotoxy(21,15);textcolor(14);
5211          cprintf("==================                    ");
5212          gotoxy(22,22);textcolor(15);
5213          cprintf("  ");// Erase "x:"
5214          if(nw_th == 1)
5215          {
5216            gotoxy(37,22);textcolor(10);
5217            cprintf("Displacement:");
5218            gotoxy(22,22);textcolor(15);
5219            cprintf("z:");
5220          }
```

```
5221          gotoxy(65,9);textcolor(15);
5222          cprintf("l");
5223          gotoxy(65,10);textcolor(15);
5224          cprintf("u");
5225          gotoxy(65,11);textcolor(15+128);
5226          cprintf("z");
5227
5228          flag15 = 0;
5229          flag11 = 0;// Lower bearing write out block deactivated
5230          flag22 = 0;// Upper bearing write out block deactivated
5231          flag33 = 1;// Thrust bearing write out block activated
5232          flag23 = 0;// Disable key press "9 & 0"
5233          flag_GG = 1;
5234
5235          goto loop;
5236        }// End of if(flag10 == 0)
5237       goto loop;
5238      }
5239 kv_up:{
5240         if(flag10 == 0)
5241         {
5242           if(flag11 == 1)
5243           {
5244             kv_bot = kv_bot + 0.1;
5245             gotoxy(34,13);textcolor(15);
5246             cprintf("%6.2f", kv_bot);
5247             goto loop;
5248           }
5249           else
5250           if(flag22 == 1)
5251           {
5252             kv_top = kv_top + 0.1;
5253             gotoxy(34,13);textcolor(15);
5254             cprintf("%6.2f", kv_top);
5255             goto loop;
5256           }
5257           else
5258           if(flag33 == 1)
5259           {
5260             kv_th = kv_th + 0.1;
5261             gotoxy(34,13);textcolor(15);
5262             cprintf("%6.2f", kv_th);
5263             goto loop;
5264           }
5265         }// End of if(flag10 == 0)
5266        goto loop;
5267       }
5268 kv_down:{
5269          if(flag10 == 0)
5270          {
5271            if(flag11 == 1)
5272            {
5273              kv_bot = kv_bot - 0.1;
5274              gotoxy(34,13); printf("%6.2f", kv_bot);
5275              goto loop;
5276            }
5277            else
5278            if(flag22 == 1)
```

```
5279            {
5280              kv_top = kv_top - 0.1;
5281              gotoxy(34,13); printf("%6.2f", kv_top);
5282              goto loop;
5283            }
5284            else
5285            if(flag33 == 1)
5286            {
5287              kv_th = kv_th - 0.1;
5288              gotoxy(34,13); printf("%6.2f", kv_th);
5289              goto loop;
5290            }
5291          }
5292          goto loop;
5293         }
5294 dh_up:{
5295        if(flag10 == 0)
5296        {
5297          if(flag11 == 1)
5298          {
5299            dh_bot = dh_bot + 0.5;
5300            gotoxy(55,14);textcolor(15);
5301            cprintf("%6.2f", dh_bot);
5302            goto loop;
5303          }
5304          else
5305          if(flag22 == 1)
5306          {
5307            dh_top = dh_top + 0.5;
5308            gotoxy(55,14);textcolor(15);
5309            cprintf("%6.2f", dh_top);
5310            goto loop;
5311          }
5312        }
5313        goto loop;
5314       }
5315 dh_down:{
5316           if(flag10 == 0)
5317           {
5318             if(flag11 == 1)
5319             {
5320               dh_bot = dh_bot - 0.5;
5321               gotoxy(55,14); printf("%6.2f", dh_bot);
5322               goto loop;
5323             }
5324             else
5325             if(flag22 == 1)
5326             {
5327               dh_top = dh_top - 0.5;
5328               gotoxy(55,14); printf("%6.2f", dh_top);
5329               goto loop;
5330             }
5331           }
5332           goto loop;
5333         }
5334 kh_up:{
5335         if(flag10 == 0)
5336         {
```

