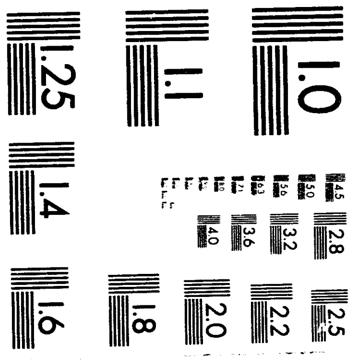

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

1 of 1

2

LA-SUB--94-39

THE LOS ALAMOS NATIONAL LABORATORY PRECISION DOUBLE CRYSTAL SPECTROMETER

by

Dane V. Morgan

Carlos J. Stevens

Robert J. Liefeld

OUTLINE

LIST OF FIGURES

INTRODUCTION

Motivation for the Construction of the Instrument

X-ray spectroscopy is an important tool for the experimental study of atomic physics and materials science, and a fully automated vacuum double crystal spectrometer is a versatile instrument for research in these fields. Desirable features of the instrument include the total automation of the x-ray spectrometer positioning and data acquisition. This saves many man-hours compared to labor-intensive manual x-ray spectrometers, and eliminates the possibility of human error. Another desirable feature of the instrument is the ability to observe the entire x-ray spectrum, including soft and ultra-soft x-rays, so the instrument must be equipped with a high-vacuum capability. Accuracy and resolution are the most important characteristics that we wish to incorporate into the instrument. The double crystal spectrometer is the best instrument for this type of work because of our ability to measure rotational positions of the crystals with great precision, and because it has an inherently low level of scattered background intensity.

X-ray spectroscopists at the Los Alamos National Laboratory (LANL) have stated that the theories of atomic structure and material properties have advanced to the point where existing measurements of atomic emission line wavelengths are inadequate, so there is a need for an instrument which can more accurately measure these wavelengths. Ultra-pure and perfect silicon crystals are now available which can measure wavelengths accurately to approximately one part-per-million. The wavelength accuracy can be converted to the spectrometer's angular accuracy using Bragg's law,

$$n\lambda = 2d(\sin\theta),$$

and

$$d\lambda/\lambda = (\cot\theta)d\theta.$$

A wavelength accuracy of one part-per-million therefore corresponds to an angular accuracy of 5×10^{-7} radians (or 0.1 arc seconds) at 26.5°. By measuring standard wavelengths, such as molybdenum $K\alpha$, with a precision double crystal spectrometer equipped with silicon crystals, the d-spacing of other crystals can be precisely determined by measuring the Bragg angle of the reflection of the standard wavelength. By extending this process, it would be possible to accurately determine the x-ray emission wavelengths and absorption edges of all the elements in the periodic chart.

Not all crystals are necessarily suitable for use in precision spectroscopy, and for those crystals that are suitable, the accuracy of the instrument is limited by the purity, quality, and other properties of the crystal. The precision double crystal spectrometer can characterize crystal properties by measurement of the crystal diffraction width in the parallel (rocking curve) position. Rocking curve widths fall in the range of 0.1 to 100 arc seconds, full-width-at-half-maximum (FWHM), with the narrow rocking curve widths occurring for the x-ray energies of the K spectra of heavy atoms. It is therefore desirable to have an instrument with a resolution on the order of 0.01 arc seconds to accurately plot high energy rocking curve profiles.

There are many difficulties in the design and construction of an instrument with an angular accuracy of 0.1 sec. and an angular resolution of 0.01 sec. Thermal gradients and vibrations can have a profound effect on the ability to achieve these specifications. Moore Special Tool Company was first contracted to design and build the instrument because of their experience and reputation with precision metrology.

A Brief History of the Instrument

The initial design contract for the Precision Double Crystal Spectrometer (PDCS) was awarded to Moore Special Tool Co. (MST) in April 1986, based on LANL P-14 PDCS Specifications. Shortly thereafter, MST began constructing the PDCS. In July 1989, MST terminated work on the PDCS, and shipped the unfinished subassemblies to NSLS. The PDCS was then shipped to LANL for evaluation by MEC-6 division.

In the summer of 1990, the N.M.S.U. Physics Department was contracted by LANL to render the PDCS operational, and all the unfinished subassemblies were shipped to Las Cruces, NM. An initial assessment was made to determine which subassemblies could be completed locally, and which subassemblies would be completed by subcontractors. The decision was made to have MST complete the precision bearings for the ΘA, ΘB, and 2ΘA axes, and to design and build the 120 t.p.i. micrometer lead screw and fine motion drive for the 2ΘA axis. All other work would be accomplished locally, including design and construction of the 2ΘB spindle and detector arm, inchworm mounts, 2ΘA lift-lock, and crystal holders. All other major subassemblies were essentially complete.

The completed subassemblies from MST were received in August 1991. At about the same time, a Manson proportional counter was received from Los Alamos for use as the detector. At N.M.S.U., a sustained effort was made to develop the software required to operate the inchworm motors, Compumotors, and data acquisition. The subassemblies built in Las Cruces were completed, and the final assembly of the PDCS was begun.

The alignment of the instrument was performed as the instrument was assembled. The ΘA axis was aligned for coaxiality and parallelism with respect to the 2ΘA axis, and the ΘA and ΘB axes were made parallel. The two quartz crystals were aligned so their optical faces were parallel and coaxial to their respective axes. The alignment procedure will be described in greater detail later in the report.

The inchworm motors and controller proved to be a major obstacle to the construction of the instrument. The inchworm system was received with a faulty Heidenhain EXE encoder electronics box, a bad inchworm motor, and faulty cabling. These problems were corrected, one by one, and in June 1992, the construction of the PDCS was complete, and testing of the operational programs was begun.

In order to test the PDCS, a copper x-ray source was attached to the PDCS vacuum chamber. The copper $K\alpha$ and $K\beta$ emission spectrum was observed and recorded as the PDCS scanned in the (1,1) dispersive mode. The PDCS was then placed in the (1,-1) position, and a rocking curve at the copper $K\alpha$ wavelength was recorded. The spectrometer motion and data acquisition were fully automated during these tests, and the only operator action required was to enter the initial scan parameters.

After initial x-ray testing of the PDCS, work began on the vacuum system. The chamber was pumped down to 10^{-2} torr by a mechanical forepump, and checked with a helium leak detector. A turbopump, controller, and a special elbow were acquired from LANL. After evacuating the vacuum chamber to 10^{-2} torr, the turbopump further reduced the ultimate pressure in the vacuum chamber to 2×10^{-6} torr, as measured by an ionization gauge.

INSTRUMENT DESCRIPTION

Mechanical Systems

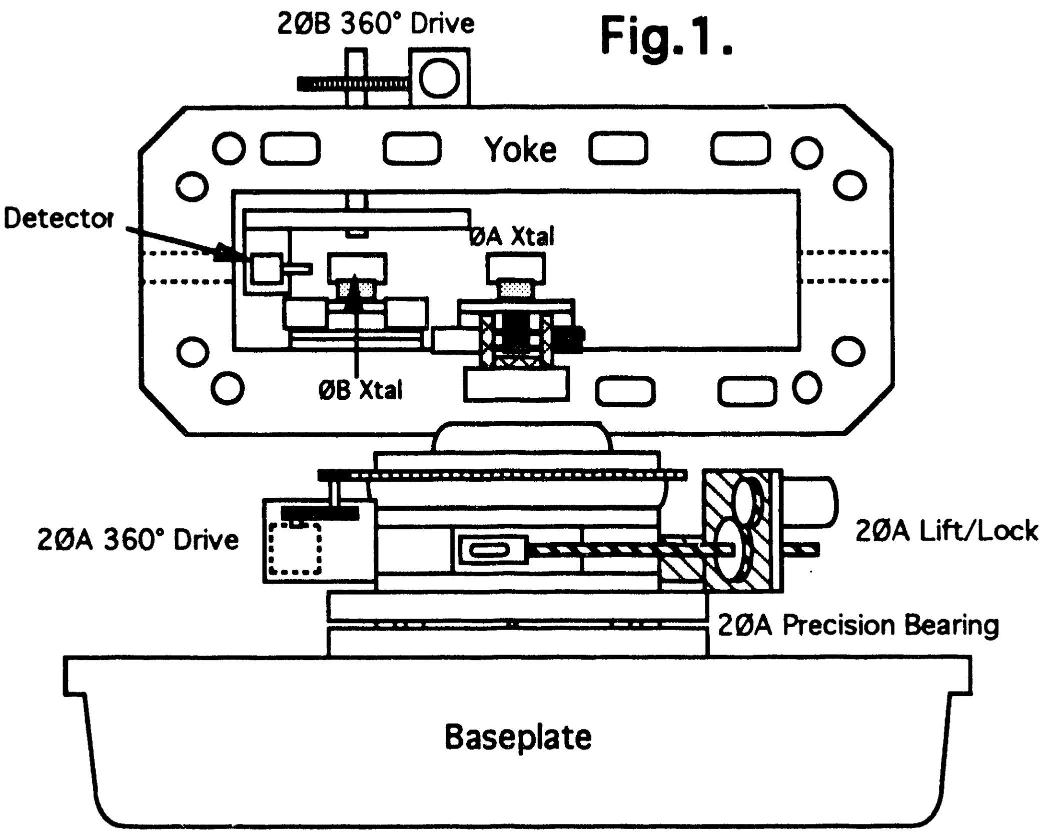
Overall Mechanical Description of the Instrument

The PDCS, shown in Fig. 1, is supported by a cast iron base plate designed for minimum deformation in a high vacuum environment. The base plate supports both the spectrometer and the vacuum chamber. It was originally intended for the spectrometer to be supported by a Barry air support system to dampen mechanical vibrations, but this air support system is not presently being used because of the lack of clearance between the hoist and the top of the vacuum chamber in Gardiner Hall room 60, where the spectrometer presently resides.

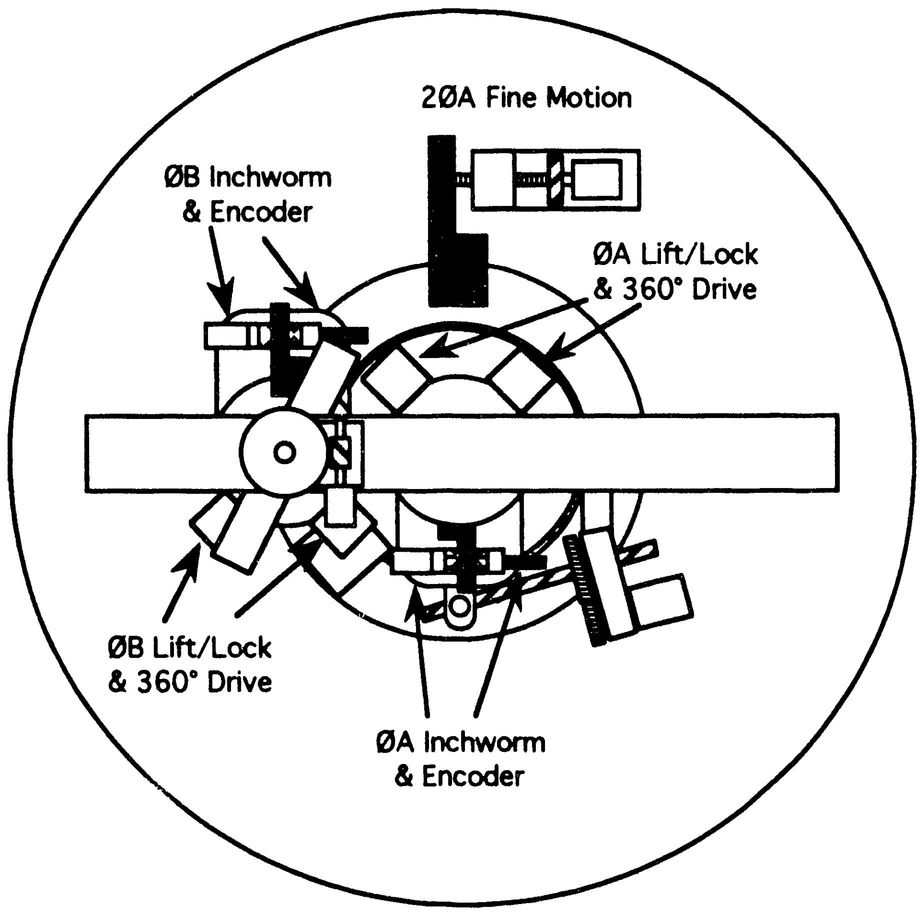
The 2ΘA axis rests on the center of the base plate, and supports the rest of the instrument. The 2ΘA axis is comprised of two rotational subassemblies: an Ultradex 720 indexing table provided by A. A. Gage, and a precision bearing fabricated by MST, which serves as a fine angle divider. The indexing table provides a 360° rotational capability in discrete increments of 0.5°, and each of these 720 positions is accurate to ±0.125 arc seconds. A lift/lock mechanism and a 360° rotational drive are used to move the indexing table to the desired position. The indexing table rides on top of the precision bearing, which provides fine motion capability for the 2ΘA axis. A tangent arm is mounted on the upper bearing plate of the precision bearing. This tangent arm is driven by a 120 t. p. i. lead screw, which is, in turn, driven by a stepper motor, providing fine motion of the 2ΘA axis in increments of 0.1 arc seconds over a 0.5° range.

The yoke is mounted to the top of the Ultradex 720 indexing table, and is centered on the table with a guide pin. The purpose of the yoke is to provide a rigid support for the ΘA and ΘB crystal axes, as well as the 2ΘB detector axis. The ΘA and ΘB axes are mounted on the lower horizontal portion of the yoke, and support the crystal holders and crystals. The ΘA and ΘB crystal axes are identical, and function in a manner similar to the 2ΘA axis. Rotations are performed by the combination of a MST 1440 indexing table and a precision bearing. The MST 1440 tables are capable of 360° rotations in discrete steps of 0.25°, with an angular uncertainty of ±0.125 arc seconds at each angular position. The 1440 tables can be rotated by the combination of a lift/lock mechanism and a 360° rotary drive mechanism. The 1440 indexing tables ride on precision bearings, also fabricated by MST, which serve as fine angle dividers. Tangent arms, mounted on the upper precision bearing plates, are driven by Burleigh inchworm motors, providing the fine angular motion

Fig.1.



