

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0380311 4

ORNL/TM-12645

ornl

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

Software for Goniometer Control in the
Triple Ion Implantation Facility

W. R. Allen

OAK RIDGE NATIONAL LABORATORY

CENTRAL RESEARCH LIBRARY

CIRCULATION SECTION

4000N ROOM 173

LIBRARY LOAN COPY

DO NOT TRANSFER TO ANOTHER PERSON

If you wish someone else to see this
report, send in name with report and
the library will arrange a loan.

UCR/PSL/4 1973

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401. FTS 620-8401

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161

This report was prepared as an account of work sponsored by an agency of the United States Government. It is neither the United States Government nor any agency thereof, nor is it the Department of Energy, nor is it the Government, nor does it endorse or recommend any specific commercial product or process, or state, or imply that its use is necessary, or that it is the best available. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-12645

504 25

95 32

Metals and Ceramics Division

SOFTWARE FOR GONIOMETER CONTROL IN THE
TRIPLE ION IMPLANTATION FACILITY

W. R. Allen

Date Published: February 1994

NOTICE: This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

Prepared for
Office of Basic Energy Sciences
KC 02 01 04 0

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6285
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



3 4456 0380311 4

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	1
1. INTRODUCTION	1
2. PROCEDURAL BACKGROUND	2
3. SPECIFICS OF THE SOFTWARE	3
4. GUIDELINES FOR OPERATION OF TIF CHANNELING SETUP	5
5. ADDITIONAL COMMENTS	7
6. ACKNOWLEDGMENTS	7
APPENDIX A: COMPUTER PROGRAM FOR GONIOMETER CONTROL ...	9

SOFTWARE FOR GONIOMETER CONTROL IN THE TRIPLE ION IMPLANTATION FACILITY*

W. R. Allen

ABSTRACT

A computer program is described that controls the goniometer employed in the ion scattering chamber of the Triple Ion Implantation Facility (TIF) in the Metals and Ceramics Division at Oak Ridge National Laboratory. Details of goniometer operation and its incorporation into the ion scattering setup specific to the TIF are also discussed.

1. INTRODUCTION

Ion-channeling applications in materials science require the accurate positioning of a crystalline direction of high symmetry relative to an analysis ion beam. If the momentum vectors of the incident ion trajectories are nearly collinear with a major crystal axis or plane, the scattering yield from constituent atoms of the host is reduced due to the steering effects of channeling and shadowing. Channeling can be exploited for alignment of a crystal symmetry direction with the ion beam direction. This document presents a computer code written for control of a dual-axis goniometer utilized in the Triple Ion Implantation Facility (TIF) of the Metals and Ceramics Division at the Oak Ridge National Laboratory. A generic goniometer is an instrument that precisely manipulates an attached target with two, or more, orthogonal motion stages. The target is mechanically attached flush to a flat sample holder on the goniometer head, which places it near the intersection of the motion axes. The two-axis goniometer of the TIF permits a 90° tilt (θ) about an axial length of the target and a 360° rotation (ϕ) about an axis perpendicular to the surface normal of the sample holder. This surface normal and the ion beam direction coincide for $\theta = 0^\circ$. Each axis can be positioned with an accuracy and reproducibility of 0.01° . The multi-function goniometer

*Research supported by the Division of Materials Sciences, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

control software facilitates crystal alignment and general target positioning. External device control was required for the stepping motors automating the two motion stages of the goniometer and for two counters interfaced to the data collection electronics.

2. PROCEDURAL BACKGROUND

Major crystal planes are identified in the initial channeling alignment phase as the target, tilted several degrees in the azimuthal angle (i.e., θ), is rotated in the polar angle (i.e., θ) where $\theta = 0^\circ$ is the ion beam direction. The Miller indices of individual planes are ascertained from relative angular position and the magnitude of the reduction in scattering yield. The coarse angular position of planar intersection should correspond to a major crystal axis. Further rocking in θ and ϕ about this position, while monitoring the scattering yield, can accurately pinpoint the location of a crystal axis relative to the surface normal and to the ion beam direction. Detailed post-alignment measurements along the identified directions of high symmetry can yield information regarding, for example, changes in crystallinity and the lattice location of impurity atoms. A sequence of energy spectra taken about a direction of high symmetry constitutes what is commonly known as an angular scan.

An energy spectrum represents ions scattered from the target during bombardment by an analysis ion beam. These ions can be produced following either elastic or inelastic interactions with target atoms. Biased to create a depletion layer free of charge carriers, a surface-barrier detector positioned in the high-vacuum chamber intercepts all ions incident within a fixed solid angle of acceptance. The number of electron-hole pairs created per ion is proportional to the energy lost in the depletion layer. If the ion is stopped, the number of electron-hole pairs is proportional to the full kinetic energy. A pulse is output to a preamplifier where it is shaped and amplified. A calibrated spectroscopy amplifier further shapes and amplifies the signal. An analog-to-digital converter (ADC) incorporated into a multichannel analyzer (MCA) converts the analog voltage into a digital form. In a process known as pulse height analysis, the digital signals are sorted according to pulse height to form an energy spectrum.

3. SPECIFICS OF THE SOFTWARE

The computer code for goniometer control was generated with Microsoft QuickBasic version 4.5 for use on IBM-compatible microcomputer (PC). The software code controlling the MCA can be run concurrently. The goniometer control program utilizes a random access memory (RAM) drive, designated as E: drive, to store needed temporary information. A graphics library from SutraSoft, called INGRAF, is applied to graphically present results of alignment diagrams produced by the subroutines PolarGraph and LinearGraph. Descriptions of the routines in the INGRAF library can be found in its documentation. Several custom subroutines, developed by the author, were adapted for use, including those that communicate with the stepping motor controller and provide cursor control on-screen. From within the QuickBasic programming environment, the source code of the goniometer control program can be modified and recompiled into a stand-alone executable form. The INGRAF library must be loaded from the command line with QuickBasic at design time with the sequence "*QB/L INGRAF.*" A complete source listing of the goniometer control program is given in Appendix A.

The PC communicates with a stepping motor controller at a 1200-baud rate with no parity, 7-bit words, and 2 stop bits utilizing RS-232 communications over serial port 1. The two stepping motors of the goniometer are driven by an indexer/driver commanded by the stepping motor controller. Automated angular scans can be acquired for either θ or ϕ . Plots of scattering yield versus angular position are produced graphically, and a hard copy can be output. The screen mode chosen for display is 640 by 480 pixel video graphics array (VGA) graphics, and hard copies are generated on an Epson-compatible printer. Either the video mode or printer type can be modified by altering the appropriate variables in calls from PolarGraph and LinearGraph to the INGRAF subroutine GOPEN.

Operation of the goniometer control software is menu driven with a series of user windows. A menu item is selected with the cursor arrows, or highlighted letter, and the ENTER key pressed to prompt for additional input. Return to the MCA software via "*Return to MCA,*" not Ctrl-C or Ctrl-Break. These control sequences for program termination are to be utilized if the program hangs or to immediately halt execution of a command. In these instances, the current angular position is irretrievably lost. To recover, the last known position can be input as an offset after returning θ and ϕ to those coordinates using the knobs on the unpowered stepping motors.

To obviate the slow process of downloading movement parameters at 1200 baud, twelve preset movements for both θ and ϕ are stored in the non-volatile memory of the stepping motor controller. These are accessed from the menu selection "*Execute Preset Move.*" The option "*Do Arbitrary Move*" permits motion at a preset angular speed for arbitrary values input for θ and ϕ . The optimum angular speeds for motion have been determined previously. Home position denotes the values of θ and ϕ which correspond to the ordered pair $(0^\circ, 0^\circ)$ as recognized by the code and controller. The menu selection "*Home Position Set*" assigns the current physical position to $(0^\circ, 0^\circ)$. The offset, selected with "*Offset Theta/Phi,*" permits an initial offset relative to the physical position to be entered. Following power up, the controller and code both initialize at $(0^\circ, 0^\circ)$. On a shell to the MCA software via "*Return to MCA,*" the angular position is stored on the RAM drive until the goniometer control code is reentered. The menu choice "*Phi Rock*" initiates slow rotation in ϕ of the target at $0.25^\circ/\text{s}$ while acquiring an energy spectrum having a random-equivalent crystal orientation. The motion proceeds in ϕ for the input angular amount in a positive sense, and then the rotation direction is reversed to return to the initial value of ϕ . Prior to acquisition, an estimate should be made regarding the time required to acquire the random-equivalent energy spectrum. During acquisition, the goniometer control screen should be monitored for an end-of-movement condition. If additional motion is needed, then respond accordingly. The keystrokes Ctrl-C or Ctrl-Break can return the user to the MCA software to view the accumulating energy spectrum. Returning to the goniometer control code presents a blank screen to the user that will refresh only when the rotation ends. Frequently appearing in the body of the code is the instruction "FOR I = 1 TO 100: NEXT I," which inserts a time delay into the code to permit successful communications between the stepping motor controller and the PC.