```
5337            if(flag11 == 1)
5338            {
5339              kh_bot = kh_bot + 0.1;
5340              gotoxy(55,13);textcolor(15);
5341              cprintf("%6.2f", kh_bot);
5342              goto loop;
5343            }
5344            else
5345            if(flag22 == 1)
5346            {
5347              kh_top = kh_top + 0.1;
5348              gotoxy(55,13);textcolor(15);
5349              cprintf("%6.2f", kh_top);
5350              goto loop;
5351            }
5352          }
5353          goto loop;
5354        }
5355 kh_down:{
5356            if(flag10 == 0)
5357            {
5358              if(flag11 == 1)
5359              {
5360                kh_bot = kh_bot - 0.1;
5361                gotoxy(55,13); printf("%6.2f", kh_bot);
5362                goto loop;
5363              }
5364              else
5365              if(flag22 == 1)
5366              {
5367                kh_top = kh_top - 0.1;
5368                gotoxy(55,13); printf("%6.2f", kh_top);
5369                goto loop;
5370              }
5371            }
5372            goto loop;
5373          }
5374 dv_up:{
5375          if(flag10 == 0)
5376          {
5377            if(flag11 == 1)
5378            {
5379              dv_bot = dv_bot + 0.5;
5380              gotoxy(34,14);textcolor(15);
5381              cprintf("%6.2f", dv_bot);
5382              goto loop;
5383            }
5384            else
5385            if(flag22 == 1)
5386            {
5387              dv_top = dv_top + 0.5;
5388              gotoxy(34,14);textcolor(15);
5389              cprintf("%6.2f", dv_top);
5390              goto loop;
5391            }
5392            else
5393            if(flag33 == 1)
5394            {
```

```
5395              dv_th = dv_th + 0.5;
5396              gotoxy(34,14);textcolor(15);
5397              cprintf("%6.2f", dv_th);
5398              goto loop;
5399            }
5400          }
5401          goto loop;
5402        }
5403 dv_down:{
5404            if(flag10 == 0)
5405            {
5406              if(flag11 == 1)
5407              {
5408                dv_bot = dv_bot - 0.5;
5409                gotoxy(34,14); printf("%6.2f", dv_bot);
5410                goto loop;
5411              }
5412              else
5413              if(flag22 == 1)
5414              {
5415                dv_top = dv_top - 0.5;
5416                gotoxy(34,14); printf("%6.2f", dv_top);
5417                goto loop;
5418              }
5419              else
5420              if(flag33 == 1)
5421              {
5422                dv_th = dv_th - 0.5;
5423                gotoxy(34,14); printf("%6.2f", dv_th);
5424                goto loop;
5425              }
5426            }
5427            goto loop;
5428          }
5429 wBias_up:{
5430             if(flag10 == 0)
5431             {
5432               if(flag11 == 1)
5433               {
5434                 wBias_bot = wBias_bot + 5;
5435                 gotoxy(55,18);textcolor(15);
5436                 cprintf("%5d", wBias_bot);
5437                 goto loop;
5438               }
5439               else
5440               if(flag22 == 1)
5441               {
5442                 wBias_top = wBias_top + 5;
5443                 gotoxy(55,18);textcolor(15);
5444                 cprintf("%5d", wBias_top);
5445                 goto loop;
5446               }
5447             }
5448             goto loop;
5449           }
5450 wBias_down:{
5451               if(flag10 == 0)
5452               {
```

```
5453                if(flag11 == 1)
5454                {
5455                  wBias_bot = wBias_bot - 5;
5456                  gotoxy(55,18);printf("%5d", wBias_bot);
5457                  goto loop;
5458                }
5459                else
5460                if(flag22 == 1)
5461                {
5462                  wBias_top = wBias_top - 5;
5463                  gotoxy(55,18);printf("%5d", wBias_top);
5464                  goto loop;
5465                }
5466              }
5467              goto loop;
5468            }
5469 tBias_up:{
5470            if(flag10 == 0)
5471            {
5472              if(flag11 == 1)
5473              {
5474                tBias_bot = tBias_bot + 5;
5475                gotoxy(55,17);textcolor(15);
5476                cprintf("%5d", tBias_bot);
5477                goto loop;
5478              }
5479              else
5480              if(flag22 == 1)
5481              {
5482                tBias_top = tBias_top + 5;
5483                gotoxy(55,17);textcolor(15);
5484                cprintf("%5d", tBias_top);
5485                goto loop;
5486              }
5487              else
5488              if(flag33 == 1)
5489              {
5490                tBias_th = tBias_th + 5;
5491                gotoxy(55,17);textcolor(15);
5492                cprintf("%5d", tBias_th);
5493                goto loop;
5494              }
5495            }
5496            goto loop;
5497          }
5498 tBias_down:{
5499              if(flag10 == 0)
5500              {
5501                if(flag11 == 1)
5502                {
5503                  tBias_bot = tBias_bot - 5;
5504                  gotoxy(55,17); printf("%5d", tBias_bot);
5505                  goto loop;
5506                }
5507                else
5508                if(flag22 == 1)
5509                {
5510                  tBias_top = tBias_top - 5;
```