2ØB 360° Drive
Yoke
Detector
ØA Xtal
ØB Xtal
2ØA 360° Drive
2ØA Lift/Lock
2ØA Precision Bearing
Baseplate



2ØA Fine Motion
ØB Inchworm
& Encoder
ØA Lift/Lock
& 360° Drive
ØB Lift/Lock
& 360° Drive
ØA Inchworm
& Encoder

for the ΘA and ΘB axes. These axes are thus capable of 360° rotations in discrete angular steps of 0.1 arc seconds.

The 2ΘB axis is mounted on the upper horizontal extension of the yoke, and is responsible for the correct positioning of the x-ray detector. This axis is parallel and coaxial to the ΘB axis. A stepper motor is used to drive a worm, which in turn drives a worm gear mounted on an aluminum shaft. The gears are preloaded by spring tension. The aluminum shaft extends through a hole in the yoke, and is supported by a thrust bearing, allowing rotation about the 2ΘB axis. The Manson x-ray detector is mounted facing the 2ΘB axis on a level aluminum channel, which is connected to the aluminum shaft. This axis can be rotated through a full 360° angle in discrete steps of 18 arc seconds.

Alignment Procedure

Several alignments must be performed for the PDCS to function as a precision instrument, and our ability to perform accurate alignments directly affects the precision capabilities of the instrument. In general, the A and B axes must be parallel, and the plane of dispersion will be defined as the plane perpendicular to the A and B axes. The Bragg planes of the A and B crystals must be aligned parallel to the axes of rotation, and the x-ray beam must be parallel to the plane of dispersion. The ΘA, ΘB, and 2ΘA axes all use tangent arms with linear actuators for the small angle rotations. These systems must be aligned and calibrated in order to measure the small angles accurately.

The first step in the alignment procedure is to make the 2ΘA and ΘA axes parallel and coaxial. The set-up for these alignments is shown in Fig. 2. Both the coaxiality and parallelism of the 2ΘA and ΘA axes must be checked after an adjustment has been made. The alignment checks should only be performed with both the ΘA and 2ΘA indexing tables in the locked position, and with the ΘA and 2ΘA fine motion bearings engaged (lifting jacks lowered). The auto-collimator and diagonal mirror should be held in fixed positions independent of the spectrometer.

To obtain parallelism between the ΘA and 2ΘA axes, we must first align the auto-collimator axis and the 2ΘA axis. Adjustment of the optical flat located on the ΘA indexing table top may be necessary to ensure that the optical flat is parallel to the 2ΘA plane of rotation. Optical wax was used to hold the optical flat to the top of the ΘA indexing table. If the optical flat was misaligned with respect to the 2ΘA axis, circular motion of the auto-collimator image was observed as the 2ΘA axis was rotated, with the ΘA axis fixed. After the optical flat was made parallel to the 2ΘA axis, the ΘA axis was rotated while the auto-collimator and 2ΘA axis were held fixed. A circular motion of the autocollimator image

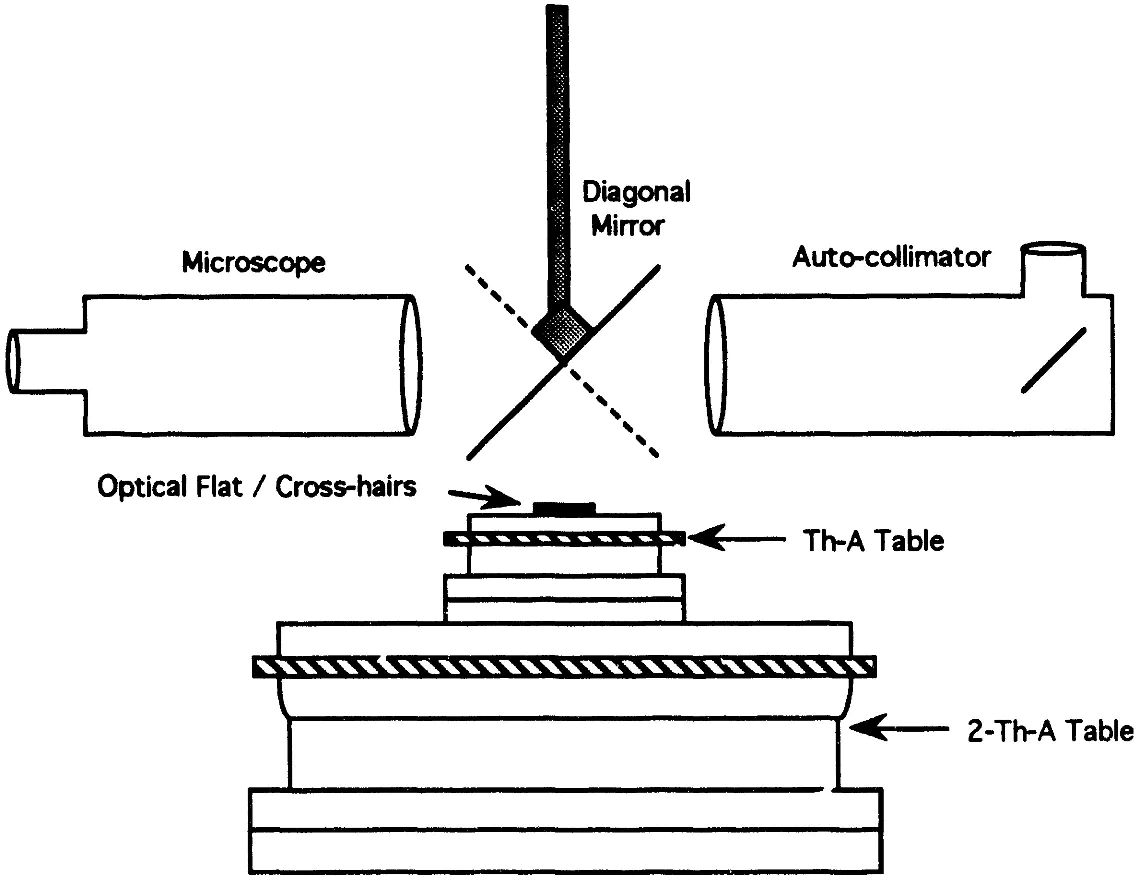


Fig. 2. Alignment Configuration for ΘA, 2ΘA parallelism, and ΘA, 2ΘA coaxiality.

indicated that the ΘA and 2ΘA axes were misaligned. The ΘA axis was then shimmed between the yoke and the precision bearing until the auto-collimator image remained fixed. The accuracy of this alignment is limited by the resolution of the auto-collimator image, however, the alignment was repeatable within approximately 1 arc second.

The optical flat was removed from the top of the ΘA indexing table, and a set of cross-hairs were placed on the table. The cross-hairs were observed through a microscope, and centered with respect to the ΘA axis. This was accomplished by moving the cross-hairs until the reference point determined by the intersection of the cross hairs remained fixed as the ΘA axis was rotated. The ΘA axis was then held fixed, while the 2ΘA axis was rotated. A circular motion of the cross-hairs indicated that the ΘA and 2ΘA axes were not coaxial. The ΘA axis was translated with respect to the yoke until the ΘA and 2ΘA axes were coaxial, and the parallelism alignment was rechecked. The seven μm width of the cross-hairs provides a good estimate for the uncertainty of the coaxial alignment.

Now that the ΘA and 2ΘA axes have been aligned for parallelism and coaxiality, the ΘB axis must be aligned parallel to the ΘA and 2ΘA axes, and coaxial to the 2ΘB axis. The alignment set-up is shown in Fig. 3. Optical flats are placed in the crystal holders such that reflections off both the front and back of the optical flats are possible. The optical flat is aligned parallel to its axis of rotation by observing the auto-collimator image reflected by the front of the optical flat. The ΘA indexing table is lifted, rotated 180°, and locked, and the auto-collimator image is observed from the back of the optical flat. The crystal holder is then tilted until the auto-collimator alignment is the same for both the front and back of the optical flat, making the optical flat parallel to the axis of rotation. The auto-collimator axis can now easily be aligned parallel to the plane of dispersion.

The optical flat is then removed from the ΘA crystal holder, while care is taken to ensure that the auto-collimator is not moved. The auto-collimator image for reflections off the ΘB axis optical flat is then located. The ΘB axis is rotated 180° to observe reflections off the back of the optical flat, and the ΘB axis crystal holder is rotated through its horizontal axis so that both the front and back optical reflections can be observed with the auto-collimator. The addition of shims of the ΘB axis between the yoke and the lower plate of the ΘB precision bearing may be needed to accomplish this alignment. The completion of this alignment indicates that the ΘB axis optical flat is parallel to the axis of rotation, and that the ΘA and ΘB axes are parallel with respect to the yoke axis.

We must also ensure that the ΘA and ΘB axes are parallel with respect to a 90° angle relative to the yoke axis. The auto-collimator is now placed in position 2 of Fig. 3, and the optical flat in the ΘA axis crystal holder is adjusted so that it is parallel to the ΘA axis of rotation as described previously. A large optical flat is attached to the yoke, and is adjusted so that it is parallel to the ΘA optical flat. The auto-collimator is now moved to position 3 in Fig. 3, and parallelism of the ΘB axis optical flat was checked for parallelism with the image from the large optical flat, adding shims if necessary. This completes the alignment procedure for parallelism of the 2ΘA, ΘA, and ΘB axes.

The 2ΘB and ΘB axes were also checked for coaxiality. A dial indicator was attached to a special extension of the 2ΘB detector arm. The dial indicator was placed against the circular edge of the indexing table, and the run-out was measured as the 2ΘB axis was rotated around the ΘB axis. The uncertainty of this coaxiality measurement is estimated to be the approximately the least count on the dial indicator, which is 0.001 inches. The parallelism of the ΘB axis with respect to the ΘA and 2ΘA axes must be rechecked to ensure that translation of the ΘB axis has not affected the alignment for parallelism.

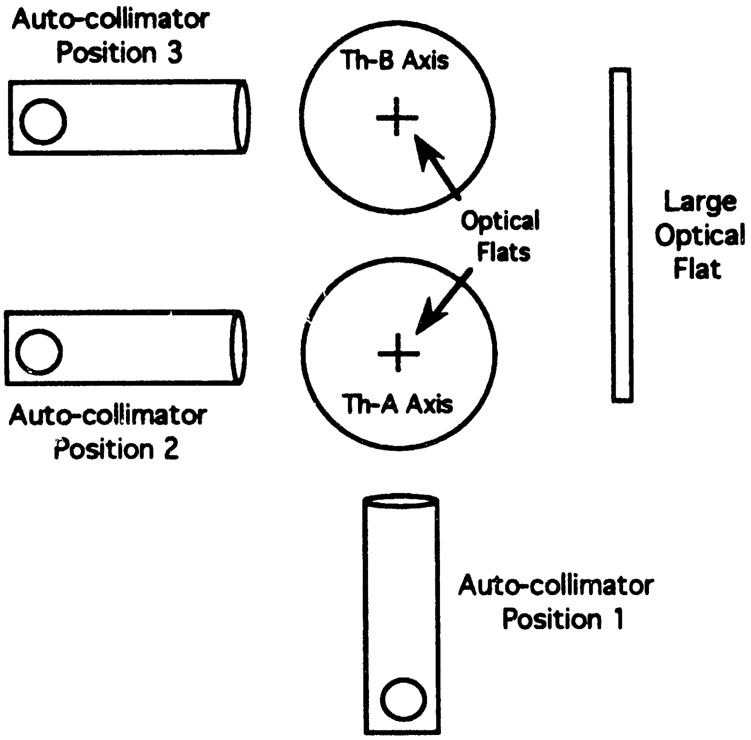


Fig. 3. Alignment Set-up for QA, QB parallelism.

Motion Control Systems

Stepper Motors

A total of eight stepper motors are used to perform various motion control functions for the PDCS. Two high-torque six-amp vacuum compatible stepper motors are used to operate the operate the 2ΘA lift/lock mechanism and the 2ΘA 360° drive assembly. The remaining six stepper motors are smaller three-amp vacuum compatible stepper motors which run the ΘA and ΘA lift-lock mechanisms, the ΘA and ΘB 360° drive assemblies, the 2ΘB 360° drive assembly, and the 2ΘA fine motion drive assembly. Each motor is accompanied by an AL drive, which provides the power and the control signals to the stepper motors. Two Compumotor 3000 indexers are programmed to control the eight stepper motors. The indexers can be operated by either local or remote control.

Precautions must be taken to prevent the three indexing tables from being rotated while they are in the locked position. This is accomplished by microswitches which are closed when the tables are in the lifted position. The 360° drive motors can then be operated to rotate their respective axes. If the tables are in the locked position, the interlock is open and the 360° drive is inoperative. The interlock microswitches also provide a TTL logic level command to the 3000 indexer. The purpose of this command is to ensure the tables can only be lifted if they are in the locked position and visa versa.

Temperature stability is important to the precision capabilities of the instrument, and the motor temperatures can warm to approximately 50° C when left on for extended periods of time. To avoid running the motors continuously, it is possible to use the shutdown command to shut off power to the motor. The motor will remain fixed in one of the 200 magnetic detents that exist in the 360° motor rotation. The Compumotor controllers are programmed to move the motor to one of its detent positions, and then shutdown the motor. Each motor has a running time of a few seconds, and there is no appreciable rise in motor temperature.

Inchworm Positioning System

The PDCS utilizes two Burleigh model IW-700-10 vacuum compatible inchworm motors to obtain fine motion in the ΘA and ΘB axes. The motors drive a tangent arm mounted on the upper precision bearing plate of the respective axis. Heidenhain electro-optical encoders are used to sense the position of the tangent arm, and these feedback signals are amplified and processed by the two Heidenhain model EXE-702 electronics units. The entire inchworm positioning system is controlled by a Burleigh 7000 controller, which is linked to the computer through the IEEE-488 interface. The controller reads the positioning orders from the computer, and directs the inchworm motors to move towards the desired position. The encoder network measures the change in position, and tells the controller when the system is correctly positioned. The controller then orders the inchworm motors to stop, and the feedback loop holds the inchworms in the desired position. For a more complete description of the inchworm motors, controller and encoder system, the reader is referred to the appropriate technical manuals.