The PC communicates with two counters via a 32-bit input/output (I/O) card in the back plane of the PC. Using 5-V transistor-transistor logic (TTL), four groups of eight individual bits can be used for input or output. As detailed in the subroutine called Counter, three groups are used here for receiving the hexadecimal number of counts indicated on the display of one counter. This counter has a peripheral interface that permits the above operations to be performed. Pin 47 monitors the input from the interval BNC connection on the counter, which indicates a high logic level when a preset is achieved. One group outputs a TTL high signal to reset and zero both counters.

As mentioned previously, manual and automatic modes of angular scanning for crystal alignment are possible. If the automated angular scan mode is desired, connect the ribbon cable between the digital I/O card interface in the back plane of the control computer and the interface of a counter. This counter registers the accumulated number of scattering events in an energy (i.e., pulse height) range specified by the output of a single channel analyzer (SCA). For manual operation, disconnect this cable. A second counter monitors the output of a current digitizer sensing the beam current on target. An energy spectrum is acquired for a specific fluence of analysis ions. When the preset is achieved, a trigger signal from the counter halts counting by the MCA. In addition, the ion beam is automatically shuttered by inserting a Faraday cup to minimize unwanted damage to the target. The number of scattering events in the range specified by the SCA window is recorded and the target indexed to the next orientation. In automatic mode, the PC positions the target in θ and ϕ according to input parameters, reads and records the number of counts, and resets both counters. In manual mode, the number of counts is recorded by hand, the counters reset manually with a remote switch, and the target repositioned as necessary by a series of movements.

To acquire an energy spectrum, the angular position of the sample is set, and one returns to the MCA control code. An energy spectrum is acquired for a preset dose of analysis ions chosen with the preset on the counter interfaced to the current integrator. The energy spectrum is stored, and one shells to the goniometer control code for repositioning of the sample. This process is repeated as necessary.

4. GUIDELINES FOR OPERATION OF THE TIF CHANNELING SETUP

Power up the microcomputer located on a table near the south wall in the console room. To execute the software for the MCA, type MCA.BAT(RET). At runtime, the batch file MCA.BAT provides command-line switches to the MCA control software. Two MCA cards, manufactured by The Nucleus, are present in the back plane of the PC.

Energize the two stepping motors of the goniometer via the stepping motor controller that is positioned on a shelf in the southeast corner of the shielded target room adjacent to the scattering chamber. The key switch should be positioned in the REMOTE position. Prior to turning the ac power for the controller on, ensure that the electrical connections are

closed between the motors and power supply and the controller and power supply, respectively. Also, check that the RS-232 cable is connected. Enable RS-232 communication with a baud rate of 1200 bits per second (bps) using the proper sequence of keystrokes from the keypad on the front face of the stepping motor controller. The stepping motors should be energized and available for computer control at this stage.

From the "*EXECUTE A BATCH FILE COMMAND*" on the pull-down menu within the MCA program, execute "*C:\path\GON.EXE*" to switch the focus to the goniometer control program. Initiation of the goniometer control program will occur if a communications link is established with the stepping motor controller. Data acquisition launched prior to shelling to the goniometer control program proceeds uninhibited.

The ion beam is aligned and collimated with the assistance of a thin, flat Al_2O_3 (or quartz) specimen mounted at the target position on the goniometer head. The beam spot glows bright blue. Place the beam initially on target by steering with magnetic field adjustments and X-Y steerer variations at the console and by vertical adjustment of the height of the goniometer head. Adjustment of vertical and horizontal pairs of slits (i.e., the "jaws") in the magnet room is facilitated by monitoring target current with the current integrator in a nuclear instrument module (NIM) rack adjacent to the target chamber. A remote meter near the jaws assembly permits beam current to be observed during adjustment. With the beam on target, insert the large aperture on the linear translation feedthrough. Minor adjustment of the scattering chamber position relative to the beam line can place the beam on target center. The beam position on target can be alternatively adjusted with coordinated magnetic steering and movement of jaw pairs. Insert the fixed aperture on the rotary feedthrough, choosing the central hole. The fixed aperture has three openings of varying size that can be selected by a minor rotation. Further adjustment of the individual jaws is now possible for finer collimation of the beam spot and further reduction of the angular divergence. A gold, thin film is present on the mask that mechanically attaches the target to the goniometer head. Backscattered ions from it appear distinctively in the energy spectrum and serve as an aid in positioning.

Secondary electron emission from the target during analysis ion bombardment is suppressed by a 50-V dc battery located on a shelf in the instrument rack in the console room. To extend battery life, disconnect the BNC cable that links the digital current integrator and the physical target during extended idle periods.

From within the MCA shell, an energy spectrum is briefly acquired to indicate the composition of the sample. An energy window can be set on the energy spectrum utilizing an SCA and a calibrated pulser. A range of pulse heights from the spectroscopy amplifier is passed to an external counter and rate meter. Preferably, this range corresponds to scattering from a single bulk element of significant abundance in the sample. Scattering within this range is also from a depth region expected to be crystalline. For alignment purposes, the number of scattering events per unit analysis ion dose (i.e., accumulated charge) collected in this region of interest is observed as a function of angular position. A typical preset ion dose is of the order of 0.1 nC.

5. ADDITIONAL COMMENTS

Instructions for movements of each stepping motor are contained in specific subroutines in the non-volatile memory of the stepping motor controller. Movement distance, sense, and speed can be altered from the keypad of the stepping motor controller or via serial communication. The program section labeled ArbMove can be modified to stand alone for use in modifying individual subroutines.

Embellishments could be made to the code to enhance its usefulness in the TIF. Since the control source code for the MCA written in C is provided by the MCA vendor (i.e., The Nucleus in Oak Ridge, Tennessee), the goniometer control program could be interfaced to automatically generate full angular scans. The procedure would entail positioning the crystal with the goniometer, shelling to a program to control the MCA, counting and electronically recording an energy spectrum, returning to the goniometer control code, repositioning the target, and repeating the process. User judgment would be required to properly adjust the angular position of the target between consecutive energy spectra. Another direction for growth would be to port the application to VisualBasic to create an enhanced user interface.

6. ACKNOWLEDGMENTS

The author would like to thank J. D. Hunn, M. B. Lewis, and L. K. Mansur for their assistance in reviewing this manuscript; M. W. Terrell for secretarial services; G. R. Carter for manuscript preparation; and K. Spence for editing.

APPENDIX A

COMPUTER PROGRAM FOR GONIOMETER CONTROL

```

' $INCLUDE: 'C:\QB45\QBFILES\G.BI
' The included file can be found at the end of this source listing.
DIM w1 AS WindowsType, w5 AS WindowsType, w6 AS WindowsType, w7 AS WindowsType
DIM w1Text$(1 TO 9), w5Text$(1 TO 2), w6Text$(1 TO 13), w7Text$(1 TO 2)
DIM X(25), Y(25)
DIM MoveAngle(25), MasterTxtTheta$(13), MasterTxtPhi$(13)
OPEN "COM1:1200,N,7,2,RS,CS,DS,CD,LF" FOR RANDOM AS #1
' Retrieve the file containing previous theta and phi offsets.
' These are retained on normal program termination for use upon reentry.
OPEN "E:\GONIOM.PAR" FOR RANDOM AS #4
GET #4, 1, OffTheta
GET #4, 2, OffPhi
CLOSE #4
' List of menu items for motion.
' The corresponding movements are stored in the non-volatile memory of the controller.
MasterTxtTheta$(1) = "+0.05 deg."
MasterTxtTheta$(2) = "+0.10 deg."
MasterTxtTheta$(3) = "+0.20 deg."
MasterTxtTheta$(4) = "+0.50 deg."
MasterTxtTheta$(5) = "+1.00 deg."
MasterTxtTheta$(6) = "+5.00 deg."
MasterTxtTheta$(7) = "-0.05 deg."
MasterTxtTheta$(8) = "-0.10 deg."
MasterTxtTheta$(9) = "-0.20 deg."
MasterTxtTheta$(10) = "-0.50 deg."
MasterTxtTheta$(11) = "-1.00 deg."
MasterTxtTheta$(12) = "-5.00 deg."
MasterTxtTheta$(13) = "Change to Phi"
MasterTxtPhi$(1) = " +0.50 deg."
MasterTxtPhi$(2) = " +1.00 deg."
MasterTxtPhi$(3) = " +2.00 deg."
MasterTxtPhi$(4) = " +5.00 deg."
MasterTxtPhi$(5) = "+10.00 deg."
MasterTxtPhi$(6) = "+20.00 deg."
MasterTxtPhi$(7) = " -0.50 deg."
MasterTxtPhi$(8) = " -1.00 deg."
MasterTxtPhi$(9) = " -2.00 deg."
MasterTxtPhi$(10) = " -5.00 deg."
MasterTxtPhi$(11) = "-10.00 deg."
MasterTxtPhi$(12) = "-20.00 deg."
MasterTxtPhi$(13) = "Change to Theta"
' The following sets of statements set parameters for screen creation.
w5.action = -1
w5.bgdEdge = Green: w5.fgdEdge = LightBlue
w5.bgdBody = Green: w5.fgdBody = Black