```
5511                    gotoxy(55,17); printf("%5d", tBias_top);
5512                    goto loop;
5513                  }
5514                  else
5515                  if(flag33 == 1)
5516                  {
5517                    tBias_th = tBias_th - 5;
5518                    gotoxy(55,17); printf("%5d", tBias_th);
5519                    goto loop;
5520                  }
5521                }
5522                goto loop;
5523              }
5524 writeout:{
5525              if(flag_B == 1)
5526              {
5527                if(flag11 == 1)
5528                {
5529                  gotoxy(37,22);textcolor(10+128);
5530                  cprintf("Displacement:");
5531                  if(flag10 == 0)
5532                  {
5533                    gotoxy(22,22);textcolor(15+128);
5534                    cprintf("x:");
5535                    gotoxy(22,23);textcolor(15+128);
5536                    cprintf("y:");
5537                  }
5538                  nw_bot = 1;// Write out enabled
5539                  gotoxy(27,23);
5540                  cprintf("                         ");// Erase [<^> to toggle D.A. ]
5541                  goto loop;
5542                }
5543                else
5544                if(flag22 == 1)
5545                {
5546                  gotoxy(37,22);textcolor(10+128);
5547                  cprintf("Displacement:");
5548                  if(flag10 == 0)
5549                  {
5550                    gotoxy(22,22);textcolor(15+128);
5551                    cprintf("x:");
5552                    gotoxy(22,23);textcolor(15+128);
5553                    cprintf("y:");
5554                  }
5555                  nw_top = 1;// Write out enabled
5556                  gotoxy(27,23);
5557                  cprintf("                         ");// Erase [<^> to toggle D.A. ]
5558                  goto loop;
5559                }
5560                else
5561                if(flag33 == 1)
5562                {
5563                  gotoxy(37,22);textcolor(10+128);
5564                  cprintf("Displacement:");
5565                  gotoxy(22,22);textcolor(15+128);
5566                  cprintf("z:");
5567                  nw_th = 1;// Write out enabled
5568                  gotoxy(27,23);
```

```
5569                cprintf("                          ");// Erase [<^> to toggle D.A. ]
5570                goto loop;
5571              }
5572            }// End of if(flag_B == 1)
5573            goto loop;
5574          }
5575 nowrite:{
5576           if(flag_B == 1)
5577           {
5578             if(flag11 == 1)
5579             {
5580               flag44 = 0;// Enable D.A/I.A. display
5581               nw_bot = 0;// Write out disabled
5582               gotoxy(22,22);
5583               printf("                                                    ");
5584               gotoxy(22,23);
5585               printf
5586               ("                                                  ");
5587               goto loop;
5588             }
5589             else
5590             if(flag22 == 1)
5591             {
5592               flag44 = 0;// Enable D.A/I.A. display
5593               nw_top = 0;// Write out disabled
5594               gotoxy(22,22);
5595               printf("                                                    ");
5596               gotoxy(22,23);
5597               printf
5598               ("                                                  ");
5599               goto loop;
5600             }
5601             else
5602             if(flag33 == 1)
5603             {
5604               flag44 = 0;// Enable D.A/I.A. display
5605               nw_th = 0;// Write out disabled
5606               gotoxy(22,22);
5607               printf("                                                    ");
5608               gotoxy(22,23);
5609               printf
5610               ("                                                  ");
5611               goto loop;
5612             }
5613           }// End of if(flag_B == 1)
5614           goto loop;
5615         }
5616 bias_up:{
5617           if(flag10 == 0)
5618           {
5619             if(flag11 == 1)
5620             {
5621               ibias_bot = ibias_bot + 0.1;
5622               bias_current_bot = round1(ibias_bot * 2.0 * 204.8);
5623               gotoxy(55,19);textcolor(15);
5624               cprintf("%6.2f", ibias_bot);
5625               goto loop;
5626             }
```

```
5627              else
5628              if(flag22 == 1)
5629              {
5630                ibias_top = ibias_top + 0.1;
5631                bias_current_top = round1(ibias_top * 2.0 * 204.8);
5632                gotoxy(55,19);textcolor(15);
5633                cprintf("%6.2f", ibias_top);
5634                goto loop;
5635              }
5636              else
5637              if(flag33 == 1)
5638              {
5639                ibias_th = ibias_th + 0.1;
5640                bias_current_th = round1(ibias_th * 2.0 * 204.8);
5641                gotoxy(55,19);textcolor(15);
5642                cprintf("%6.2f", ibias_th);
5643                goto loop;
5644              }
5645            }
5646            goto loop;
5647          }
5648 bias_down:{
5649              if(flag10 == 0)
5650              {
5651                if(flag11 == 1)
5652                {
5653                  ibias_bot = ibias_bot - 0.1;
5654                  bias_current_bot = round1(ibias_bot * 2.0 * 204.8);
5655                  gotoxy(55,19);
5656                  printf("%6.2f", ibias_bot);
5657                  goto loop;
5658                }
5659                else
5660                if(flag22 == 1)
5661                {
5662                  ibias_top = ibias_top - 0.1;
5663                  bias_current_top = round1(ibias_top * 2.0 * 204.8);
5664                  gotoxy(55,19); printf("%6.2f", ibias_top);
5665                  goto loop;
5666                }
5667                else
5668                if(flag33 == 1)
5669                {
5670                  ibias_th = ibias_th - 0.1;
5671                  bias_current_th = round1(ibias_th * 2.0 * 204.8);
5672                  gotoxy(55,19); printf("%6.2f", ibias_th);
5673                  goto loop;
5674                }
5675              }
5676              goto loop;
5677            }
5678
5679 /*----------------------RAMP DOWN WHILE SUPPORTED--------------------------*/
5680
5681 ramp_down:{
5682              gotoxy(10,6);textcolor(14);
5683              cprintf("CONTROL RAMPING DOWN.....          ");
5684
```