For the fine positioning systems that uses the combination of a tangent arm and a translational positioning system, it is important to have the tangent arm and the encoder head aligned perpendicular to each other. The angular rotation of the axis, Θ, is related to the inchworm distance of travel, x, by the equation

$$\Theta = \tan^{-1}(x/a),$$

where a is the fixed distance between the inchworm translational centerline and the rotational axis. Using the Taylor Series expansion, we have

$$\Theta = (x/a) - 1/3\,(x/a)^3 + O(x/a)^5.$$

In order to meet the requirement that Θ have an angular accuracy of 0.1 seconds of arc, we must have

$$1/3\,(x/a)^3 < 5 \times 10^{-7},$$

or

$$|x/a| < 0.0114 \cong 0.65^\circ$$

Since 0.25° of total rotation is required for the ΘA and ΘB fine motion, and 0.50° of total rotation is required for the 2ΘA axis, the tangent arms must be closely perpendicular to the centerline of the translational device.

The inchworm and encoder system can be programmed to move in discrete steps of 0.1 µm. This can be converted to least count angular accuracy by

$$Q = x/a$$

and

$$\delta\Theta = (1/a)\,\delta x.$$

For a = 20.6265 cm for the ΘA and ΘB tangent arm lengths, a step of 0.1 µm in inchworm travel corresponds to a rotational motion of 0.1 arc seconds. To obtain 0.25° of total rotation, 9000 discrete inchworm positions are required. To maintain 0.1 arc seconds of absolute accuracy over the entire range of travel, the tangent arm length must be made accurate to approximately one part in 20,000, or

$$a = 20.6265 \pm 0.0010 \text{ cm}.$$

Alternatively, the tangent arm length may be made arbitrary, and calibration data in the form of number of discrete inchworm steps per quarter degree of angular motion can be placed in the computer program. The calibration data is obtained by observing the autocollimator image from an optical flat on the table, and counter-rotating the indexing table and the fine motion subassembly for the appropriate axis by 0.25°. This is the method presently being used.

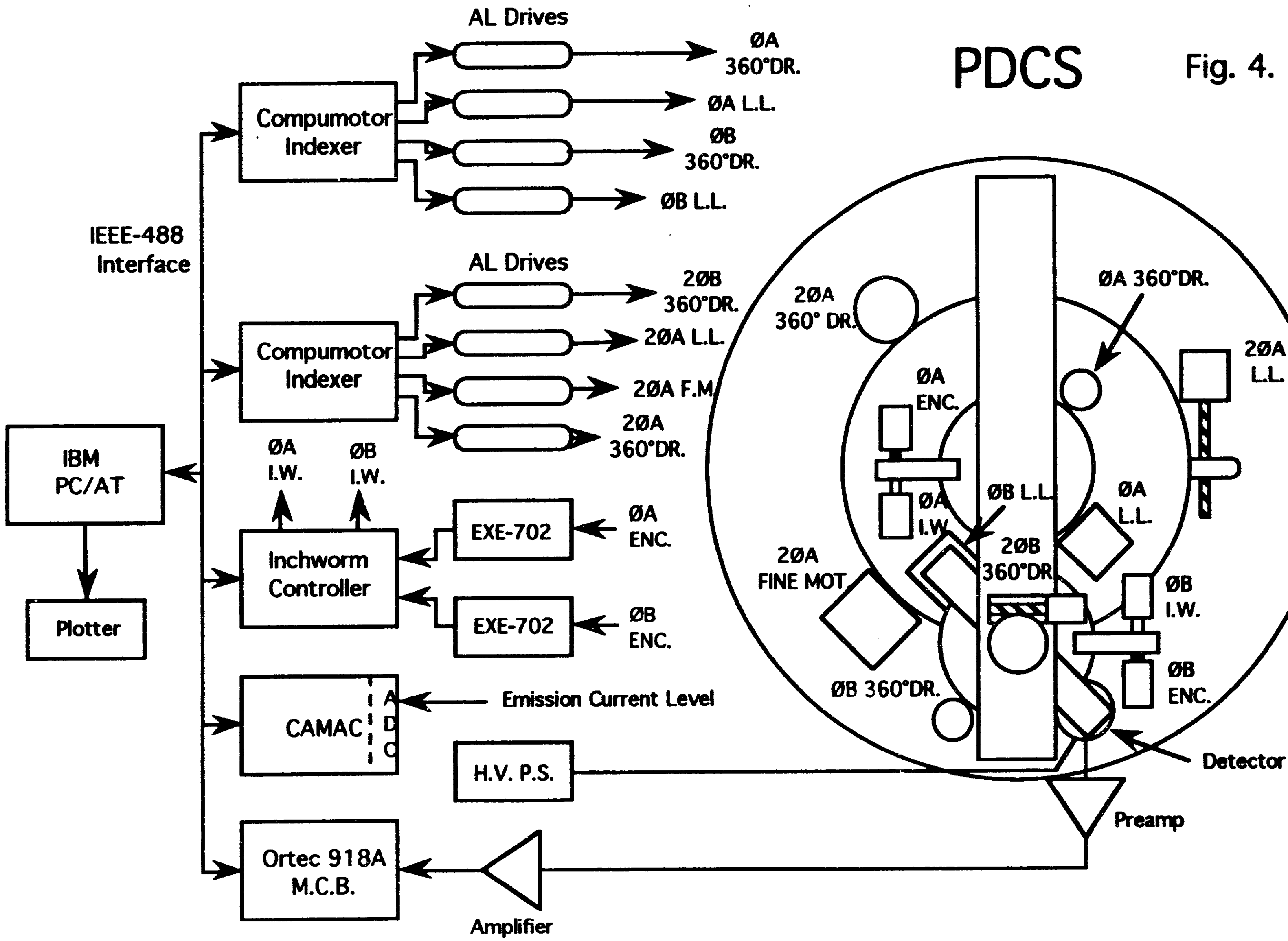
In order to obtain an angular resolution of 0.01 arc seconds, which is the desired resolution for high-energy rocking curves, the inchworm must be capable of making discrete translational steps of 0.01 μm. This is beyond the capability of our system in its present configuration. To achieve this specification, it will be necessary to either obtain an encoder with 0.01 μm of resolution, or find a method for microstepping one of the inchworm motors.

Computer Control System

The PDCS is run by an IBM model AT personal computer utilizing software written in the ASYST 2.0 language. The devices used to operate the instrument are controlled by the computer through an IEEE-488 parallel interface, as shown in Fig. 4. Five devices are linked to the computer through the interface: two Compumotor 3000 indexers for the eight stepper motors, the Burleigh inchworm controller, an Ortec model 918A multi-channel buffer, and a CAMAC crate. The interface is operated in a synchronous (foreground only) mode.

The software responsible for the initialization of the instrumentation associated with the PDCS is contained within the MAINLOAD.UTL program. This program (1) defines the windows used for the display, (2) defines large memory arrays and variables, (3) allows access to the position memory, (4) defines GPIB addresses, (5) initializes the GPIB bus, (6) loads specialized programming for the operation of the PDCS instrumentation, and (7) initializes the PDCS instrumentation.

Since the stepper motors for the PDCS are operated open loop i.e., no devices such as encoders are used to obtain positional feedback, a system has been devised to keep track of the positions of the different axes as the PDCS is moved from position to position. The ASYST data file POS.MEM stores five integer values associated with the 2ΘA, ΘA, ΘB, 2ΘB course positioning mechanisms and the 2ΘA fine motion. If any of the axes are moved, the operating program will retrieve the data values from POS.MEM. The data values will be changed according to the change in position of the axes, and the new position data will be restored in the POS.MEM file. The inchworm systems do not require this type of data storage, because they are operated closed loop.


PDCS
Fig. 4.
AL Drives
ØA 360°DR.
ØA L.L.
ØB 360°DR.
ØB L.L.
IEEE-488 Interface
Compumotor Indexer
AL Drives
2ØB 360°DR.
2ØA L.L.
2ØA F.M.
2ØA 360°DR.
Compumotor Indexer
IBM PC/AT
Plotter
ØA I.W.
ØB I.W.
Inchworm Controller
EXE-702
ØA ENC.
EXE-702
ØB ENC.
CAMAC
A D C
Emission Current Level
H.V. P.S.
Ortec 918A M.C.B.
Amplifier
Preamp
Detector
2ØA 360° DR.
ØA 360°DR.
2ØA L.L.
ØA ENC.
ØA I.W.
ØB L.L.
ØA L.L.
2ØA FINE MOT.
2ØB 360°DR
ØB I.W.
ØB ENC.
ØB 360°DR.

A program called ALIGNMENT.WRD is responsible for assigning an initial integer value to each of the five open loop positioning sub-systems. A HeNe laser is set up opposite the x-ray source to simulate the x-ray beam exiting the x-ray tube. The yoke is then manually positioned parallel to the beam axis with the ΘB axis opposite the x-ray tube, with the inchworm system energized and engaged. The ΘA and ΘB crystals are positioned to reflect the laser beam off the back of the crystals, at a 180° angle. The detector is aligned on the beam axis, "looking into" the x-ray tube. This is the "zero position" of the spectrometer, and the appropriate integer values are assigned to the position variables, and down-loaded into POS.MEM.

Vacuum System

The vacuum system for the PDCS is shown in Fig. 5. The vacuum chamber is initially at atmospheric pressure with both the gate valve and the throttle valve closed. The rotary vane pump is turned on, and the zeolite trap is outgassed by a resistance heater. Once the zeolite trap has been sufficiently outgassed, the throttle valve is opened, and evacuation of the vacuum chamber is begun. The pressure is monitored with a thermocouple gauge, and overnight, the pressure drops into the 10^{-3} torr range.

The turbo-pump is responsible for obtaining high vacuum for the PDCS. The turbo-pump is initially roughed out to approximately 10^{-2} torr by a mechanical fore-pump with the gate valve closed. The gate valve is operated by a cylinder of compressed air. The turbo-pump is then turned on, the gate valve is opened, and the throttle valve is closed. The rotary vane pump may then be taken off line, and returned to atmospheric pressure. The vacuum chamber ultimately reached a pressure of 2×10^{-6} torr.

It is necessary to protect the pumping system components from a loss of power. The vacuum system is powered through the emergency power system in room 60, but this power system has been unreliable in the past. A power sensor system has been incorporated which closes the gate valve on a loss of power. This power sensor system must be turned on by the operator, and the compressed air system valve line up must be in the proper position. A loss of power (for more than ten minutes) also causes a valve in the turbo-pump foreline to open, which initializes a slow leak. This leak is designed to slow the turbo-pump rotor, thereby preventing damage caused by the loss of the levitating magnetic field. A battery pack provides emergency power to the levitating magnetic field, if power is lost to the turbo-pump controller.

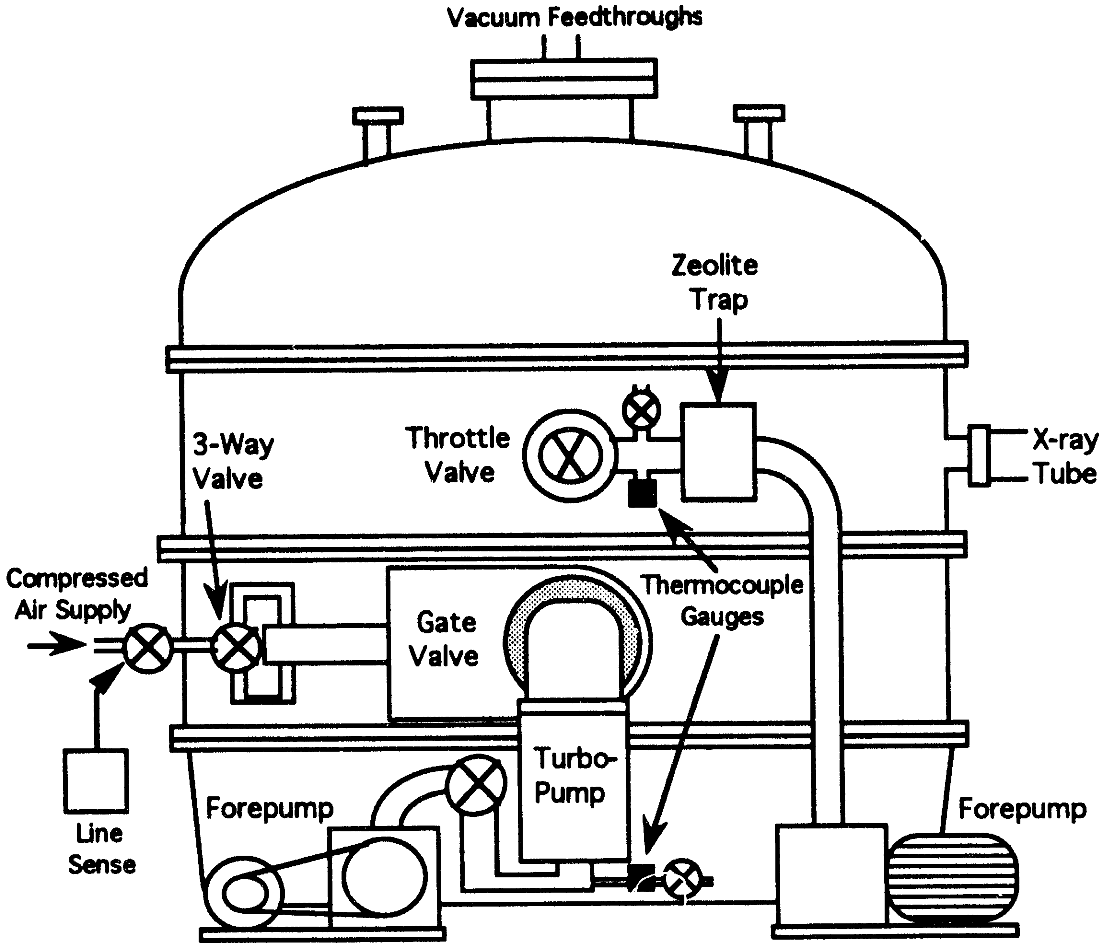


Fig. 5. PDCS Vacuum System.