```

```

w5.fgdTitle = Green: w5.bgdTitle = LightBlue
w5.fgdPrompt = 0: w5.bgdPrompt = 0
w5.fgdHilite = 0: w5.bgdHilite = 0
w5.fgdBorder = Green: w5.bgdBorder = Blue
w5.screenColor = Cyan
w5Title$ = "Position"
w5Prompt$ = ""
w5.displayPage = 0: w5.createPage = 2: w5.overPage = -1
w5.row = 17: w5.column = 37
w1Prompt$ = ""
w1Title$ = "GONIOMETER CONTROL"
w1Text$(1) = "Auto-Alignment"
w1Text$(2) = "Do Arbitrary Move"
w1Text$(3) = "Execute Preset Move"
w1Text$(4) = "Home Position Set"
w1Text$(5) = "Offset Theta/Phi"
w1Text$(6) = "Phi Rock"
w1Text$(7) = "Quit to DOS"
w1Text$(8) = "Return to MCA"
w1Text$(9) = "Scan"
w1.displayPage = 0: w1.createPage = 1: w1.overPage = 2
w1.row = 4: w1.column = 5
w1.action = 1
w1.fgdEdge = LightMagenta: w1.bgdEdge = Blue
w1.fgdBody = White: w1.bgdBody = Blue
w1.fgdTitle = LightRed: w1.bgdTitle = Blue
w1.fgdPrompt = LightGreen: w1.bgdPrompt = Blue
w1.fgdHilite = BrightWhite: w1.bgdHilite = LightCyan
w1.fgdBorder = -1: w1.bgdBorder = 0
w1.screenColor = -1
w6.action = 1
w6.fgdEdge = LightMagenta: w6.bgdEdge = Blue
w6.fgdBody = White: w6.bgdBody = Blue
w6.fgdTitle = LightRed: w6.bgdTitle = Blue
w6.fgdPrompt = LightGreen: w6.bgdPrompt = Blue
w6.fgdHilite = BrightWhite: w6.bgdHilite = LightCyan
w6.fgdBorder = LightBlue: w6.bgdBorder = Blue
w6.screenColor = Cyan
w6Prompt$ = ""
w7.action = -1
w7.fgdEdge = Green: w7.bgdEdge = LightBlue
w7.fgdBody = Green: w7.bgdBody = Black
w7.fgdTitle = Green: w7.bgdTitle = LightBlue
w7.fgdPrompt = 0: w7.bgdPrompt = 0
w7.fgdHilite = 0: w7.bgdHilite = 0
w7.fgdBorder = Green: w7.bgdBorder = Blue
w7.screenColor = Cyan
w7Title$ = ""
w7Prompt$ = ""
w7.displayPage = 7: w7.createPage = 2: w7.overPage = -1
w7.row = 10: w7.column = 47

```

```

w6.displayPage = 7: w6.createPage = 1: w6.overPage = 2
w6.row = 5: w6.column = 10
CLS
GOSUB WhereAreYou
DO
  Theta = CLNG((100 * Theta) + .001) / 100
  Phi = CLNG((100 * Phi) + .001) / 100
  P1$ = LTRIM$(RTRIM$(STR$(Phi)))
  T1$ = LTRIM$(RTRIM$(STR$(Theta)))
  P2$ = LTRIM$(RTRIM$(STR$(OffPhi)))
  T2$ = LTRIM$(RTRIM$(STR$(OffTheta)))
  w5Text$(1) = "Theta is " + T1$ + " deg. Offset(" + T2$ + ")"
  w5Text$(2) = "Phi is " + P1$ + " deg. Offset(" + P2$ + ")"
  MakeWindow w5, w5Text$, w5Title$, w5Prompt$, "C"
  MakeWindow w1, w1Text$, w1Title$, w1Prompt$, "C"
  SCREEN 0, 0, 7, 7
  COLOR BrightWhite, Cyan: CLS
  Border w5.fgdBorder, w5.bgdBorder
  SELECT CASE w1.returnCode
    CASE 1
      GOSUB AutoAlign
    CASE 2
      GOSUB ArbMove
    CASE 3
      GOSUB ExecuteALine
    CASE 4
      GOSUB SetHome
    CASE 5
      GOSUB Offset
    CASE 6
      GOSUB RockIt
    CASE 7
      GOSUB Quit
    CASE 8, -1
      OPEN "E:\GONIOM.PAR" FOR RANDOM AS #4
      PUT #4, 1, OffTheta
      PUT #4, 2, OffPhi
      CLOSE #4
      STOP
    CASE 9
      GOSUB AngScan
  END SELECT
  SELECT CASE w1.returnCode
    CASE 1, 2, 3, 4, 5, 6, 9
      GOSUB WhereAreYou
  END SELECT
LOOP
STOP
' Performs a phi angular scan for alignment purposes.
AutoAlign:
  Border Green, Blue

```

```

COLOR Black, Green
LOCATE 4, 18: PRINT SPACES$(41)
LOCATE 6, 18: PRINT SPACES$(41)
LOCATE 5, 18
PRINT " Automatic Crystal Alignment Procedure "
COLOR BrightWhite, Blue
DO:
  LOCATE 8, 8
  INPUT " Choose the angular interval to scan in degrees (0-359) "; AngInt
  IF AngInt = 0 THEN RETURN
LOOP UNTIL AngInt > 0 AND AngInt < 360
DO:
  LOCATE 10, 8
  INPUT " Choose the scan Direction (+ or -) "; AngDir$
LOOP UNTIL AngDir$ = "+" OR AngDir$ = "-"
DO:
  LOCATE 12, 8
  PRINT " Choose the angular increment (0.5 or 1.0) ";
  INPUT AngInc
LOOP UNTIL AngInc = .5 OR AngInc = 1!
LOCATE 14, 25: COLOR BrightWhite, Red
INPUT " Continue (Y/N) "; GoOn$
IF LEFT$(UCASE$(GoOn$), 1) <> "Y" THEN RETURN
IF AngDir$ = "+" THEN
  IF AngInc = .5 THEN
    SubNum = 13
  ELSEIF AngInc = 1! THEN
    SubNum = 14
  END IF
ELSEIF AngDir$ = "-" THEN
  IF AngInc = .5 THEN
    SubNum = 19
  ELSEIF AngInc = 1! THEN
    SubNum = 20
  END IF
END IF
JMax = INT(AngInt / AngInc)
DummyPhi = Phi
DummyTheta = Theta
SumTotal = 0
COLOR White, Blue
CLS
Border Green, Blue
LOCATE 6, 24
COLOR 31, Green
PRINT " ALIGNMENT IN PROGRESS "
COLOR BrightWhite, Blue
LPRINT "Theta", "Phi", "Counts"
REDIM Counts(400), PhiVal(400), NrmYld(400), Plane(30), PlaneYld(30)
FOR J = 0 TO JMax
  IF J <> 0 THEN

```

```

GOSUB EndRoutine
FOR I = 1 TO 100: NEXT I
PRINT #1, "L"
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
LOCATE 12, 24: PRINT " Moving ..... "; SPACE$(5);
PRINT #1, SubNum; ", "; SubNum
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
IF AngDir$ = "+" THEN
    DummyPhi = DummyPhi + AngInc
ELSEIF AngDir$ = "-" THEN
    DummyPhi = DummyPhi - AngInc
END IF
END IF
LOCATE 10, 24
PRINT " Theta "; DummyTheta; SPACE$(5); "Phi "; DummyPhi; SPACE$(1)
Counter Counts(J)
LPRINT DummyTheta, DummyPhi, Counts(J)
PhiVal(J) = DummyPhi
SumTotal = SumTotal + Counts(J)
NEXT J
RanYld = SumTotal / JMax
LimitVal = .8
LPRINT "Phi", "Norm. Yield"
FOR J = 0 TO JMax
    NrmYld(J) = Counts(J) / RanYld
NEXT J
LPRINT "Plane", "Phi", "Yield"
DO:
    NumP = 1
    FOR J = 0 TO JMax
        IF J <> 0 THEN
            IF NrmYld(J) < LimitVal THEN

                IF NrmYld(J) <= NrmYld(J - 1) AND NrmYld(J) < NrmYld(J + 1) THEN
                    IF ABS(NrmYld(J) - NrmYld(J + 1)) < .05 AND ABS(NrmYld(J) - NrmYld(J -
1))_ < .05 THEN
                        Plane(NumP) = PhiVal(J)
                        PlaneYld(NumP) = NrmYld(J)
                    ELSEIF ABS(NrmYld(J) - NrmYld(J - 1)) < .05 THEN
                        Plane(NumP) = (PhiVal(J) + PhiVal(J - 1)) / 2!
                        PlaneYld(NumP) = (NrmYld(J) + NrmYld(J - 1)) / 2!
                    ELSEIF ABS(NrmYld(J) - NrmYld(J + 1)) < .05 THEN
                        Plane(NumP) = (PhiVal(J) + PhiVal(J + 1)) / 2!
                        PlaneYld(NumP) = (NrmYld(J) + NrmYld(J + 1)) / 2!
                    ELSE
                        Plane(NumP) = PhiVal(J)
                        PlaneYld(NumP) = NrmYld(J)
                END IF
            END IF
        END FOR
    END DO