```
5685              outport(out_chan1_0, t48);
5686              outport(out_chan1_1, t48);
5687              outport(out_chan1_2, t48);
5688              outport(out_chan1_3, t48);
5689              outport(out_chan1_4, t48);
5690              outport(out_chan1_5, t48);
5691
5692 //           outport(out_chan2_0, t48);
5693              outport(out_chan2_1, t48);
5694              outport(out_chan2_2, t48);
5695              outport(out_chan2_3, t48);
5696              outport(out_chan2_4, t48);
5697              outport(out_chan2_5, t48);
5698
5699              gotoxy(31,7);textcolor(14);
5700              cprintf(" ......COMPLETE !     ");
5701
5702              gotoxy(1, 8);printf("[   THE MAGNETIC  ]");
5703              gotoxy(1, 9);printf("[BEARING SYSTEM IS]");
5704              gotoxy(1,10);printf("[                 ]");
5705              gotoxy(8,10);textcolor(10+128);
5706              cprintf("OFF !\a\a   ");
5707              if(diag == 0)
5708              {
5709                gotoxy(26,13);
5710                cprintf("                                ");// ERASE LBE
5711                gotoxy(26,14);
5712                cprintf("                                ");// ERASE UBE
5713                gotoxy(26,15);
5714                cprintf("                                ");// ERASE TBE
5715              }
5716 loop3:       gotoxy(1,13);textcolor(14);
5717              cprintf("                    ");// ERASE AYS
5718
5719              gotoxy(3,13);textcolor(10);
5720              cprintf("CONTINUE(y/n)?:");
5721
5722              resp = getch();
5723
5724              if(resp == 'y')
5725              {
5726                if(diag == 0)
5727                {
5728                  gotoxy(26,13);textcolor(14);
5729                  cprintf("==>                          <==");
5730                  gotoxy(30,13);textcolor(12+128);
5731                  cprintf("LOWER BEARING ENERGIZED");
5732                  gotoxy(26,14);textcolor(14);
5733                  cprintf("==>                          <==");
5734                  gotoxy(30,14);textcolor(12+128);
5735                  cprintf("UPPER BEARING ENERGIZED");
5736                  gotoxy(26,15);textcolor(14);
5737                  cprintf("==>                          <==");
5738                  gotoxy(30,15);textcolor(12+128);
5739                  cprintf("THRST BEARING ENERGIZED");
5740                }
5741              }
5742
```

```
5743              if (resp == 'y' || resp == 'Y')
5744              {
5745                gotoxy(10,6); printf("                                ");// CRD
5746                gotoxy(31,7); printf("                                ");// C
5747                gotoxy(3,13);printf("                 ");// ERASE "CONTINUE?"
5748                gotoxy(1,8);textcolor(15);
5749                cprintf("[   THE MAGNETIC  ]");
5750                gotoxy(1,9);textcolor(15);
5751                cprintf("[BEARING SYSTEM IS]");
5752                gotoxy(1,10);textcolor(15);
5753                cprintf("[                  ]");
5754                gotoxy(4,10);textcolor(12+128);
5755                cprintf("OPERATIONAL !\a  ");
5756                flag_L = 0;
5757                goto loop;
5758              }
5759              else
5760              if (resp == 'n' || resp == 'N')
5761              goto loop2;
5762              goto loop3;
5763            }// End ramp_down:
5764 loop2: textcolor(7);cprintf("\b");clrscr();
5765 return(0);
5766 }//                * * * End of main function * * *
5767
5768 float round1(float u)
5769 {
5770   int g,v;
5771   float z;
5772
5773   g = ceil(u);
5774   z = u + 0.5;
5775
5776   if(g >= z)
5777      v = floor(u);
5778   else
5779      v = g;
5780   return (v);
5781 }
```

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	3. REPORT TYPE AND DATES COVERED
	April 2001	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	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	