OPERATING INSTRUCTIONS

To operate the PDCS, the power to the NIM bin, CAMAC crate, Compumotor controllers, AL drives, inchworm controller, and the EXE electronics must all be turned on. The operator must enter the properly configured ASYST mode, and receive an "OK" prompt. The operator must type

OK LOAD MAINLOAD.UTL

and type a carriage return. This will load most of the special programming for the PDCS. After MAINLOAD.UTL has been loaded, the computer should respond with:

Hit F3 to initialize the system...
OK

The operator hits the F3 key, and the on-line equipment will report the status of the initialization back to the computer. If all the on-line equipment is properly initialized, the computer will automatically load the display menu programming, and respond with:

Hit F3 to display the main menu...
OK

The operator hits the F3 key, and the main menu is displayed. The main menu is shown in Fig. 6. This display shows the lift/lock status of the indexing tables, the position data for all the axes, and the list of available PDCS operations. The operator must now select the desired operational program.

Alignment Program

By selecting F9 on the main menu, the operator loads the alignment program. This program is presently designed to give an approximate alignment of the rotational positions of the axes using a He-Ne laser, when the PDCS is equipped with crystals whose front and back optical faces are parallel to the Bragg planes. The crystals, yoke, and detector are aligned according to the instructions given on the computer screen by the alignment program. Once the alignment has been achieved, the operator should hit the F10 key to store the zero-position data in POS.MEM. After this has been accomplished, the PDCS

AXIS	LIFT/LOCK	COURSE POS.DAT	FINE POS.DAT
2.TH.A	LOCKED	(+/- 360) 0	(0-9000) 0
TH.A	LOCKED	(+/- 360) 0	(0-9000) 0
TH.B	LOCKED	(+/- 360) 0	(0-9000) 0
2.TH.B			(+/-36E3) 0

OK

MAIN MENU

F4 Axis Positioning	F5 Scan Sequence (n,n)
F6 Rocking Curve (n,-n)	F7 One-Crystal Instrument
F8 Set Detector High Voltage	F9 Align Instrument

Fig. 6. Main Menu.

position may only be changed by using the scan programs which are designed to continuously update the position of the PDCS axes.

Scan Programs

Before starting a scan program, the detector system should be made ready. The gas valve for the flowing gas proportional counter should be opened, and the detector high voltage should be turned on. The power to the detector electronics should be turned on, and the x-ray tube should be energized by turning on the anode high voltage and the filament power supply. The operator should then select either the Scan Sequence (n,n), or the Rocking Curve (n,-n) operating program from the main menu by selecting the appropriate "F" key.

The Scan Sequence (n,n) is the operating program used for obtaining dispersive x-ray spectra, and loaded when the operator selects the F5 key from the main menu. The program will prompt the operator for the initial and final Bragg angles of the scan. The operator must subdivide these Bragg angles into degrees (truncated to the nearest 0.25 degrees) and arc seconds (0-900 arc seconds). The operator must also choose the scan increment and the preset charge. The preset charge is simply the x-ray tube emission current, which is monitored by an analog-to-digital converter in the CAMAC crate, integrated over time. The counting time interval is therefore determined by the x-ray tube emission current and the value for preset charge entered by the operator. The PDCS will perform the scan automatically according to the input parameters. The data, which contains the number of x-ray counts as a function of the Bragg angle, is stored in a standard data file named SCAN.DAT. The operator must rename this data file to retain the data, because SCAN.DAT will be deleted the next time the program is run.

The operator may run the Rocking Curve (n,-n) program by selecting the F6 key on the main menu. The operator selects the appropriate parameters when prompted to do so by the computer, in a manner similar to the Scan Sequence (n,n) program. The PDCS will perform the rocking curve automatically, according to the input parameters, and store the data in ROCKING.DAT, which is a standard data file. The operator must rename this data file to save the data, because ROCKING.DAT will be deleted the next time the program is run.

OBSERVATIONS OF THE COPPER Kα LINES

X-ray Tube

The x-ray source used to test the PDCS is shown in Fig. 7. X-rays are produced as electrons from the hot filament strike the anode. The x-rays leave the anode, travel through the thin aluminum window and the collimator, and are then observed by the PDCS. An ultra-high vacuum system maintains the x-ray tube at 10^{-8} torr.

The x-ray tube utilizes the combination of an ion pump and a titanium sublimation pump to maintain vacuum in the x-ray tube. Indium gaskets were used to make the necessary seals in the vacuum system. The system was initially roughed-out to approximately 1×10^{-7} torr by an 80 l/sec diffusion pump backed by a mechanical fore-pump. After the system was roughed-out, a low-temperature (~100° C) bakeout was performed. The ion pump and sublimation pump were turned on, and the vacuum system was isolated from the roughing pumps. The ultimate vacuum pressure achieved was $< 1.0 \times 10^{-9}$ torr, but the system typically ran at $\sim 10^{-8}$ torr with the x-ray tube turned on. These pressures were measured by an ionization gauge mounted on the x-ray tube.

A copper anode was used to generate a characteristic x-ray spectrum for observation by the PDCS. The copper anode was cooled by cooling fins located outside the vacuum, and a fan. The electron beam was generated by thermionic emission from a got thoria-coated iridium filament. The electrons, incident on the copper anode, generated x-rays emitted in a direction perpendicular to the anode, thereby limiting anode self-absorption. The x-rays emitted by the anode must travel through a thin aluminum window which serves as the vacuum boundary between the x-ray tube and the PDCS.

Copper Kα Spectrum and Rocking Curves

The copper $K\alpha_1$ and $K\alpha_2$ lines obtained by the PDCS are shown in Fig. 8. The data was obtained by operating the scan program in first order. The x-ray tube was operated at 15 kV, with an emission current of approximately 3.0 mA. The preset charge was set for 1.5 C, which corresponds to a counting time interval of about 500 sec. The scan was performed in increments of 4.5 arc seconds, from a Bragg angle of about 10.25° to 10.35°.

The rocking curve obtained by the PDCS for the copper source is shown in Fig. 9. This rocking curve was taken at the Bragg angle corresponding to the $K\alpha_1$ line determined by the data shown in Fig. 8. The x-ray tube anode was held at 15 kV, with an emission

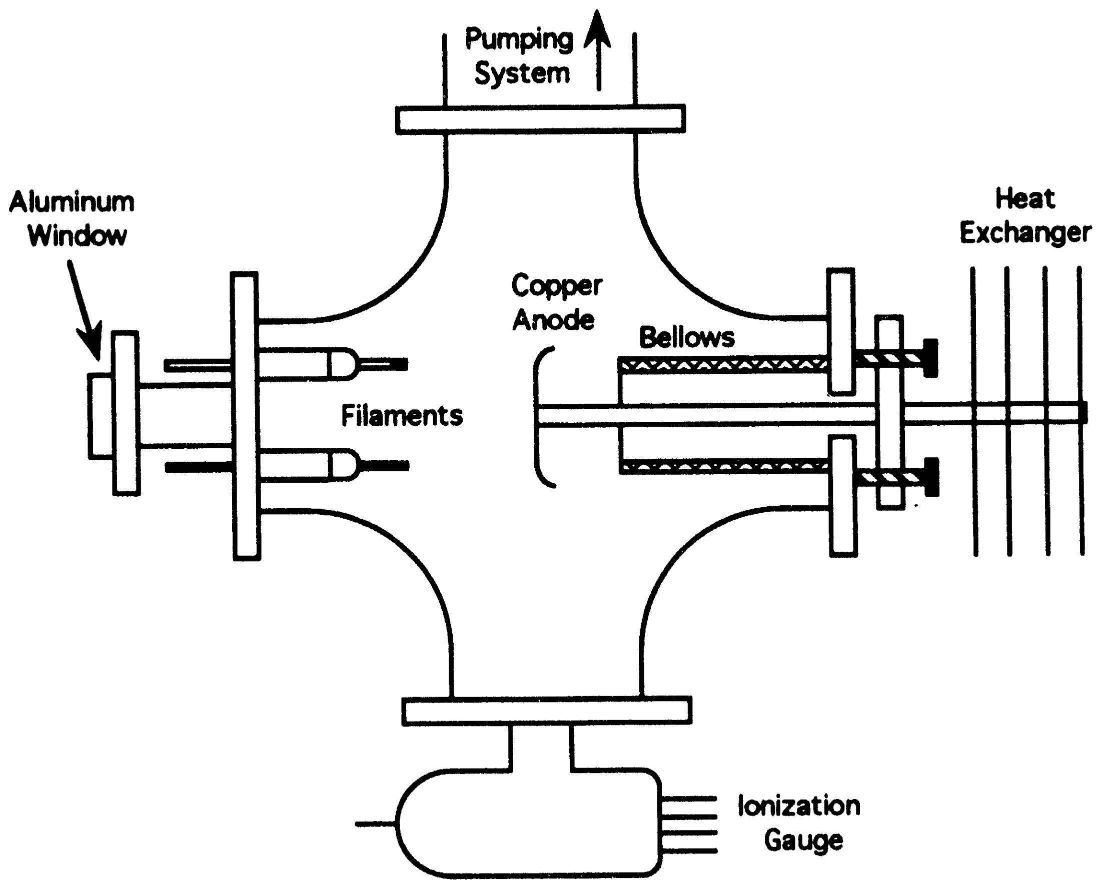


Fig. 7. X-ray Source.

current of approximately 2.0 mA. The preset charge was 0.5 C, which corresponds to a counting time interval of about 25 sec. The rocking curve was performed in increments of 1.0 arc seconds over a range of approximately 30 arc seconds.

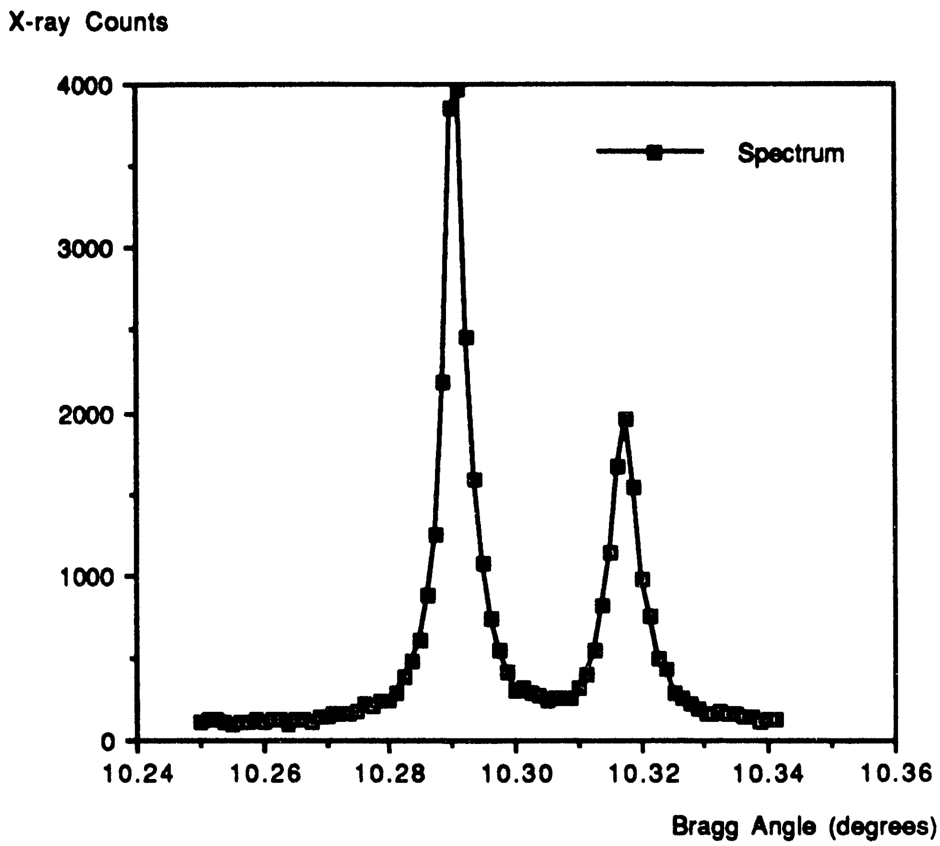


Fig. 8. Copper Kα X-ray Emission Spectrum.

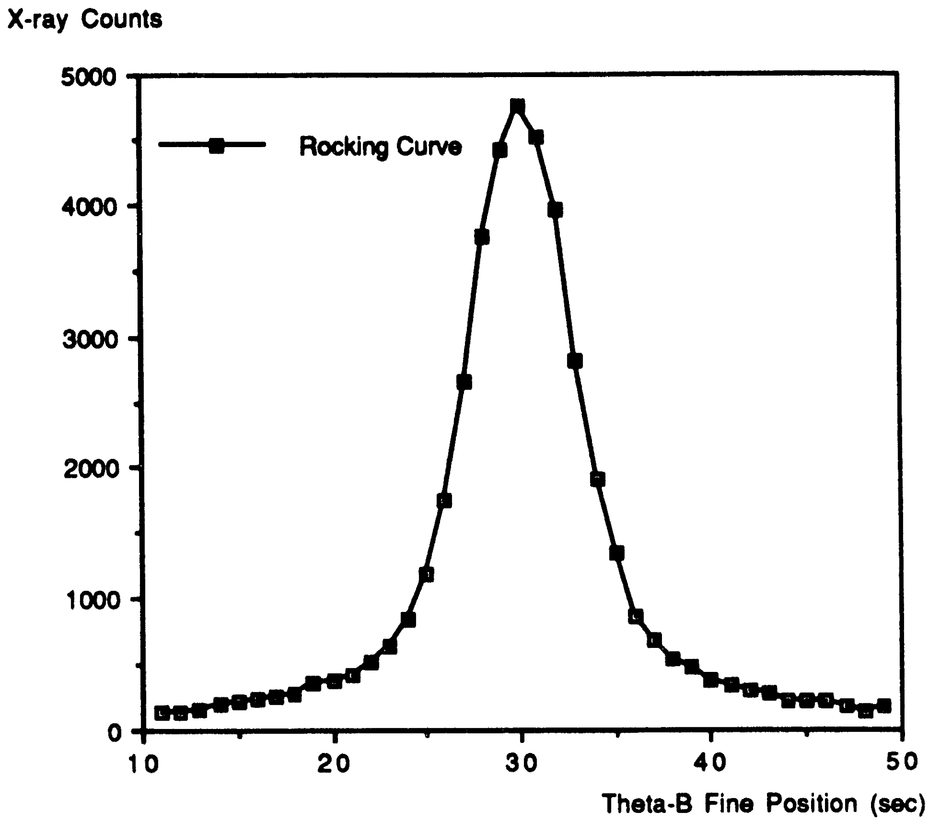


Fig. 9. Rocking Curve.