```

```

        END IF
        LPRINT NumP, Plane(NumP), PlaneYld(NumP)
        NumP = NumP + 1
    END IF
END IF
ELSE
    IF NrmYld(J) < LimitVal THEN
        IF NrmYld(J) < NrmYld(J + 1) THEN
            IF ABS(NrmYld(J) - NrmYld(J + 1)) < .05 THEN
                Plane(NumP) = (PhiVal(J) + PhiVal(J + 1)) / 2!
                PlaneYld(NumP) = (NrmYld(J) + NrmYld(J + 1)) / 2!
            ELSE
                Plane(NumP) = PhiVal(J)
                PlaneYld(NumP) = NrmYld(J)
            END IF
            LPRINT NumP, Plane(NumP), PlaneYld(NumP)
            NumP = NumP + 1
        END IF
    END IF
END IF
NEXT J
LimitVal = LimitVal - .05
LOOP UNTIL NumP MOD 2 <> 0
Mode$ = "V"
NumP = NumP - 1
CALL PolarGraph(Mode$, NumP, DummyTheta, Plane(), PlaneYld())
SCREEN 0, 0, 0, 0
COLOR BrightWhite, Blue
CLS
LOCATE 10, 15
INPUT "Do you want a hardcopy of the alignment diagram"; HardOn$
IF LEFT$(UCASE$(HardOn$), 1) = "Y" THEN
    CALL PolarGraph("E", NumP, DummyTheta, Plane(), PlaneYld())
END IF
SCREEN 0, 0, 0, 0
COLOR BrightWhite, Blue
CLS
ERASE Counts, PhiVal, NrmYld, Plane, PlaneYld
RETURN
' Performs an automated angular scan in either theta or phi.
AngScan:
    Border Green, Blue
    COLOR Black, Green
    LOCATE 4, 18: PRINT SPACE$(38)
    LOCATE 6, 18: PRINT SPACE$(38)
    LOCATE 5, 18
    PRINT "  Automatic Angular Scan Procedure  "
    COLOR BrightWhite, Blue
    LOCATE 8, 8
    INPUT " Choose motion - Theta(T) or Phi(P)"; Motor$
    Motor$ = LEFT$(UCASE$(Motor$), 1)

```

```

DO:
  LOCATE 10, 8
  INPUT " Choose the angular interval to scan in degrees (0-360) "; AngInt
  IF AngInt = 0 THEN RETURN
LOOP UNTIL AngInt > 0 AND AngInt < 361
DO:
  LOCATE 12, 8
  INPUT " Choose the scan Direction (+ or -) "; AngDir$
LOOP UNTIL AngDir$ = "+" OR AngDir$ = "-"
DO:
  LOCATE 14, 8
  IF Motor$ = "T" THEN
    PRINT " Choose the angular increment (0.1, 0.2, or 0.5)"
  ELSE
    PRINT " Choose the angular increment (0.5, 1.0, 2.0 or 5.0)"
  END IF
  LOCATE 16, 20
  INPUT " Increment "; AngInc
  LOOP UNTIL AngInc = .1 OR AngInc = .2 OR AngInc = .5 OR AngInc = 1! OR AngInc =
2! OR AngInc = 5!
  LOCATE 18, 8: COLOR LightGreen, Blue
  INPUT " Continue (Y)"; GoOn$
  COLOR White, Red
  IF LEFT$(UCASE$(GoOn$), 1) <> "Y" THEN RETURN
  IF Motor$ = "T" THEN
    IF AngDir$ = "+" THEN
      IF AngInc = .1 THEN
        SubNum = 2
      ELSEIF AngInc = .2 THEN
        SubNum = 3
      ELSEIF AngInc = .5 THEN
        SubNum = 4
      END IF
    ELSEIF AngDir$ = "-" THEN
      IF AngInc = .1 THEN
        SubNum = 8
      ELSEIF AngInc = .2 THEN
        SubNum = 9
      ELSEIF AngInc = .5 THEN
        SubNum = 10
      END IF
    END IF
  ELSEIF Motor$ = "P" THEN
    IF AngDir$ = "+" THEN
      IF AngInc = .5 THEN
        SubNum = 13
      ELSEIF AngInc = 1! THEN
        SubNum = 14
      ELSEIF AngInc = 2! THEN
        SubNum = 15
      ELSEIF AngInc = 5! THEN

```

```

        SubNum = 16
    END IF
ELSEIF AngDir$ = "-" THEN
    IF AngInc = .5 THEN
        SubNum = 19
    ELSEIF AngInc = 1! THEN
        SubNum = 20
    ELSEIF AngInc = 2! THEN
        SubNum = 21
    ELSEIF AngInc = 5! THEN
        SubNum = 22
    END IF
END IF
ELSE
    RETURN
END IF
JMax = INT(AngInt / AngInc)
DummyPhi = Phi: DummyTheta = Theta
CtsMax = 0: CtsMin = 10000
COLOR White, Blue
CLS
Border Green, Blue
REDIM Counts(400), PhiVal(400), NrmYld(400), Plane(30), PlaneYld(30)
LOCATE 6, 20
COLOR 31, Green
PRINT "  ANGULAR SCAN IN PROGRESS  "
COLOR White, Green
LOCATE 5, 20: PRINT SPACES$(34)
LOCATE 7, 20: PRINT SPACES$(34)
COLOR BrightWhite, Blue
LPRINT "Theta", "Phi", "Counts"
LOCATE 20, 20
PRINT " PRESS Q TO EXIT. "
FOR J = 0 TO JMax
    IF J <> 0 THEN
        GOSUB EndRoutine
        FOR I = 1 TO 100: NEXT I
        PRINT #1, "L"
        FOR I = 1 TO 100: NEXT I
        INPUT #1, Reply$
        FOR I = 1 TO 200: NEXT I
        LOCATE 12, 24: PRINT " Moving ..... "; SPACES$(5);
        PRINT #1, SubNum; ", "; SubNum
        FOR I = 1 TO 100: NEXT I
        INPUT #1, Reply$
        FOR I = 1 TO 200: NEXT I
        IF Motor$ = "P" AND AngDir$ = "+" THEN
            DummyPhi = DummyPhi + AngInc
        ELSEIF Motor$ = "P" AND AngDir$ = "-" THEN
            DummyPhi = DummyPhi - AngInc
        ELSEIF Motor$ = "T" AND AngDir$ = "+" THEN

```

```

        DummyTheta = DummyTheta + AngInc
    ELSEIF Motor$ = "T" AND AngDir$ = "-" THEN
        DummyTheta = DummyTheta - AngInc
    END IF
    IF LEFT$(UCASE$(INKEY$), 1) = "Q" THEN EXIT FOR
END IF
LOCATE 10, 24
PRINT " Theta "; CLNG(((100 * DummyTheta) + .001) / 100; SPACES(5); "Phi ";
Print CLNG(((100 * DummyPhi) + .001) / 100; SPACES(1)
Counter Counts(J)
IF Counts(J) > CtsMax THEN CtsMax = Counts(J)
IF Counts(J) < CtsMin THEN CtsMin = Counts(J)
IF Motor$ = "P" THEN
    PhiVal(J) = DummyPhi
ELSEIF Motor$ = "T" THEN
    PhiVal(J) = DummyTheta
END IF
LPRINT CLNG(((100 * DummyTheta) + .001) / 100, CLNG(((100 * DummyPhi) + .001) /
100, Counts(J)
NEXT J
Mode$ = "V"
CALL LinearPlot(Mode$, PhiVal(), Counts(), CtsMin, CtsMax, JMax)
SCREEN 0, 0, 0, 0
COLOR BrightWhite, Blue
CLS
LOCATE 10, 15
INPUT "Do you want a hardcopy of the alignment diagram"; HardOn$
IF LEFT$(UCASE$(HardOn$), 1) = "Y" THEN
    CALL LinearPlot("E", PhiVal(), Counts(), CtsMin, CtsMax, JMax)
END IF
SCREEN 0, 0, 0, 0
COLOR BrightWhite, Blue
CLS
ERASE Counts, PhiVal, NrmYld, Plane, PlaneYld
RETURN
' Returns the current motor positions
WhereAreYou:
    GOSUB EndRoutine
    FOR I = 1 TO 100: NEXT I
    PRINT #1, "W"
    FOR I = 1 TO 100: NEXT I
    INPUT #1, Reply$
    Theta = VAL(Reply$) / 10000!
    Theta = Theta + OffTheta
    INPUT #1, Reply$
    Phi = VAL(Reply$) / 10000!
    Phi = Phi + OffPhi
    INPUT #1, Reply$
RETURN
' Executes one subroutine stored in the non-volatile memory of the controller
ExecuteALine:

```

```

DummyPhi = Phi
DummyTheta = Theta
LOCATE 12, 35
PRINT " Please Wait "
GOSUB EndRoutine
FOR I = 1 TO 100: NEXT I
PRINT #1, "L"
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 100: NEXT I
InitMotor$ = "T"
LOCATE 12, 35
PRINT SPACES$(13)
DO:
  DO:
    DummyTheta = CLNG((100 * DummyTheta) + .001) / 100
    DummyPhi = CLNG((100 * DummyPhi) + .001) / 100
    w7Text$(1) = "Theta " + LTRIM$(RTRIM$(STR$(DummyTheta))) + " deg."
    w7Text$(2) = "Phi_" + LTRIM$(RTRIM$(STR$(DummyPhi))) + " deg."
    MakeWindow w7, w7Text$(1), w7Title$, w7Prompt$, ""
    FOR I = 1 TO 100: NEXT I
    IF InitMotor$ = "T" THEN
      FOR Kntr = 1 TO 13
        w6Text$(Kntr) = MasterTxtTheta$(Kntr)
      NEXT Kntr
      w6Title$ = "Theta Position"
    ELSEIF InitMotor$ = "P" THEN
      FOR Kntr = 1 TO 13
        w6Text$(Kntr) = MasterTxtPhi$(Kntr)
      NEXT Kntr
      w6Title$ = "Phi Position"
    ELSE
      RETURN
    END IF
    MakeWindow w6, w6Text$(1), w6Title$, w6Prompt$, ""
    IF InitMotor$ = "P" THEN
      OffSub = 12
    ELSE
      OffSub = 0
    END IF
    SELECT CASE w6.returnCode
    CASE 1
      SubNum1 = 1 + OffSub
      IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + .05
      IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + .5
    CASE 2
      SubNum1 = 2 + OffSub
      IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + .1
      IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + 1!
    CASE 3
      SubNum1 = 3 + OffSub

```

```

    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + .2
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + 2!
CASE 4
    SubNum1 = 4 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + .5
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + 5!
CASE 5
    SubNum1 = 5 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + 1!
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + 10!
CASE 6
    SubNum1 = 6 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta + 5!
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi + 20!
CASE 7
    SubNum1 = 7 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - .05
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - .5
CASE 8
    SubNum1 = 8 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - .1
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - 1!
CASE 9
    SubNum1 = 9 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - .2
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - 2!
CASE 10
    SubNum1 = 10 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - .5
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - 5!
CASE 11
    SubNum1 = 11 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - 1!
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - 10!
CASE 12
    SubNum1 = 12 + OffSub
    IF InitMotor$ = "T" THEN DummyTheta = DummyTheta - 5!
    IF InitMotor$ = "P" THEN DummyPhi = DummyPhi - 20!
CASE 13
    IF InitMotor$ = "P" THEN
        InitMotor$ = "T"
    ELSEIF InitMotor$ = "T" THEN
        InitMotor$ = "P"
    END IF
EXIT DO
CASE -1
    RETURN
END SELECT
LOCATE 22, 50
COLOR BrightWhite, Red
PRINT " WORKING ..... "

```

```

SubNum2 = SubNum1
PRINT #1, SubNum1; ","; SubNum2
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
LOCATE 18, 55: COLOR Cyan, Cyan: PRINT Reply$
FOR I = 1 TO 100: NEXT I
LOCATE 22, 50
PRINT SPACE$(15)
LOOP
LOOP
RETURN
' Slowly rock the x-tal in phi while acquiring rotating random spectrum.
RockIt:
Border Green, Blue
LOCATE 5, 5
PRINT " Routine to slowly rock target at 0.25 degrees per second "
LOCATE 6, 5
PRINT " during the acquisition of ROTATING RANDOM spectrum "
LOCATE 8, 5
INPUT " Input angular interval for motion "; ArbPhi
PlusPhi = ArbPhi
MinusPhi = -1 * ArbPhi
CLS
Border Green, Blue
GOSUB EndRoutine
ArbMotion 25, 0, PlusPhi, .1, .25
FOR I = 1 TO 100: NEXT I
GOSUB EndRoutine
ArbMotion 26, 0, MinusPhi, .1, .25
FOR I = 1 TO 100: NEXT I
GOSUB EndRoutine
FOR I = 1 TO 100: NEXT I
PRINT #1, "L"
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 100: NEXT I
DO:
LOCATE 6, 25
COLOR 31, Blue
PRINT " ROCKING TARGET "
PRINT #1, 25; ","; 26
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
LOCATE 6, 25
PRINT SPACE$(30): COLOR White, Green
LOCATE 10, 25: COLOR BrightWhite, Blue
PRINT " Repeat rocking (Y/N) ";
INPUT GoAgain$
LOCATE 10, 25: COLOR 31, Blue
PRINT SPACE$(30)

```

```

    LOOP WHILE LEFT$(UCASE$(GoAgain$), 1) = "Y"
RETURN
' Directs controller command string to input command mode.
EndRoutine:
    PRINT #1, "E": E$ = ""
    FOR I = 1 TO 100: NEXT I
    INPUT #1, E$
    IF E$ = "READY" THEN RETURN
    IF E$ <> "READY" THEN
        FOR I = 1 TO 100: NEXT I
        GOTO EndRoutine
    END IF
RETURN
' Generates an arbitrary theta/phi movement.
ArbMove:
    COLOR Black, Green
    LOCATE 5, 20
    PRINT "  Arbitrary Theta/Phi Movement  "
    COLOR BrightWhite, Blue
    LOCATE 20, 20
    PRINT " THETA "; CLNG((100 * (Theta + OffTheta)) + .001) / 100
    LOCATE 20, 60
    PRINT " PHI "; CLNG((100 * (Phi + OffPhi)) + .001) / 100
    LOCATE 8, 25
    OPEN "E:\OldMove.PAR" FOR RANDOM AS #4
    GET #4, 1, ThetaLast
    GET #4, 2, PhiLast
    CLOSE #4
    INPUT " Input Theta Movement "; ArbTheta
    LOCATE 10, 25
    INPUT " Input Phi Movement "; ArbPhi
    IF ArbTheta = 0 AND ArbPhi = 0 THEN RETURN
    COLOR 31, LightGreen
    LOCATE 13, 28
    PRINT "  MOVING ....  "
    IF ArbTheta <> ThetaLast OR ArbPhi <> PhiLast THEN
        OPEN "E:\OldMove.PAR" FOR RANDOM AS #4
        PUT #4, 1, ArbTheta
        PUT #4, 2, ArbPhi
        CLOSE #4
        GOSUB EndRoutine
        ArbMotion 25, ArbTheta, ArbPhi, 1!, 1.75
    END IF
    GOSUB EndRoutine
    FOR I = 1 TO 100: NEXT I
    PRINT #1, "L"
    FOR I = 1 TO 100: NEXT I
    INPUT #1, Reply$
    FOR I = 1 TO 200: NEXT I
    PRINT #1, 25; ";"; 25
    FOR I = 1 TO 100: NEXT I

```

```

INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
RETURN
' Sets the home position (i.e., theta and phi=0).
SetHome:
  GOSUB EndRoutine
  LOCATE 12, 25: COLOR BrightWhite, Blue
  INPUT " Confirm Set Home (N)"; Confirm$
  IF LEFT$(UCASE$(Confirm$), 1) <> "Y" THEN RETURN
  FOR I = 1 TO 100: NEXT I
  PRINT #1, "S"
  FOR I = 1 TO 100: NEXT I
  INPUT #1, S$
  FOR I = 1 TO 200: NEXT I
RETURN
' An offset can occur since the controller considers theta and phi to
' be zero on power up. This routine permits adjustment for this.
Offset:
  LOCATE 10, 20: COLOR BrightWhite, Blue
  INPUT " THETA offset in degrees"; OffTheta
  LOCATE 12, 20
  INPUT " PHI offset in degrees"; OffPhi
RETURN
' Exit program routine.
Quit:
  GOSUB EndRoutine
  INPUT "Confirm Exit (N)"; Confirm$
  IF LEFT$(UCASE$(Confirm$), 1) <> "Y" THEN RETURN
  PRINT #1, "Q"
  CLOSE #1
  STOP
END
' *****
' ** Name:      ArbMotion          **
' ** Type:     Subroutine          **
' ** Module:   ANAL.BAS           **
' ** Language: MS QuickBASIC Version 4.5 **
' *****
' Permits an arbitrary movement in  $\theta$  and/or  $\phi$ 
' EXAMPLE OF USE: ArbMotion (SubNum%, TAmount, PAmount, TSpeed, PSpeed)
' PARAMETERS:      SubNum%  Subroutine number to program and execute
                   TAmount  Theta movement
                   PAmount  Phi movement
                   TSpeed   Theta motor speed
                   PSpeed   Phi motor speed
' MODULE LEVEL
' DECLARATIONS:  DECLARE SUB ArbMotion (SubNum%, TAmount, PAmount, TSpeed,
PSpeed)
SUB ArbMotion (SubNum%, TAmount, PAmount, TSpeed, PSpeed)
  FOR I = 1 TO 100: NEXT I    'Timing Delay frequently used
  PRINT #1, "P"