PDCS CHARACTERISTICS AND SPECIFICATIONS

Comparison of LANL P-14 Specifications with Measured Specifications

A comparison of LANL P-14 Specifications with the actual achieved characteristics of the PDCS is shown in Table 1. Some of the specifications described the P-14 statement of work for the MST contract have not been included in Table 1 for various reasons. The need for encoders for the stepping motors were included as an instrument specifications, and we have shown that these encoders are not necessary. Specifications were made for the fine scan range of motion to be 1.0° for ΘA and ΘB, and 2.0° for 2ΘA and 2ΘB, but in fact the fine scan range is 360° for all axes. The temperature stability and control specification of ±0.1°C has not yet been tested.

Table 1.
PDCS Characteristics and Specifications.

	Specification	*Achieved Value*	*Measurement Method*
Vacuum	1 x 10^{-7} torr	2 x 10^{-6} torr	Ionization Gauge
Scan Range	0° to ±80°	0° to ±80°	Visual Observation
Angular Precision, ΘA, ΘB, 2ΘA	0.1 arc sec	--------------	--------------
Angular Resolution, ΘA, ΘB, 2ΘA	0.01 arc sec	0.1 arc sec	Heidenhain Inch-worm Encoder
Parallelism, ΘA, ΘB, 2ΘA	10 arc sec	10 arc sec	Auto-collimator
Coaxiality, ΘA, 2ΘA	0.5 µm	7 µm	Cross-hair Width
Angular Resolution, 2ΘB	10 arc sec	18 arc sec	Auto-collimator

The ultimate vacuum pressure obtained by our pumping system was 2 x 10^{-6} torr as measured by an ionization gauge, while the specifications called for a vacuum pressure of 1.0 x 10^{-7} torr. A vacuum pressure of 2 x 10^{-6} torr allows for a sufficiently long mean free path for ultra-soft x-rays, and is also low enough to prevent damage to the inchworm motors and stepper motors in the plasma breakdown region of 1 x10^{-4} to 100 torr. These motors should never be operated in this pressure range. The only reason for operating the PDCS at 1 x 10^{-7} torr is to use a windowless photo-electric detector for the observation of ultra-soft x-rays. If this type of experiment is to be performed, it may be necessary to improve the vacuum chamber pressure. This may be accomplished by the implementation of a high vacuum pump with a pumping speed that is an order of magnitude faster.

The achieved value for angular accuracy has been omitted from Table 1. Our method for determining this value was to use an auto-collimator mounted on a separated MST precision 1440 indexing table, to observe the reflection from an optical flat mounted on the axis under test. The reference table and the associated axis were counter-rotated by 0.25°, and displacement of the auto-collimator image would have been observed if it were present. This calibration system is limited by the visual acuity of the auto-collimator observer, and we estimate this uncertainty to be slightly greater than 1.0 arc seconds. The random counter-rotation checks showed no visible displacement of the auto-collimator image, but all 1440 positions of the ΘA and ΘB indexing tables, and all 720 positions of the 2ΘA indexing table have yet to be checked.

Future Work

Having achieved the goal of making the PDCS a functioning instrument, we must now focus on achieving the design specifications. Since the d-spacing of the crystals is dependent on the temperature of the crystals, accurate temperature measurements of the crystals must be made. No attempts have yet been made to determine if the 0.1°C temperature stability and control specification can be achieved. Another problem which needs to be addressed is the specification for an absolute angular accuracy of 0.1 arc seconds for the ΘA, ΘB and 2ΘA axis positions. The observations of these small angles will only be possible with an improved auto-collimator system.

For the PDCS to perform precision measurements of crystal d-spacings and x-ray wavelengths, two additional programs must be added to the list of PDCS operational programs. The PDCS is capable of operating as a single crystal instrument for x-ray reflections from both the ΘA and ΘB crystals. Using rotatable crystal holders, the angle of the Bragg reflecting planes compared to the optical face of the crystal could be determined, and the crystals could be positioned until the Bragg planes are parallel to the crystal axes of rotation.

The second computer program which needs to be developed is a four-theta-Bragg ($4\Theta_B$) program, which takes advantage of the symmetry of the PDCS. The scan sequence (n,n) program can also be run in a position (-n,-n) that is a mirror image with respect to the input x-ray beam. The position difference for the ΘA and ΘB axes for the spectral peak in the (n,n) position compared to the (-n,-n) position is 180° - $2\Theta_B$, while the difference in the 2ΘA position will be $4\Theta_B$ (which gives the operating program its name). By comparing the (n,n) data with the (-n,-n) data, the Bragg angle of the spectral peaks can be determined with a high degree of accuracy.

APPENDIX A

PDCS PROGRAMMING

MAINLOAD.UTL

```
ECHO.OFF

\ MAINLOAD.UTL is a command file for setting up a utilites menu and loading
\ the various utility programs.

\ The following instrument control subprograms are required:

\ 918.WRD.............Provides for control of the Ortec 918A MCB
\ MOTOR.WRD...........Provides for control of the Compumotor Indexers
\ IW.WRD..............Provides for control of the Inchworms
\ MAINTCAM.WRD........Provides for control of the CAMAC Crate
\ READ.FIL............Converts BASIC data files to ASYST variables

\ The following spectrometer operational subprograms are required:

\ MENU.WRD............Loads the menu & display
\ ONE-XTAL.WRD........Operates the Spectrometer in Single-Crystal Mode
\ SCAN.WRD............Operates the Spectrometer in (n,n) Scan Mode
\ SCAN2.WRD...........Operates the Spectrometer for (n,-n) Rocking Curves
\ AXIS-POS.WRD........Allows independent axis positioning & position memory
\ ALIGNMENT.WRD.......Sets the zero DCS position into position memory

\ DEFINE WINDOWS

0 0 1 79 WINDOW {SCALE}
2 0 5 9 WINDOW {AXES}
2 10 2 22 WINDOW {LIFT/LOCK.2.TH.A}
3 10 3 22 WINDOW {LIFT/LOCK.TH.A}
4 10 4 22 WINDOW {LIFT/LOCK.TH.B}
2 23 5 33 WINDOW {CPD1}
2 34 2 42 WINDOW {CPD.2.TH.A}
3 34 3 42 WINDOW {CPD.TH.A}
4 34 4 42 WINDOW {CPD.TH.B}
2 43 5 53 WINDOW {FPD1}
2 54 2 64 WINDOW {FPD.2.TH.A}
3 54 3 64 WINDOW {FPD.IW.A}
4 54 4 64 WINDOW {FPD.IW.B}
5 54 5 64 WINDOW {FPD.2.TH.B}
2 65 5 79 WINDOW {BRAGG}
19 0 24 79 WINDOW {MAIN.BOT}
22 32 22 40 WINDOW {BRAGG0}
23 32 23 40 WINDOW {BRAGGF}
24 30 24 40 WINDOW {SCAN.INC}
0 0 18 79 WINDOW {MAIN.TOP}
6 0 18 79 WINDOW {MAIN.MID}

\ DEFINE LARGE MEMORY VARIABLES
```

```
1000 STRING "MOTOR.DATA
20 STRING "918.STAT
20 STRING "918.$STAT
100 STRING "IW.DATA
REAL DIM[ 2000 ] ARRAY XX
REAL DIM[ 2000 ] ARRAY YY
INTEGER DIM[ 7 ] ARRAY POS.DAT

\ DEFINE POSITION MEMORY

: POS.DAT>MEM
  FILE.OPEN POS.MEM
  1 SUBFILE POS.DAT ARRAY>FILE
  FILE.CLOSE
;

: POS.MEM>DAT
  FILE.OPEN POS.MEM
  1 SUBFILE POS.DAT FILE>ARRAY
  FILE.CLOSE
;

\ DEFINE GPIB.ADDRESSES

 1 GPIB.DEVICE MOTOR1
 2 GPIB.DEVICE MOTOR2
 3 GPIB.DEVICE 972AMP
 4 GPIB.DEVICE CRATE1
 5 GPIB.DEVICE 918MCB
 6 GPIB.DEVICE 974AMP
 7 GPIB.DEVICE IW

\ PROGRAM TO INITIALIZE GPIB BUS

 : GPIB.INIT
    BUS.INIT                  \INITIALIZE THE BUS
    SEND.INTERFACE.CLEAR      \CLEAR THE BUS
    REMOTE.ENABLE.ON          \ALLOW DEVICES TO RESPOND
 ;

 : INIT.OK
    STACK.CLEAR
    LOAD MENU.WRD
    ECHO.ON
 ;

 LOAD C:MAINTCAM.WRD
 LOAD C:918.WRD
 LOAD C:IW.WRD
 LOAD C:MOTOR.WRD
 LOAD C:READ.FIL

 : DO.INIT
```

```
   NORMAL.DISPLAY
   GPIB.INIT
   MOTOR1.INIT
   MOTOR2.INIT
   918.INIT
   CAMAC.INIT
   IW.INIT
   INIT.OK
;

STACK.CLEAR
F3 FUNCTION.KEY.DOES DO.INIT
NORMAL.DISPLAY
CR ." Hit F3 to initilize the system."

ECHO.ON
```

918.WRD

```
\PROGRAM TO WRITE COMMANDS TO THE ORTEC 918 MULTI-CHANNEL
BUFFER,
\THEN READ THE STATUS
\COMMAND MUST BE IN THE SYMBOL STACK
\MAINLOAD.UTL MUST BE RUN FIRST

20 STRING "918.STAT
20 STRING "918.$STAT

DP.REAL SCALAR COUNTS
REAL SCALAR N
REAL SCALAR M
INTEGER SCALAR PRESET

: ORTEC.CHK
   " %" "918.STAT "WITHIN        \TEST FOR STATUS WORD (%)
   IF 1 + 3 "918.STAT "SUB "DUP   \SAVE FIRST 3 CHAR
      " 000" "WITHIN           \000 = SUCCESS
      IF ." OPERATION SUCCESSFUL." CR
         ELSE " 001" "WITHIN    \001 = POWER UP
         IF ." POWER-UP, " "918.STAT GPIB.READ
         " %000" "918.STAT "WITHIN IF ." OPERATION SUCCESSFUL." CR THEN
            ELSE ." ERROR " "918.STAT "TYPE CR
         THEN
      THEN
   ELSE ." OUT OF SYNC" CR " ORTEC.PRT.DATA" "EXEC
   THEN
;

: ORTEC.PRT.DATA
   " $A" "918.$STAT "WITHIN        \TEST FOR DATA RECORD
   IF 2 + 3 "918.$STAT "SUB        \IF $A, SAVE 3 CHAR
      "TYPE CR
   ELSE " $C" "918.$STAT "WITHIN    \TEST FOR C DATA RECORD
```

```
  IF 2 + 5 "918.$STAT "SUB          \ IF $C, SAVE 5 CHAR
    "TYPE CR
  ELSE " $F" "918.$STAT "WITHIN    \ TEST FOR F DATA RECORD
   IF 2 + 3 "918.$STAT "SUB         \ IF $F, SAVE 3 CHAR
    "TYPE CR
   ELSE " $G" "918.$STAT "WITHIN     \ TEST FOR G DATA RECORD
    IF 2 + 10 "918.$STAT "SUB        \ IF $G, SAVE 10 CHAR
    "TYPE CR
    ELSE " $D" "918.$STAT "WITHIN    \ TEST FOR D DATA RECORD
     IF 2 + 10 "918.$STAT "SUB       \ IF $D, SAVE 10 CHAR
      "TYPE CR
     ELSE " $H" "918.$STAT "WITHIN     \ TEST FOR H DATA RECORD
      IF 2 + 8 "918.$STAT "SUB        \ IF $H, SAVE 8 CHAR
       "TYPE CR
      ELSE ." NO $A,$C,$D,$F,$G OR $H RECORD " CR
      THEN
     THEN
    THEN
   THEN
  THEN
 THEN
  "918.STAT GPIB.READ
  ORTEC.CHK
;

\ PROGRAM TO SEND COMMANDS TO 918 MCB

: ORTEC.COM
 918MCB
 GPIB.WRITE              \ WRITE TO 918
  50 MSEC.DELAY
  "918.STAT GPIB.READ       \ READ REPLY
  ORTEC.CHK
;

\ PROGRAM TO SHOW REQUESTED VALUES - COMMAND ON THE SYMBOL
STACK

: ORTEC.SHOW
 918MCB
  GPIB.WRITE              \ WRITE TO 972
  "918.$STAT GPIB.READ       \ READ REPLY
  CR ORTEC.PRT.DATA
;

: 918.INIT
  918MCB
  60000 TIMEOUT
  EOS.ON
  10 EOS.CHARACTER
  CR ." INITIALIZATION OF ORTEC 918 MCB..."
  " INIT" ORTEC.COM
;
```

```
: COUNTS/SEC
  918MCB
  0 M :=
  " CLEAR_PRESETS" ORTEC.COM
  " START" ORTEC.COM
  SCREEN.CLEAR
  {SCALE} SCREEN.CLEAR
."       10      100     1000     10000   100000 (Xrays/sec.)" CR
."        v       v        v        v        v "
  500 0 DO
   140 MSEC.DELAY
   " SHOW_INTEGRAL" GPIB.WRITE
   "918.$STAT GPIB.READ
   {ASTERISKS} SCREEN.CLEAR
   "918.$STAT 3 10 "SUB 0 "NUMBER DUP DUP
   M - 0.1 > IF
    M - SWAP M :=
     I 0 <> IF
     2 * LOG 10 * 0 DO ." *" LOOP        \ LOG AMP 0.5 SEC LIVE GATE
     THEN
    THEN
   "918.STAT GPIB.READ                  \ EXTRA READ
   STACK.CLEAR
  LOOP
  " STOP" ORTEC.COM
  MAIN.MENU.DISPLAY
;

: 918.COUNT
 " START" ORTEC.COM
  BEGIN STACK.CLEAR
  100 MSEC.DELAY
  " SHOW_LIVE" ORTEC.SHOW "918.$STAT 3 10 "SUB 0 "NUMBER ?T/F
  PRESET =
  UNTIL STACK.CLEAR
 " SHOW_INTEGRAL" ORTEC.SHOW "918.$STAT 3 10 "SUB 0 "NUMBER ?T/F
;

: SET.LIVE.PRESET
 " SET_LIVE_PRESET "
 ." Enter detector live dwell time (# of 20 msec increments):  " "INPUT CR
 "CAT "DUP 16 7 "SUB 0 "NUMBER ?T/F "DROP PRESET := ORTEC.COM
;
```