```

```

FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
TAmount = TAmount * 10000
PRINT #1, SubNum%, 0, TAmount
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
TSpeed = INT(TSpeed / .0072)
PRINT #1, SubNum%, 1, TSpeed
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
PAmount = PAmount * 10000
PRINT #1, SubNum%, 2, PAmount
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 200: NEXT I
PSpeed = INT(PSpeed / .0072)
PRINT #1, SubNum%, 3, PSpeed
FOR I = 1 TO 100: NEXT I
INPUT #1, Reply$
FOR I = 1 TO 100: NEXT I
END SUB
' *****
' ** Name:      Border                **
' ** Type:      Subroutine             **
' ** Module:    ANAL.BAS               **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Draws a Border about screen perimeter
' EXAMPLE OF USE:  Border Green, Blue
' PARAMETERS:     Clr1%  Foreground Color
                  Clr2%  Background Color
' MODULE LEVEL
' DECLARATIONS:  DECLARE SUB Border (Clr1%, Clr2%)
SUB Border (Clr1%, Clr2%)
  COLOR Clr1%, Clr2%
  Top% = 1: Bottom% = 25
  LOCATE Top%, 1
  PRINT SPACES$(1); CHR$(201);
  FOR I% = 3 TO 78: PRINT CHR$(205); : NEXT I%
  PRINT CHR$(187); SPACES$(1)
  FOR I% = Top% + 1 TO Bottom% - 1
    LOCATE I%, 1
    PRINT SPACES$(1); CHR$(186); SPACES$(1);
    LOCATE I%, 78
    PRINT SPACES$(1); CHR$(186); SPACES$(1);
  NEXT I%
  LOCATE Bottom%, 1
  PRINT SPACES$(1); CHR$(200);

```

```

    FOR I% = 3 TO 78: PRINT CHR$(205); : NEXT I%
    PRINT CHR$(188); SPACES(1);
END SUB
' *****
' ** Name:      Counter          **
' ** Type:     Subroutine        **
' ** Module:    ANAL.BAS         **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Reads the number of counts from a counter
' EXAMPLE OF USE: Counter (Counts)
' PARAMETERS:   Counts Decimal Number of counts.
' MODULE LEVEL
' DECLARATIONS: DECLARE SUB Counter (Counts)
SUB Counter (Counts)
    DIM DataReg%(3), PortValue%(3)
    ConReg% = &H228
    OUT ConReg%, &H8
    DataReg%(0) = &H229
    DataReg%(1) = &H22A
    DataReg%(2) = &H22B
    DataReg%(3) = &H22C
    ' Port 3 - - - - - > Bit 1 = Strobe
    '                   Bit 2 = Preset
    '                   Bit 3 = Reset
    '                   H0 Zeros
    '                   H1 Halts and Zeros
    '                   H4 Halts and Reads
    '                   H5 Halts Without Read
    '                   H7 Starts and Restarts Without Zeroing or Reading
    OUT DataReg%(3), &H7
    LOCATE 12, 24: PRINT " Counting ..... ";
    DO:
        PortValue%(2) = INP(DataReg%(2))
        Value$ = MID$(HEX$(PortValue%(2)), 1, 1)
    LOOP UNTIL Value$ = "7"
    OUT DataReg%(3), &H4
    FOR I = 0 TO 2
        PortValue%(I) = INP(DataReg%(I))
    NEXT I
    IF MID$(HEX$(PortValue%(2)), 1, 1) < "8" THEN 'Corrects for Pin 47 Being
        PortValue%(2) = PortValue%(2) + 128 'Connected to Interval
    END IF
' Read Counter for Preset
    FOR I = 2 TO 0 STEP -1
        FOR J = 1 TO 2
            Value$ = MID$(HEX$(PortValue%(I)), J, 1)
            IF Value$ = "A" THEN Value$ = "10"
            IF Value$ = "B" THEN Value$ = "11"
            IF Value$ = "C" THEN Value$ = "12"
            IF Value$ = "D" THEN Value$ = "13"

```

```

        IF Value$ = "E" THEN Value$ = "14"
        IF Value$ = "F" THEN Value$ = "15"
        Digit$ = Digit$ + STR$(15 - VAL(Value$))
    NEXT J
NEXT I
Counts = VAL(Digit$)
OUT DataReg%(3), &H0
LOCATE 12, 24
PRINT " Recorded Counts "; Counts; SPACES$(10);
END SUB
' *****
' ** Name:      InKeyCode%                **
' ** Type:      Function                   **
' ** Module:    ANAL.BAS                   **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Returns a unique integer for any key pressed or
' a zero if no key was pressed.
'
' EXAMPLE OF USE:  k%=InKeyCode%
' PARAMETERS:     (none)
' VARIABLES:      (none)
' MODULE LEVEL
' DECLARATIONS:  DECLARE FUNCTION InKeyCode% ()
FUNCTION InKeyCode% STATIC
    InKeyCode% = CVI(INKEY$ + STRING$(2, 0))
END FUNCTION
' *****
' ** Name:      LinearPlot                 **
' ** Type:      Subroutine                  **
' ** Module:    ANAL.BAS                   **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Draws a linear plot of angular scan data
' EXAMPLE OF USE:  LinearPlot (Mode$, X(), Y(), CtsMin, CtsMax, JMax)
' PARAMETERS:     Mode$
                  X()
                  Y()
                  CtsMin
                  CtsMax
                  JMax
' MODULE LEVEL
' DECLARATIONS:  DECLARE SUB LinearPlot (Mode$, X(), Y(), CtsMin, CtsMax, JMax)
SUB LinearPlot (Mode$, X(), Y(), CtsMin, CtsMax, JMax)
    DIM KEYCHR AS STRING * 1
    DIM K AS INTEGER
    IF Mode$ = "V" THEN
        CALL GOPEN("VGA2\0", "CON:\0", K)
    ELSEIF Mode$ = "E" THEN
        CALL GOPEN("EPS1\0", "LPT1:\0", K)
    ELSE

```

```

EXIT SUB
END IF
UpLim = 10 ^ (INT(LOG(CtsMax) / LOG(10!)) + 1)
SELECT CASE ABS((X(JMax) - X(0)))
CASE 1 TO 20
  XTSpace = 1: XTicks% = 2
CASE 21 TO 100
  XTSpace = 2: XTicks% = 5
CASE 101 TO 360
  XTSpace = 5: XTicks% = 10
END SELECT
SELECT CASE (UpLim)
CASE 1 TO 10
  YTSpace = 1: YTicks% = 2
CASE 11 TO 100
  YTSpace = 5: YTicks% = 5
CASE 101 TO 1000
  YTSpace = 50: YTicks% = 5
END SELECT
IF (K > 0) THEN
CALL NEWWND(.5, .5, 8!, 7!, 2)
IF Mode$ = "V" THEN
  CALL FILLWND(1)
  CALL NEWPEN(7)
ELSEIF Mode$ = "E" THEN
  CALL NEWPEN(1)
END IF
CALL WFRAME
CALL PLIMIT(1!, 1!, 6!, 5!, 2)
CALL CLIPON
CALL SETMET
CALL SCALE(X(0), X(JMax), 0, UpLim)
CALL XTSIZE(.35, .35, .2, .2)
CALL YTSIZE(.35, .35, .2, .2)
CALL MSIZE(.1)
CALL XAXIS(X(0), X(JMax), 0, XTSpace, XTicks%)
CALL YAXIS(0, UpLim, X(0), YTSpace, YTicks%)
CALL XAXIS(X(0), X(JMax), UpLim, XTSpace, XTicks%)
CALL YAXIS(0, UpLim, X(JMax), YTSpace, YTicks%)
FOR J = 0 TO JMax STEP (XTSpace * XTicks%)
  CALL MOVE(X(J), 0)
  SELECT CASE ABS(X(J))
CASE 0 TO 9
  CALL CMOVE(-1!, -2!)
CASE 10 TO 99
  CALL CMOVE(-2!, -2!)
CASE 100 TO 999
  CALL CMOVE(-3!, -2!)
END SELECT
  CALL LABEL(STR$(X(J)) + "\0")
NEXT J