<u>MOTOR.WRD</u>

```
\ PROGRAMS FOR THE TWO COMPUMOTOR 3000 INDEXERS

\ 1000 STRING "MOTOR.DATA (MAINLOAD.UTL)

INTEGER SCALAR STEP

: SPOLL
```

```
  BEGIN
  10 MSEC.DELAY
  32 SERIAL.POLL
  32 = IF "MOTOR.DATA GPIB.READ
  THEN
  48 SERIAL.POLL 16 =
  UNTIL
;

: MOTOR.INIT
  1000 TIMEOUT
  SPOLL
  " ACCESS 3000;" GPIB.WRITE
  SPOLL
  " RECALL;" GPIB.WRITE
  SPOLL
;

: MOTOR1.INIT
  MOTOR1
 CR ." INITIALIZATION OF COMPUMOTOR INDEXER #1..."
  MOTOR.INIT
  " INPUT?" "MOTOR.DATA "WITHIN IF ." OPERATION SUCCESSFUL."
  ELSE ." UNSUCCESSFUL."  BELL BELL BELL
  THEN
;

: MOTOR2.INIT
  MOTOR2
 CR ." INITIALIZATION OF COMPUMOTOR INDEXER #2..."
  MOTOR.INIT
  " INPUT?" "MOTOR.DATA "WITHIN IF ." OPERATION SUCCESSFUL."
  ELSE ." UNSUCCESSFUL."  BELL BELL BELL
  THEN
;

: MOTOR1.COM
 MOTOR1
  GPIB.WRITE
  SPOLL
;

: MOTOR2.COM
 MOTOR2
  GPIB.WRITE
  SPOLL
;

: SET.MOTOR.VARIABLE
 " VARIABLE Q1 = "
 ." Input number of steps (integer value) " "INPUT
 "CAT " ;" "CAT
;
```

```
: CHANGE.MOTOR.VARIABLE
  " VARIABLE Q1 = "
  "." "CAT
  " ;" "CAT
;

: CHECK.L/L.TH.A
  MOTOR2 SPOLL
\ " LOAD;" MOTOR2.COM
\ " 1200DISP   BLANK BLANK BLANK BLANK;" MOTOR2.COM
\ " 1210BRANCH  X0XX  XXXX  XXXX  XXXX THEN 1230;" MOTOR2.COM
\ " 1220DISP   BLANK BLANK BLANK  STAT;" MOTOR2.COM
\ " 1230DONE;" MOTOR2.COM
\ " *;" MOTOR2.COM
  " RUN FROM 1200;" MOTOR2.COM   200 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  10 MSEC.DELAY
  32 SERIAL.POLL
  32 = IF "MOTOR.DATA GPIB.READ
    " STOPPED" "MOTOR.DATA "WITHIN IF
     {LIFT/LOCK.TH.A} SCREEN.CLEAR ." LIFTED" ELSE
     {LIFT/LOCK.TH.A} SCREEN.CLEAR ." LOCKED" THEN
   THEN
  SPOLL
  STACK.CLEAR {MAIN.MID} SCREEN.CLEAR
;

: CHECK.L/L.TH.B
  MOTOR1 SPOLL
  " RUN FROM 1300;" MOTOR2.COM   200 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  10 MSEC.DELAY
  32 SERIAL.POLL
  32 = IF "MOTOR.DATA GPIB.READ
    " STOPPED" "MOTOR.DATA "WITHIN IF
     {LIFT/LOCK.TH.B} SCREEN.CLEAR ." LIFTED" ELSE
     {LIFT/LOCK.TH.B} SCREEN.CLEAR ." LOCKED" THEN
   THEN
  SPOLL
  STACK.CLEAR {MAIN.MID} SCREEN.CLEAR
;

: CHECK.L/L.2.TH.A
  MOTOR1 SPOLL
  " RUN FROM 600;" MOTOR1.COM   200 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  10 MSEC.DELAY
  32 SERIAL.POLL
  32 = IF "MOTOR.DATA GPIB.READ
    " STOPPED" "MOTOR.DATA "WITHIN IF
     {LIFT/LOCK.2.TH.A} SCREEN.CLEAR ." LIFTED" ELSE
     {LIFT/LOCK.2.TH.A} SCREEN.CLEAR ." LOCKED" THEN
   THEN
  SPOLL
```

```
STACK.CLEAR {MAIN.MID} SCREEN.CLEAR
;

: CHECK.L/L
 100 MSEC.DELAY
 CHECK.L/L.2.TH.A 100 MSEC.DELAY
 CHECK.L/L.TH.A 100 MSEC.DELAY
 CHECK.L/L.TH.B 100 MSEC.DELAY
;

: 2.TH.A.SHTDN
 " LOAD;" MOTOR1.COM
 " 0950SHTDN NULL  YES NULL NULL;" MOTOR1.COM
 " 0960DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 950;" MOTOR1.COM
;
: 2.TH.A.LOCK
 " LOAD;" MOTOR1.COM
 " 0300BRANCH X0XX XXXX XXXX XXXX THEN 0380;" MOTOR1.COM
 " 0310DISP  BLANK BLANK   STAT BLANK;" MOTOR1.COM
 " 0320SHTDN  NULL NULL      NO NULL;" MOTOR1.COM
 " 0330VELC   0000 0000    50000 0000;" MOTOR1.COM
 " 0340ACEL   0000 0000     5000 0000;" MOTOR1.COM
 " 0350MOVE   0000 0000 -1388888 0000;" MOTOR1.COM
 " 0360SHTDN  NULL NULL      YES NULL;" MOTOR1.COM
 " 0380DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 300;" MOTOR1.COM
50 0 DO
 {LIFT/LOCK.2.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
 ." LOCKING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.2.TH.A
20 MSEC.DELAY 2.TH.A.SHTDN
;

: 2.TH.A.STARTUP
 " LOAD;" MOTOR1.COM
 " 0900SHTDN  NULL   NO NULL NULL;" MOTOR1.COM
 " 0910DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 900;" MOTOR1.COM
;

: 2.TH.A.LIFT
 2.TH.A.STARTUP 200 MSEC.DELAY
 " LOAD;" MOTOR1.COM
 " 0400BRANCH X1XX XXXX XXXX XXXX THEN 0480;" MOTOR1.COM
 " 0410DISP  BLANK BLANK  STAT BLANK;" MOTOR1.COM
 " 0420SHTDN  NULL NULL      NO NULL;" MOTOR1.COM
 " 0430VELC   0000 0000   50000 0000;" MOTOR1.COM
 " 0440ACEL   0000 0000    5000 0000;" MOTOR1.COM
 " 0450MOVE   0000 0000 1388888 0000;" MOTOR1.COM
```

```
 " 0460SHTDN  NULL NULL     YES NULL;" MOTOR1.COM
 " 0480DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 400;" MOTOR1.COM
50 0 DO
 {LIFT/LOCK.2.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
 ." LIFTING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.2.TH.A
;

: TH.A.LIFT
 " LOAD;" MOTOR2.COM
 " 0500BRANCH X1XX  XXXX  XXXX  XXXX THEN 560;" MOTOR2.COM
 " 0510DISP  BLANK BLANK BLANK  STAT;" MOTOR2.COM
 " 0520SHTDN  NULL  NULL  NULL    NO;" MOTOR2.COM
 " 0530VELC   0000  0000  0000 10000;" MOTOR2.COM
 " 0540ACEL   0000  0000  0000  5000;" MOTOR2.COM
 " 0550MOVE   0000  0000  0000 -50000;" MOTOR2.COM
 " 0560SHTDN  NULL  NULL  NULL   YES;" MOTOR2.COM
 " 0570DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 500;" MOTOR2.COM
10 0 DO
 {LIFT/LOCK.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
 {LIFT/LOCK.TH.A} ." LIFTING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.TH.A
;

: TH.B.LIFT
 " LOAD;" MOTOR2.COM
 " 0800BRANCH XX1X  XXXX  XXXX  XXXX THEN 860;" MOTOR2.COM
 " 0810DISP  BLANK  STAT BLANK BLANK;" MOTOR2.COM
 " 0820SHTDN  NULL    NO  NULL  NULL;" MOTOR2.COM
 " 0830VELC   0000 10000  0000  0000;" MOTOR2.COM
 " 0840ACEL   0000  5000  0000  0000;" MOTOR2.COM
 " 0850MOVE   0000 -50000 0000  0000;" MOTOR2.COM
 " 0860SHTDN  NULL   YES  NULL  NULL;" MOTOR2.COM
 " 0870DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 800;" MOTOR2.COM
10 0 DO
 {LIFT/LOCK.TH.B} SCREEN.CLEAR 400 MSEC.DELAY
 ." LIFTING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.TH.B
;

: TH.A.LOCK
 " LOAD;" MOTOR2.COM
 " 0700BRANCH X0XX  XXXX  XXXX  XXXX THEN 770;" MOTOR2.COM
 " 0710DISP  BLANK BLANK BLANK  STAT;" MOTOR2.COM
 " 0720SHTDN  NULL  NULL  NULL    NO;" MOTOR2.COM
```

```
 " 0730VELC   0000  0000  0000 10000;" MOTOR2.COM
 " 0740ACEL   0000  0000  0000  5000;" MOTOR2.COM
 " 0750MOVE   0000  0000  0000 50000;" MOTOR2.COM
 " 0760SHTDN  NULL  NULL  NULL   YES;" MOTOR2.COM
 " 0770DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 700;" MOTOR2.COM
10 0 DO
 {LIFT/LOCK.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
 ." LOCKING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.TH.A
;

: TH.B.LOCK
 " LOAD;" MOTOR2.COM
 " 0900BRANCH XX0X  XXXX  XXXX  XXXX THEN 970;" MOTOR2.COM
 " 0910DISP  BLANK  STAT BLANK BLANK;" MOTOR2.COM
 " 0920SHTDN  NULL    NO  NULL  NULL;" MOTOR2.COM
 " 0930VELC   0000 10000  0000  0000;" MOTOR2.COM
 " 0940ACEL   0000  5000  0000  0000;" MOTOR2.COM
 " 0950MOVE   0000 50000  0000  0000;" MOTOR2.COM
 " 0960SHTDN  NULL   YES  NULL  NULL;" MOTOR2.COM
 " 0970DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 900;" MOTOR2.COM
10 0 DO
 {LIFT/LOCK.TH.B} SCREEN.CLEAR 400 MSEC.DELAY
 ." LOCKING" 400 MSEC.DELAY
LOOP
20 MSEC.DELAY CHECK.L/L.TH.B
;

: 2.TH.A.MOVE
 " LOAD;" MOTOR1.COM
 " 0500SHTDN NULL   NO NULL NULL;" MOTOR1.COM
 " 0510DISP BLANK STAT BLANK BLANK;" MOTOR1.COM
 " 0520VELC  0000 1500 0000 0000;" MOTOR1.COM
 " 0530ACEL  0000 1000 0000 0000;" MOTOR1.COM
 " 0540MATH  Q2 = Q1 * 1000;" MOTOR1.COM
 " 0550MOVE  0000   Q2 0000 0000;" MOTOR1.COM
 " 0560SHTDN NULL NULL NULL NULL;" MOTOR1.COM
 " 0570DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 500;" MOTOR1.COM
;

: 2.TH.A.ROTATE
 MOTOR1 SPOLL
 SET.MOTOR.VARIABLE MOTOR1.COM
 2.TH.A.MOVE
;

: CHANGE.2.TH.A
```

```
 DUP 0 = NOT IF
  STEP :=
  2.TH.A.LIFT
  POS.MEM>DAT STEP POS.DAT [ 1 ] + POS.DAT [ 1 ] := POS.DAT>MEM
  MOTOR1 SPOLL
  STEP CHANGE.MOTOR.VARIABLE MOTOR1.COM
  2.TH.A.MOVE
  BEGIN
    {CPD.2.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
    {CPD.2.TH.A} ." ROTATING" 400 MSEC.DELAY
    " REPORT;" GPIB.WRITE
    20 MSEC.DELAY
    32 SERIAL.POLL 32 = IF
    "MOTOR.DATA GPIB.READ THEN
    " STOPPED" "MOTOR.DATA "WITHIN
  UNTIL
  SPOLL STACK.CLEAR
  {CPD.2.TH.A} SCREEN.CLEAR POS.DAT [ 1 ] .
  {MAIN.MID}
  2.TH.A.LOCK
 THEN
 STACK.CLEAR
;

: TH.A.MOVE
 " LOAD;" MOTOR2.COM
 " 1000SHTDN  NULL  NULL   NO  NULL;" MOTOR2.COM
 " 1010DISP  BLANK BLANK  STAT BLANK;" MOTOR2.COM
 " 1020VELC  0000  0000 1900  0000;" MOTOR2.COM
 " 1030ACEL  0000  0000  500  0000;" MOTOR2.COM
 " 1040MATH  Q2 = Q1 * 500;" MOTOR2.COM
 " 1050MOVE  0000  0000   Q2  0000;" MOTOR2.COM
 " 1060SHTDN  NULL  NULL   YES  NULL;" MOTOR2.COM
 " 1070DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 1000;" MOTOR2.COM
;

: TH.A.ROTATE
 MOTOR2 SPOLL
 SET.MOTOR.VARIABLE MOTOR2.COM
 TH.A.MOVE
;

: CHANGE.TH.A
DUP 0 = NOT IF
 STEP :=
 TH.A.LIFT
 POS.MEM>DAT STEP POS.DAT [ 2 ] + POS.DAT [ 2 ] := POS.DAT>MEM
 MOTOR2 SPOLL
 STEP CHANGE.MOTOR.VARIABLE MOTOR2.COM
 TH.A.MOVE
 BEGIN
  {CPD.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
```

```
  ." ROTATING" 400 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  20 MSEC.DELAY
  32 SERIAL.POLL 32 = IF
  "MOTOR.DATA GPIB.READ THEN
  " STOPPED" "MOTOR.DATA "WITHIN
 UNTIL
 SPOLL STACK.CLEAR
 {CPD.TH.A} SCREEN.CLEAR POS.DAT [ 2 ] .
 {MAIN.MID}
 TH.A.LOCK
THEN
STACK.CLEAR
;

: TH.B.MOVE
 " LOAD;" MOTOR2.COM
 " 1100SHTDN   NO  NULL  NULL  NULL;" MOTOR2.COM
 " 1110DISP   STAT BLANK BLANK BLANK;" MOTOR2.COM
 " 1120VELC   1900  0000  0000  0000;" MOTOR2.COM
 " 1130ACEL    500  0000  0000  0000;" MOTOR2.COM
 " 1140MATH  Q2 = Q1 * 500;" MOTOR2.COM
 " 1150MOVE    Q2  0000  0000  0000;" MOTOR2.COM
 " 1160SHTDN   YES  NULL  NULL  NULL;" MOTOR2.COM
 " 1170DONE;" MOTOR2.COM
 " *;" MOTOR2.COM
 " RUN FROM 1100;" MOTOR2.COM
;

: TH.B.ROTATE
 MOTOR2 SPOLL
 SET.MOTOR.VARIABLE MOTOR2.COM
 TH.B.MOVE
;

: CHANGE.TH.B
DUP 0 = NOT IF
 STEP :=
 TH.B.LIFT
 POS.MEM>DAT STEP POS.DAT [ 3 ] + POS.DAT [ 3 ] := POS.DAT>MEM
 MOTOR2 SPOLL
 STEP CHANGE.MOTOR.VARIABLE MOTOR2.COM
 TH.B.MOVE
 BEGIN
  {CPD.TH.B} SCREEN.CLEAR 400 MSEC.DELAY
  ." ROTATING" 400 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  20 MSEC.DELAY
  32 SERIAL.POLL 32 = IF
  "MOTOR.DATA GPIB.READ THEN
  " STOPPED" "MOTOR.DATA "WITHIN
 UNTIL
 SPOLL STACK.CLEAR
 {CPD.TH.B} SCREEN.CLEAR POS.DAT [ 3 ] .
```

```
 {MAIN.MID}
 TH.B.LOCK
THEN
STACK.CLEAR
;

: 2.TH.B.MOVE
 " LOAD;" MOTOR1.COM
 " 0800SHTDN   NO  NULL  NULL  NULL;" MOTOR1.COM
 " 0810DISP   STAT BLANK BLANK BLANK;" MOTOR1.COM
 " 0820VELC  8000  0000  0000  0000;" MOTOR1.COM
 " 0830ACEL  2000  0000  0000  0000;" MOTOR1.COM
 " 0840MATH   Q2 = Q1 * -125;" MOTOR1.COM
 " 0850MOVE   Q2  0000  0000  0000;" MOTOR1.COM
 " 0860SHTDN  YES  NULL  NULL  NULL;" MOTOR1.COM
 " 0870DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 800;" MOTOR1.COM
;

: 2.TH.A.FM.MOVE
 " LOAD;" MOTOR1.COM
 " 0700SHTDN  NULL  NULL  NULL   NO;" MOTOR1.COM
 " 0710DISP  BLANK BLANK BLANK  STAT;" MOTOR1.COM
 " 0720VELC  0000  0000  0000 50000;" MOTOR1.COM
 " 0730ACEL  0000  0000  0000  4000;" MOTOR1.COM
 " 0740MATH   Q2 = Q1 * -250;" MOTOR1.COM
 " 0750MOVE  0000  0000  0000   Q2;" MOTOR1.COM
 " 0760SHTDN  NULL  NULL  NULL   YES;" MOTOR1.COM
 " 0770DONE;" MOTOR1.COM
 " *;" MOTOR1.COM
 " RUN FROM 700;" MOTOR1.COM
;

: 2.TH.B.ROTATE
 MOTOR1 SPOLL
 SET.MOTOR.VARIABLE MOTOR1.COM
 2.TH.B.MOVE
;

: CHANGE.2.TH.B
 STEP :=
 POS.MEM>DAT STEP POS.DAT [ 4 ] + POS.DAT [ 4 ] := POS.DAT>MEM
 MOTOR1 SPOLL
 STEP CHANGE.MOTOR.VARIABLE MOTOR1.COM
 2.TH.B.MOVE
 BEGIN
  {FPD.2.TH.B} SCREEN.CLEAR 400 MSEC.DELAY
  ." ROTATING" 400 MSEC.DELAY
  " REPORT;" GPIB.WRITE
  20 MSEC.DELAY
  32 SERIAL.POLL 32 = IF
  "MOTOR.DATA GPIB.READ THEN
  " STOPPED" "MOTOR.DATA "WITHIN
```

```
 UNTIL
 SPOLL STACK.CLEAR
 {FPD.2.TH.B} SCREEN.CLEAR POS.DAT [ 4 ] .
 {MAIN.MID}
;

: CHANGE.2.TH.A.FM
 STEP :=
 POS.MEM>DAT STEP POS.DAT [ 7 ] + POS.DAT [ 7 ] := POS.DAT>MEM
 STEP CHANGE.MOTOR.VARIABLE MOTOR1.COM
 2.TH.A.FM.MOVE
 BEGIN
   {FPD.2.TH.A} SCREEN.CLEAR 400 MSEC.DELAY
   ." MOVING" 400 MSEC.DELAY
   " REPORT;" GPIB.WRITE
   20 MSEC.DELAY
   32 SERIAL.POLL 32 = IF
   "MOTOR.DATA GPIB.READ THEN
   " STOPPED" "MOTOR.DATA "WITHIN
 UNTIL
 SPOLL STACK.CLEAR
 {FPD.2.TH.A} SCREEN.CLEAR POS.DAT [ 7 ] .
 {MAIN.MID}
;
```

<u>IW.WRD</u>

```
\ PROGRAMS TO CONTROL THE INCHWORM MOTORS

: IW.READ
 IW
 BEGIN
   "IW.DATA GPIB.READ
\  CR "IW.DATA "TYPE
   " >" "IW.DATA "WITHIN
 UNTIL
 STACK.CLEAR
;

: IW.COM
 IW
 13 EOS.CHARACTER
 GPIB.WRITE
 2500 MSEC.DELAY
 IW.READ
;

: IW.DISPLAY
 IW
 13 EOS.CHARACTER
 GPIB.WRITE
 2500 MSEC.DELAY
 "IW.DATA GPIB.READ
```

```
;

: IW.DISPLAY.1
 IW.DISPLAY
 {FPD.IW.B} SCREEN.CLEAR
 "IW.DATA 3 8 "SUB 44 "NUMBER 7 1 FIX.FORMAT . STACK.CLEAR
 {MAIN.MID}
 9 4 FIX.FORMAT
 IW.READ
;

: IW.DISPLAY.2
 IW.DISPLAY
 {FPD.IW.A} SCREEN.CLEAR
 "IW.DATA 3 8 "SUB 44 "NUMBER 7 1 FIX.FORMAT . STACK.CLEAR
 {MAIN.MID}
 9 4 FIX.FORMAT
 IW.READ
;

: IW.INSTRUCT
 IW
 10 EOS.CHARACTER
 GPIB.WRITE
;

: IW.PARAMETERS
 " resol 1, 0.1" IW.INSTRUCT
 " resol 2, 0.1" IW.INSTRUCT
 " speed 1, 50" IW.INSTRUCT
 " speed 2, 50" IW.INSTRUCT
 " mspeed 1, 20" IW.INSTRUCT
 " mspeed 2, 20" IW.INSTRUCT
 " ;" IW.COM
;

: IW.INIT
 CR ." INITIALIZATION OF INCHWORM MOTORS..."
 IW
 EOS.OFF
 1000 TIMEOUT
 13 ASCII" GPIB.WRITE
 2500 MSEC.DELAY
 EOS.ON
 IW.READ
 " >" "IW.DATA "WITHIN IF
  ." OPERATION SUCCESSFUL." ELSE
  ." UNSUCCESSFUL." BELL BELL BELL
 THEN
 IW.PARAMETERS
;

: IW1.STATUS
 IW
```

```
 BEGIN
  " status 1, 2" IW.INSTRUCT
  " :" IW.DISPLAY
  " on" "IW.DATA "WITHIN NOT
 WHILE
  IW.READ STACK.CLEAR
 REPEAT
 IW.READ
 " status 1, 1" IW.INSTRUCT
 " ;" IW.DISPLAY.1
;

: IW2.STATUS
 IW
 BEGIN
  " status 2, 2" IW.INSTRUCT
  " ;" IW.DISPLAY
  " on" "IW.DATA "WITHIN NOT
 WHILE
  IW.READ STACK.CLEAR
 REPEAT
 IW.READ
 " status 2, 1" IW.INSTRUCT
 " ;" IW.DISPLAY.2
;

: IW.STATUS
 IW1.STATUS
 IW2.STATUS
;

: IW.GO.HOME
 " home 1" IW.INSTRUCT
 " home 2" IW.INSTRUCT
 " ;" IW.COM
 IW.STATUS
;

: IW1.STEP
 DUP 0 < IF -1 *
  " trav 1," "." "CAT IW.INSTRUCT
  " step -1" IW.INSTRUCT
 ELSE
  " trav 1," "." "CAT IW.INSTRUCT
  " step 1" IW.INSTRUCT
 THEN
  IW1.STATUS
;

: IW2.STEP
 DUP 0 < IF -1 *
  " trav 2," "." "CAT IW.INSTRUCT
  " step -2" IW.INSTRUCT
 ELSE
```

```
  " trav 2," "." "CAT IW.INSTRUCT
  " step 2" IW.INSTRUCT
 THEN
 IW2.STATUS
;

: IW1.MVABS
  " mvabs 1," "." "CAT IW.INSTRUCT
  IW1.STATUS
;

: IW2.MVABS
  " mvabs 2," "." "CAT IW.INSTRUCT
  IW2.STATUS
;
```

MAINTCAM.WRD

```
\ PROGRAM FOR OPERATING THE CAMAC CRATE

\ REAL DIM[ 3000 ] ARRAY XX (MAINLOAD.UTL)
\ REAL DIM[ 3000 ] ARRAY YY (MAINLOAD.UTL)

INTEGER DIM[ 2 ] ARRAY []F
INTEGER DIM[ 3 ] ARRAY []FAND2
INTEGER DIM[ 2 ] ARRAY []INBUF

: CRATE1.F
   PACK []F :=
   CRATE1
   []F []GPIB.BUFFER
   ME TALKER
   CRATE1 LISTENER
   BUFFER.TALK
   UNTALK
   UNLISTEN
   CRATE1 TALKER
   UNTALK
;

: CRATE1.FAND2
   PACK []FAND2 :=
   CRATE1
   []FAND2 []GPIB.BUFFER
   ME TALKER
   CRATE1 LISTENER
   BUFFER.TALK
   UNTALK
   UNLISTEN
   CRATE1 TALKER
   UNTALK
;
```

```
: CRATE1,IN
   CRATE1
   ME LISTENER
   CRATE1 TALKER
   []INBUF []GPIB.BUFFER
   BUFFER.LISTEN
   UNTALK
   UNLISTEN
;

INTEGER DIM[ 3 ] ARRAY WARY4
INTEGER DIM[ 5 ] ARRAY WARY6
INTEGER DIM[ 5 ] ARRAY DET
INTEGER DIM[ 1 ] ARRAY WARY2
INTEGER DIM[ 1 ] ARRAY TEST1
INTEGER DIM[ 2 ] ARRAY TEST2
INTEGER SCALAR HV 0 HV :=
INTEGER SCALAR NHV
INTEGER SCALAR HV1
INTEGER SCALAR HV2
REAL SCALAR V/SEC
REAL SCALAR VCORR
\ PROGRAM TO PROVIDE MAINT. MODE GPIB-CAMAC FUNCTIONS

: MAINT.FAN
   CR ." F - Function Code ? " #INPUT WARY4 [ 1 ] :=
   CR ." A - Subaddress    ? " #INPUT WARY4 [ 2 ] :=
   CR ." N - Station Number? " #INPUT WARY4 [ 3 ] :=
   WARY4 CRATE1.F
   CR ." OK "
;

: CAMAC.INIT
   16 WARY6 [ 1 ] :=
   7 WARY6 [ 2 ] :=
   8 WARY6 [ 3 ] :=
   0 WARY6 [ 4 ] :=
   0 WARY6 [ 5 ] :=
   WARY6 CRATE1.FAND2
." INITIALIZATION OF CRATE1...
   BEGIN 127 SERIAL.POLL 3 = UNTIL
." OPERATION SUCCESSFUL." CR
;

: MAINT.FAND2
   CR ." F - Function Code ? " #INPUT WARY6 [ 1 ] :=
   CR ." A - Subaddress    ? " #INPUT WARY6 [ 2 ] :=
   CR ." N - Station Number? " #INPUT WARY6 [ 3 ] :=
   CR ." Enter Data          " #INPUT TEST1 :=
   TEST1 UNPACK
   TEST2 :=
   TEST2 [ 1 ] WARY6 [ 4 ] :=
   TEST2 [ 2 ] WARY6 [ 5 ] :=
   WARY6 CRATE1.FAND2
```

```
;

: SET.DET.HV
  16 DET [ 1 ] :=
  CR ." A - Subaddress ? " #INPUT DET [ 2 ] :=
  CR ." N - Station Number ? " #INPUT DET [ 3 ] :=
  CR ." Counter High Voltage (Volts) " #INPUT 1.05 / NHV :=
   NHV HV DO
    I TEST1 :=
    TEST1 UNPACK
    TEST2 :=
    TEST2 [ 1 ] DET [ 4 ] :=
    TEST2 [ 2 ] DET [ 5 ] :=
    DET CRATE1.FAND2
    20 MSEC.DELAY
   NHV HV > IF 1 ELSE -1 THEN +LOOP
  NHV HV :=
CR ." OK "
;

: READ.ADC                    \READS ADC CHANNEL 0
 28 WARY4 [ 1 ] :=
 0  WARY4 [ 2 ] :=
 10 WARY4 [ 3 ] :=
 WARY4 CRATE1.F
 0  WARY4 [ 1 ] :=
 0  WARY4 [ 2 ] :=
 10 WARY4 [ 3 ] :=
 WARY4 CRATE1.F
 CRATE1.IN
 []INBUF [ 1 ]
 BEGIN 127 SERIAL.POLL 3 = UNTIL
;

: READ.ADC1                   \READS ADC CHANNEL 1
 28 WARY4 [ 1 ] :=
 1  WARY4 [ 2 ] :=
 10 WARY4 [ 3 ] :=
 WARY4 CRATE1.F
 0  WARY4 [ 1 ] :=
 1  WARY4 [ 2 ] :=
 10 WARY4 [ 3 ] :=
 WARY4 CRATE1.F
 CRATE1.IN
 []INBUF [ 1 ]
 BEGIN 127 SERIAL.POLL 3 = UNTIL
;

: MAINT.F
  CR ." F - Function Code ? " #INPUT WARY2 [ 1 ] :=
  WARY2 CRATE1.F
;

ECHO.OFF
```