```

```

CALL CLIPOFF
FOR J = 0 TO UpLim STEP (YTSpace * YTicks%)
  CALL MOVE(X(0), J)
  SELECT CASE J
  CASE 0 TO 9
    CALL CMOVE(-3!, 0!)
  CASE 10 TO 99
    CALL CMOVE(-4!, 0!)
  CASE 100 TO 999
    CALL CMOVE(-5!, 0!)
  CASE 1000 TO 9999
    CALL CMOVE(-6!, 0!)
  END SELECT
  CALL LABEL(STR$(J) + "\0")
NEXT J
' CALL CLIPOFF
FOR J = 0 TO JMax
  CALL MOVE(X(J), Y(J))
  CALL SETMARK(1)
  CALL MARK
NEXT J
CALL GCLOSE
DO: LOOP UNTIL LEN(INKEY$)
CALL SETTXT
ELSE
  PRINT " *** DEVICE SETUP ERROR ****"
END IF
END SUB
' *****
' ** Name:      MakeWindow      **
' ** Type:      Subprogram      **
' ** Module:    ANAL.BAS        **
' ** Language:  MS QuickBASIC Version 4.5  **
' *****
' Creates a window containing a menu on the screen.
' PARAMETERS:  wText$( ), Row, Center$, Column%
'              wText$(I%)      Labels to print
'              Row%            Row on screen to place upper left hand corner of centered
'                               window
'              Center$         Center box left-to-right ("C" = yes)
'              Column%         Column number on screen to place upper left hand corner
of '                             centered window
' VARIABLES:
'   action -1 Create "no input" window and return
'           0 Create "no input" window, display and return
'           1 Create "active input" window, display and return
'   returncode choice selected (line number of wText$ or -1 for ESC)
'   row        row of display of upper left edge
'   column     column of display of upper left edge
'   displaypage -1 for no display on this screen page
'   createpage screen page to create window image on

```

```

'      overpage      page to overlay if multiple windows to be displayed
' MODULE LEVEL
' DECLARATIONS:  DECLARE SUB MakeWindow (w AS WindowsType, wText$, Title$, '
wPrompt$, Center$)
SUB MakeWindow (w AS WindowsType, wText$, wTitle$, wPrompt$, Center$) STATIC
  'Key code numbers for cursor control
  CONST DOWNARROW = 20480
  CONST ENTER = 13
  CONST ESCAPE = 27
  CONST UPARROW = 18432
  CONST ENDKEY = 20224
  CONST HOMEKey = 18176
  CONST PGUP = 18688
  CONST PGDN = 20736
  CONST SPACEBAR = 32
  cursorRow% = CSRLIN
  cursorCol% = POS(0)
  SCREEN 0, 0, w.createPage, w.displayPage
  IF w.screenColor > -1 THEN
    COLOR White, w.screenColor: CLS
  END IF
  IF w.fgdBorder <> -1 THEN
    Border w.fgdBorder, w.bgdBorder
  END IF
  IF w.overPage > 0 AND w.overPage < 7 THEN
    PCOPY w.overPage, w.createPage
  END IF
  NumStrings% = UBOUND(wText$)
  AMaxLen% = LEN(wText$(1))
  FOR I% = 2 TO NumStrings%
    IF AMaxLen% < LEN(wText$(I%)) THEN AMaxLen% = LEN(wText$(I%))
  NEXT I%
  IF UCASE$(Center$) = "C" THEN
    Col% = 37 - INT((AMaxLen% / 2))
  ELSE
    Col% = w.column
  END IF
  LOCATE w.row, Col%
  COLOR w.fgdEdge, w.bgdEdge
  PRINT SPACE$(1); CHR$(201); STRING$(AMaxLen% + 4, 205); CHR$(187); SPACE$(1)
  FOR I% = 1 TO NumStrings%
    LOCATE , Col%
    COLOR w.fgdEdge, w.bgdEdge
    FOR K = 1 TO 10: NEXT K
    PRINT SPACE$(1); CHR$(186); SPACE$(1);
    COLOR w.fgdBody, w.bgdBody
    PRINT SPC(1); wText$(I%); SPACE$(1 + AMaxLen% - LEN(wText$(I%)));
    COLOR w.fgdEdge, w.bgdEdge
    PRINT SPACE$(1); CHR$(186); SPACE$(1);
    COLOR Blue, Black: PRINT SPACE$(2)
  NEXT I%

```

```

LOCATE , Col%
COLOR w.fgdEdge, w.bgdEdge
  PRINT SPACES(1); CHR$(200); STRING$(AMaxLen% + 4, 205); CHR$(188) + SPACES(1);
COLOR Blue, Black: PRINT SPACES(2)
LOCATE , Col% + 2
COLOR Blue, Black: PRINT SPACES(AMaxLen% + 8);
IF w.action = 1 THEN
  FOR I% = 1 TO NumStrings%
    LOCATE w.row + I%, Col% + 4
    COLOR Yellow, w.bgdBody
    PRINT LEFT$(wText$(I%), 1)
  NEXT I%
END IF
IF wTitle$ <> "" THEN
  colTitle% = 3 + Col% + (AMaxLen% / 2) - (LEN(wTitle$) / 2)
  LOCATE w.row, colTitle%
  COLOR w.fgdTitle, w.bgdTitle
  PRINT SPACES(1) + wTitle$ + SPACES(1);
END IF
IF wPrompt$ <> "" THEN
  rowPrompt% = w.row + 1 + NumStrings%
  colPrompt% = Col% + 3 + (AMaxLen% / 2) - (LEN(wPrompt$) / 2)
  LOCATE rowPrompt%, colPrompt%
  COLOR w.fgdPrompt, w.bgdPrompt
  PRINT SPACES(1) + wPrompt$ + SPACES(1);
END IF
IF w.action = -1 THEN EXIT SUB
PCOPY w.createPage, w.displayPage
SCREEN 0, 0, w.displayPage, w.displayPage
IF w.action = 0 THEN EXIT SUB
IF w.returnValue > 1 AND w.returnValue < NumStrings% + 1 THEN
  ptr% = w.returnValue
ELSE
  ptr% = 1
END IF
IF ptr% = 1 THEN
  lastPtr% = 2
ELSE
  lastPtr% = ptr% - 1
END IF
choice$ = ""
FOR I% = 1 TO NumStrings%
  tmp$ = UCASE$(LTRIM$(wText$(I%)))
  DO
    IF tmp$ <> "" THEN
      t$ = LEFT$(tmp$, 1)
      tmp$ = MID$(tmp$, 2)
      IF INSTR(choice$, t$) = 0 THEN
        choice$ = choice$ + t$
      END IF
    END IF
  END IF
END IF

```

```

LOOP UNTIL LEN(choice$) = I%
NEXT I%
DO
  kee% = InKeyCode%
  IF kee% >= ASC("a") AND kee% <= ASC("z") THEN
    kee% = ASC(UCASE$(CHR$(kee%)))
  END IF
  SELECT CASE kee%
  CASE UPARROW
    IF ptr% > 1 THEN
      ptr% = ptr% - 1
    ELSE
      ptr% = NumStrings%
    END IF
  CASE DOWNARROW
    IF ptr% < NumStrings% THEN
      ptr% = ptr% + 1
    ELSE
      ptr% = 1
    END IF
  CASE HOMEKey, PGUP
    ptr% = 1
  CASE ENDKEY, PGDN
    ptr% = NumStrings%
  CASE ENTER, SPACEBAR
    w.returnValue = ptr%
  CASE ESCAPE
    w.returnValue = -1
  CASE ELSE
    IF kee% < 256 THEN w.returnValue = INSTR(choice$, CHR$(kee%))
    IF w.returnValue THEN
      ptr% = w.returnValue
    END IF
  END SELECT
  IF ptr% <> lastPtr% THEN
    LOCATE lastPtr% + w.row, Col% + 4, 0
    COLOR w.fgdBody, w.bgdBdy
    tmp$ = LEFT$(wText$(lastPtr%) + SPACES$(AMaxLen%), AMaxLen%)
    PRINT tmp$;
    LOCATE ptr% + w.row, Col% + 4, 0
    COLOR w.fgdHilite, w.bgdHilite
    tmp$ = LEFT$(wText$(ptr%) + SPACES$(AMaxLen%), AMaxLen%)
    PRINT tmp$;
    COLOR Yellow, w.bgdBdy
    LOCATE lastPtr% + w.row, Col% + 4, 0
    PRINT LEFT$(wText$(lastPtr%), 1)
    COLOR Yellow, w.bgdHilite
    LOCATE ptr% + w.row, Col% + 4, 0
    PRINT LEFT$(wText$(ptr%), 1)
    lastPtr% = ptr%
  END IF

```

```

    LOOP WHILE w.returnCode = 0
    LOCATE cursorRow%, cursorCol%
END SUB
' *****
' ** Name:      PolarGraph      **
' ** Type:      Subroutine      **
' ** Module:    ANAL.BAS        **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Draws a polar plot of angular scan data
' EXAMPLE OF USE: PolarGraph (Mode$, NPts, Th, X(), Y())
' PARAMETERS:   Mode$
                X()
                Y()
                CtsMin
                CtsMax
                JMax
' MODULE LEVEL
' DECLARATIONS: DECLARE SUB PolarGraph (Mode$, NPts, Th, X(), Y())
SUB PolarGraph (Mode$, NPts, Th, X(), Y())
    DIM KEYCHR AS STRING * 1
    DIM K AS INTEGER
    IF Mode$ = "V" THEN
        CALL GOPEN("VGA2\0", "CON:\0", K)
    ELSEIF Mode$ = "E" THEN
        CALL GOPEN("EPS1\0", "LPT1:\0", K)
    ELSE
        EXIT SUB
    END IF
    Th = Th / 8!
    IF (K > 0) THEN
        CALL NEWWND(1.25, .1, 7.25, 7.25, 2)
        IF Mode$ = "V" THEN
            CALL FILLWND(1)
            CALL NEWPEN(7)
        ELSEIF Mode$ = "E" THEN
            CALL NEWPEN(1)
        END IF
        CALL WFRAME
        CALL HGLTYPE(0)
        CALL VGLTYPE(0)
        IF Mode$ = "V" THEN
            CALL PLGRID(3!)
            CALL NEWPEN(13)
        ELSEIF Mode$ = "E" THEN
            CALL PRGRID(3!)
            CALL NEWPEN(1)
        END IF
        CALL CSIZE(.35, .35)
        CALL MSIZE(.4)
        FOR I = 1 TO NPts

```

```

        CALL MOVE(Th, X(I))
        CALL SETMARK(4)
        CALL MARK
        CALL SETMARK(1)
        CALL MARK
    NEXT I
    IF Mode$ = "V" THEN
        CALL NEWPEN(10)
    END IF
    FOR J = 1 TO NPts
        CALL MOVE(Th, X(J))
        CALL NLABEL(Y(J), 2)
    NEXT J
    CALL SETSU
    FOR J = 1 TO (NPts / 2)
        CALL MOVE(Th, X(J))
        CALL GLINE(Th, X(J + (NPts / 2)))
    NEXT J
    CALL GCLOSE
    CALL RDKEY(KEYCHR, K)
    CALL SETTXT
    SCREEN 0, 0, 7, 7
    COLOR BrightWhite, Cyan: CLS
ELSE
    PRINT " *** DEVICE SETUP ERROR ****"
END IF
END SUB
' *****
' ** Name:      G.BI                **
' ** Type:      Include File        **
' ** Module:    ANAL.BAS            **
' ** Language:  MS QuickBASIC Version 4.5 **
' *****
' Included File of constants and declarations
' EXAMPLE OF USE: ' $INCLUDE: 'C:\QB45\qbfiles\G.BI'
CONST Black = 0
CONST Blue = 1
CONST Green = 2
CONST Cyan = 3
CONST Red = 4
CONST Magenta = 5
CONST Brown = 6
CONST White = 7
CONST Grey = 8
CONST LightBlue = 9
CONST LightGreen = 10
CONST LightCyan = 11
CONST LightRed = 12
CONST LightMagenta = 13
CONST Yellow = 14
CONST BrightWhite = 15

```

TYPE WindowsType

```

action AS INTEGER
returnCode AS INTEGER
row AS INTEGER
column AS INTEGER
bgdEdge AS INTEGER
fgdEdge AS INTEGER
fgdBody AS INTEGER
bgdBody AS INTEGER
fgdTitle AS INTEGER
bgdTitle AS INTEGER
fgdPrompt AS INTEGER
bgdPrompt AS INTEGER
fgdHilite AS INTEGER
bgdHilite AS INTEGER
fgdBorder AS INTEGER
bgdBorder AS INTEGER
screenColor AS INTEGER
displayPage AS INTEGER
createPage AS INTEGER
overPage AS INTEGER

```

END TYPE

TYPE Allen

```

Phi AS LONG
Theta AS LONG
DummyPhi AS LONG
DummyTheta AS LONG

```

END TYPE

' Subroutines outside of library that are part of main code

DECLARE SUB ArbMotion (SubNum%, TAmount, PAmount, TSpeed, PSpeed)

DECLARE SUB Border (Clr1%, Clr2%)

DECLARE SUB Counter (Counts)

DECLARE SUB PolarGraph (Mode\$, NumP, Theta, X(), Y())

DECLARE SUB MakeWindow (w AS WindowsType, wText\$, wTitle\$, wPrompt\$, Center\$)

DECLARE SUB LinearPlot (Mode\$, X(), Y(), CtsMin, CtsMax, JMax)

DECLARE FUNCTION InKeyCode% ()

' SubRoutine declarations for INGRAF Library

DECLARE SUB CLIPOFF ()

DECLARE SUB CLIPON ()

DECLARE SUB CMOVE (XCHAR AS SINGLE, YCHAR AS SINGLE)

DECLARE SUB CSIZE (CW AS SINGLE, CH AS SINGLE)

DECLARE SUB FILLWND (ICOLOR AS INTEGER)

DECLARE SUB GCLOSE ()

DECLARE SUB GLINE (XPOINT AS SINGLE, YPOINT AS SINGLE)

DECLARE SUB GMOVE (XLENTH AS SINGLE, YLENTH AS SINGLE, IUNITS AS INTEGER)

DECLARE SUB GOPEN (GDNAME AS STRING, PFILE AS STRING, ISTAT AS INTEGER)

DECLARE SUB GRID (XGSPC AS SINGLE, YGSPC AS SINGLE, XTSPC AS SINGLE, YTSPC AS SINGLE, XMAJCT AS INTEGER, YMAJCT AS INTEGER)

DECLARE SUB HGLTYPE (HLCODE AS INTEGER)

DECLARE SUB LABEL (LBLSTR AS STRING)

```
DECLARE SUB LBLDIR (DANGLE AS SINGLE)
DECLARE SUB LBLORG (ORGCOD AS INTEGER)
DECLARE SUB LTYPE (LCODE AS INTEGER)
DECLARE SUB MARK ()
DECLARE SUB MOVE (XVAL AS SINGLE, YVAL AS SINGLE)
DECLARE SUB MSIZE (MRKSIZ AS SINGLE)
DECLARE SUB NEWPEN (PENNO AS INTEGER)
DECLARE SUB NEWWND (WX1 AS SINGLE, WY1 AS SINGLE, XLENTH AS SINGLE,
YLENTH_ AS SINGLE, IUNITS AS INTEGER)
DECLARE SUB NLABEL (FP AS SINGLE, NFRAC AS INTEGER)
DECLARE SUB PGRID (RADVAL AS SINGLE)
DECLARE SUB PLGRID (RADIUS AS SINGLE)
DECLARE SUB PLIMIT (XREF AS SINGLE, YREF AS SINGLE, XLENTH AS SINGLE,
YLENTH_ AS SINGLE, IUNITS AS INTEGER)
DECLARE SUB PRGRID (RADVAL AS SINGLE)
DECLARE SUB RDKEY (KEYCHAR AS STRING, KEYCOD AS INTEGER)
DECLARE SUB SETGU ()
DECLARE SUB SETMET ()
DECLARE SUB SETMARK (MARKER AS INTEGER)
DECLARE SUB SETSU ()
DECLARE SUB SETTXT ()
DECLARE SUB VGLTYPE (VLCODE AS INTEGER)
DECLARE SUB WFRAME ()
DECLARE SUB XAXIS (XMIN AS SINGLE, XMAX AS SINGLE, YINT AS SINGLE, TICSPC
AS_SINGLE, MAJCT AS INTEGER)
DECLARE SUB XGRID (XGSPC AS SINGLE, XTSPC AS SINGLE, MAJCT AS INTEGER)
DECLARE SUB XTSIZE (PMAJ AS SINGLE, NMAJ AS SINGLE, PMIN AS SINGLE, NMIN
AS_SINGLE)
DECLARE SUB YAXIS (YMIN AS SINGLE, YMAX AS SINGLE, XINT AS SINGLE, TICSPC
AS_SINGLE, MAJCT AS INTEGER)
DECLARE SUB YGRID (YGSPC AS SINGLE, YTSPC AS SINGLE, MAJCT AS INTEGER)
DECLARE SUB YTSIZE (PMAJ AS SINGLE, NMAJ AS SINGLE, PMIN AS SINGLE, NMIN
AS_SINGLE)
```

INTERNAL DISTRIBUTION

- | | |
|------------------------------------|----------------------------------|
| 1-2. Central Research Library | 20. M. B. Lewis |
| 3. Document Reference Section | 21. G. M. Ludtka |
| 4-5. Laboratory Records Department | 22. L. K. Mansur |
| 6. Laboratory Records, ORNL RC | 23. R. M. Moon |
| 7. ORNL Patent Section | 24. E. D. Specht |
| 8-10. M&C Records Office | 25. L. J. Turner |
| 11-15. W. R. Allen | 26. H. W. Foglesong (Consultant) |
| 16. B. R. Appleton | 27. E. L. Menger (Consultant) |
| 17. C. A. Baldwin | 28. J. G. Simon (Consultant) |
| 18. L. L. Horton | 29. K. E. Spear (Consultant) |
| 19. K. J. Kozaczek | |

EXTERNAL DISTRIBUTION

30. DOE, OAK RIDGE OPERATIONS OFFICE, P.O. Box 2001, Oak Ridge,
TN 37831-8600
- Assistant Manager for Energy Research and Development
- 31-32. DOE, OFFICE OF SCIENTIFIC AND TECHNICAL INFORMATION, Office of
Information Services, P.O. Box 62, Oak Ridge, TN 37831
- For distribution by microfiche as shown in DOE/OSTI-4500
Distribution Category UC-25 (Materials)