MENU.WRD

```
\ Program for displaying the main menu and loading operational subprograms

\ " SET ROI 2000, 5000" ORTEC.COM

REAL SCALAR BETA
INTEGER SCALAR CHARGE

: 918.COUNT
 " CLEAR" ORTEC.COM
 " START" ORTEC.COM
 0 BETA :=
 BEGIN
  50 MSEC.DELAY
  READ.ADC1 BETA + BETA :=
  BETA CHARGE 6250 * >=          \ CALIBRATION OF PRESET CHARGE (mC)
 UNTIL
 " STOP" ORTEC.COM STACK.CLEAR
 " SHOW_INTEGRAL" ORTEC.SHOW
;

: CLEAR.FKEYS
  F3 FUNCTION.KEY.DOES NOP
  F4 FUNCTION.KEY.DOES NOP
  F5 FUNCTION.KEY.DOES NOP
  F6 FUNCTION.KEY.DOES NOP
  F7 FUNCTION.KEY.DOES NOP
  F8 FUNCTION.KEY.DOES NOP
  F9 FUNCTION.KEY.DOES NOP
  F10 FUNCTION.KEY.DOES NOP
;

: CHECK.POSITION
  POS.MEM>DAT
  {CPD.2.TH.A} SCREEN.CLEAR POS.DAT [ 1 ] .
  {CPD.TH.A} SCREEN.CLEAR POS.DAT [ 2 ] .
  {CPD.TH.B} SCREEN.CLEAR POS.DAT [ 3 ] .
  {FPD.2.TH.B} SCREEN.CLEAR POS.DAT [ 4 ] .
\  IW.STATUS
  {FPD.2.TH.A} SCREEN.CLEAR POS.DAT [ 7 ] .
  {MAIN.MID} SCREEN.CLEAR CR
;

: DCS.STATUS.DISPLAY
  {MAIN.TOP} SCREEN.CLEAR
  {SCALE} SCREEN.CLEAR
." AXIS    LIFT/LOCK    COURSE POS. DATA    FINE POS. DATA    BRAGG
ANGLE "
  CR
." ------  -----------  -----------------  ---------------  ------------- "
  {AXES} SCREEN.CLEAR ." 2.TH.A " CR ." TH.A " CR ." TH.B " CR ." 2.TH.B "
```

```
  {CPD1} SCREEN.CLEAR ." (+/-360) " CR ." (+/-360) " CR ." (+/-360) "
  {FPD1} SCREEN.CLEAR ." (0-9000) " CR ." (0-9000) " CR ." (0-9000) " CR
   ." (+/-36E3)"
  {MAIN.MID}
;

: MAIN.MENU.DISPLAY
  {MAIN.BOT} SCREEN.CLEAR
  80 1 DO 95 EMIT LOOP
  35 SPACES ." MAIN MENU"
  CR        ." F4 Axis positioning"
  26 SPACES ." F5 Scan Sequence (n,n)"
  CR        ." F6 Rocking Curve (n,-n)"
  22 SPACES ." F7 One crystal Instrument"
  CR        ." F8 Set Detector High Voltage"
  17 SPACES ." F9 Align Instrument"
 DCS.STATUS.DISPLAY
;

: AXIS.POSITION
  ." Axis Position program in development." CR
;

: SCAN1
  LOAD SCAN.WRD
;

: SCAN2
  LOAD SCAN2.WRD
;

: ONE.XTAL
\  LOAD ONE-XTAL.WRD
  ." Single crystal diffraction pattern program in development. " CR
;

LOAD C:ALIGNMENT.WRD

: MAIN.MENU
  CLEAR.FKEYS
  NORMAL.DISPLAY
  MAIN.MENU.DISPLAY
  400 MSEC.DELAY
  CHECK.L/L
  CHECK.POSITION
  IW.STATUS
  F4 FUNCTION.KEY.DOES AXIS.POSITION
  F5 FUNCTION.KEY.DOES SCAN1
  F6 FUNCTION.KEY.DOES SCAN2
  F7 FUNCTION.KEY.DOES ONE.XTAL
  F8 FUNCTION.KEY.DOES SET.DET.HV
  F9 FUNCTION.KEY.DOES SET.ZERO
;
```

```
F3 FUNCTION.KEY.DOES MAIN.MENU
CR ." Hit F3 to Display the Main Menu."

ECHO.ON
```

ALIGNMENT.WRD

```
\PROGRAMS TO ALIGN THE DCS

: ZERO.AXES
\CLEAR IW'S
 0 POS.DAT [ 1 ] :=
 720 POS.DAT [ 2 ] :=
 0 POS.DAT [ 3 ] :=
 0 POS.DAT [ 4 ] :=
 0 POS.DAT [ 5 ] :=
 0 POS.DAT [ 6 ] :=
 0 POS.DAT [ 7 ] :=
POS.DAT>MEM
MAIN.MENU.DISPLAY
300 MSEC.DELAY
CHECK.L/L
CHECK.POSITION
;

: SET.ZERO
 {MAIN.MID} SCREEN.CLEAR CR
." SET THE SPECTROMETER TO ITS ZERO POSITION SUCH THAT:  " CR
."  1. THE YOKE IS PARALLEL TO THE BEAM AXIS, WITH TH.B OPPOSITE
THE BEAM" CR
."  2. TH.B CRYSTAL REFLECTS THE BEAM 180 DEG." CR
."  3. TH.A CRYSTAL REFLECTS THE BEAM 180 DEG." CR
."  4. THE DETECTOR IS OPPOSITE THE BEAM INPUT." CR
."  5. ALL TABLES SHOULD BE IN THE LOCKED POSITION." CR
."  6. INCHWORM MOTORS SHALL BE ENGAGED."
." WHEN THESE SETTINGS HAVE BEEN ACHIEVED, TYPE F10 TO ZERO THE
DCS."
F10 FUNCTION.KEY.DOES ZERO.AXES
;
ECHO.OFF
```

SCAN.WRD

```
\SCAN.WRD IS A PROGRAM FOR OPERATING THE DCS IN VARIOUS SCAN
MODES

REAL SCALAR BRAGG1
REAL SCALAR BRAGG2
REAL SCALAR BRAGG1.SEC
REAL SCALAR BRAGG2.SEC
REAL SCALAR DEL.TH
REAL SCALAR ALPHA
```

```
: SCAN.DISPLAY
 DCS.STATUS.DISPLAY
 300 MSEC.DELAY
 CHECK.L/L
 CHECK.POSITION
 IW.STATUS
 {MAIN.BOT} SCREEN.CLEAR 80 1 DO 95 EMIT LOOP
  30 SPACES ." DISPERSIVE SCAN " CR
  4 SPACES ." SCAN PARAMETERS " CR
  ." Initial Bragg angle (deg.) " CR
  ." Final Bragg angle (deg.) " CR
  ." Scan increment (sec.) "
  7 2 FIX.FORMAT
  {BRAGG0} SCREEN.CLEAR BRAGG1 .
  {BRAGGF} SCREEN.CLEAR BRAGG2 .
  {SCAN.INC} SCREEN.CLEAR DEL.TH .
 {MAIN.MID} 9 4 FIX.FORMAT SCREEN.CLEAR CR
;

: SET.INITIAL.SCAN
 POS.MEM>DAT
 4 BRAGG1 * POS.DAT [ 1 ] -
 CHANGE.2.TH.A
 -4 BRAGG1 * 360 + POS.DAT [ 2 ] -
 CHANGE.TH.A
 4 BRAGG1 * 360 - POS.DAT [ 3 ] -
 CHANGE.TH.B
 400 BRAGG1 * BRAGG1.SEC 9. / + POS.DAT [ 4 ] -
 CHANGE.2.TH.B
 -1 POS.DAT [ 7 ] * BRAGG1.SEC 10 * + CHANGE.2.TH.A.FM
 IW.GO.HOME
;

: SCAN.POS
 DELETE SCAN.DAT
 SET.INITIAL.SCAN
 BRAGG2 4 * 1 + BRAGG1 4 * DO
  BRAGG2 4 * I = IF
  BRAGG2.SEC DEL.TH / ELSE
  900. DEL.TH / THEN
  BRAGG1 4 * I = IF
  BRAGG1.SEC DEL.TH / DUP ALPHA := ELSE
  0 DUP ALPHA := THEN
  DO
    7 1 FIX.FORMAT
    -931.5 900.0 / DEL.TH * I * IW2.MVABS   \TH.A.CAL = 931.5
    7 1 FIX.FORMAT
    929.0 900.0 / DEL.TH * I * IW1.MVABS   \TH.B.CAL = 929.0
    9 4 FIX.FORMAT
    918.COUNT
    OUT>FILE SCAN.DAT
    J 4. / I DEL.TH * 3600. / + . . CR
    OUT>FILE.CLOSE
```

```
7 2 FIX.FORMAT
DEL.TH 10 * CHANGE.2.TH.A.FM
  DEL.TH 9. >= IF
    DEL.TH 9. / CHANGE.2.TH.B
  ELSE 9 DEL.TH / I ALPHA - 1 + <=
    IF 1 CHANGE.2.TH.B
      9. DEL.TH / ALPHA + ALPHA :=
    THEN
  THEN
\    STACK.CLEAR
 900 del.th / 1 - I = if
  IW.GO.HOME
  -9000 CHANGE.2.TH.A.FM
  1 CHANGE.2.TH.A
  -1 CHANGE.TH.A
  1 CHANGE.TH.B
 then
 STACK.CLEAR
 loop
 LOOP
;

\: SCAN.NEG
\ DELETE NEG-SCAN.DAT
\ SET.INITIAL.SCAN
\ -1 CHANGE.2.TH.A
\ -1 CHANGE.TH.B
\ 1 CHANGE.TH.A
\ 9000 CHANGE.2.TH.A.FM
\ BRAGG2 -4 * BRAGG1 -4 * DO
\   0 ALPHA :=
\   -900 DEL.TH / 0 DO
\     918.COUNT
\     OUT>FILE NEG-SCAN.DATA
\     J -4. / I DEL.TH * 3600 / - . . CR
\     OUT>FILE.CLOSE
\     -931.5 900. / DEL.TH * I 1 + * IW2.MVABS     \CAL=931.5
\     929.0 900. / DEL.TH * I 1 + * IW1.MVABS      \CAL=929.0
\     DEL.TH 10 * CHANGE.2.TH.A.FM
\;

 : SCAN
  0 BRAGG1 < IF
   SCAN.POS
\ ELSE
\  DEL.TH 0 < IF
\  SCAN.NEG
  THEN
\THEN
 ;

 : INPUT.SCAN.PARAMETERS
  {MAIN.MID} SCREEN.CLEAR CR
  ." Enter the initial Bragg angle in degrees (nearest 1/4 deg.) "
```

```
 #INPUT BRAGG1 := CR
." Enter the initial Bragg angle in sec. (0-900) "
 #INPUT BRAGG1.SEC := CR
." Enter the final Bragg angle in degrees (nearest 1/4 deg.) "
 #INPUT BRAGG2 := CR
." Enter the final Bragg angle in sec. (0-900) "
 #INPUT BRAGG2.SEC := CR
." Enter one of the following choices for the scan increment in seconds " CR
." 900 450 180 90 36 18 9 3.0 1.0 0.5 0.2 0.1 "
 #INPUT DEL.TH := CR
." Enter a value for preset charge (mC) " #INPUT CHARGE :=
SCAN.DISPLAY
F5 FUNCTION.KEY.DOES SCAN
SCREEN.CLEAR CR
." Hit F5 to run the DCS scan program "
STACK.CLEAR
;

ECHO.ON

INPUT.SCAN.PARAMETERS
```

<u>SCAN2.WRD</u>

```
ECHO.OFF

\ SCAN2.WRD IS A PROGRAM FOR OBTAINING DCS ROCKING CURVES

REAL SCALAR BRAGG.DEG
REAL SCALAR BRAGG.SEC
REAL SCALAR DEL.TH.RC
REAL SCALAR ROCKING.WIDTH
REAL SCALAR GAMMA

: ROCKING.CURVE.DISPLAY
 DCS.STATUS.DISPLAY
 300 MSEC.DELAY
 CHECK.L/L
 CHECK.POSITION
 {MAIN.BOT} SCREEN.CLEAR 80 1 DO 95 EMIT LOOP
   30 SPACES ." ROCKING CURVE " CR
   4 SPACES ." SCAN PARAMETERS " CR
   ." Bragg angle (deg.) " CR
   ." Bragg angle (sec.) " CR
   ." Scan increment (sec.) "
   {BRAGG0} SCREEN.CLEAR BRAGG.DEG .
   {BRAGGF} SCREEN.CLEAR BRAGG.SEC .
   {SCAN.INC} SCREEN.CLEAR DEL.TH.RC .
 {MAIN.MID} SCREEN.CLEAR CR
;

: SET.INITIAL.ROCKING.CURVE
 POS.MEM>DAT
```

```
 4 BRAGG.DEG * POS.DAT [ 1 ] -
  CHANGE.2.TH.A
 -4 BRAGG.DEG * 360 + POS.DAT [ 2 ] -
  CHANGE.TH.A
 -4 BRAGG.DEG * 359 + POS.DAT [ 3 ] -
  CHANGE.TH.B
 -400 BRAGG.DEG * BRAGG.SEC -9 / ROCKING.WIDTH -18 / + + POS.DAT [ 4 ] -
  CHANGE.2.TH.B
7 1 fix.format
 -931.5 900 / BRAGG.SEC * IW2.MVABS
7 1 fix.format
 -929.0 900 / BRAGG.SEC 900 - ROCKING.WIDTH 2 / + * IW1.MVABS
 BRAGG.SEC 10 * POS.DAT [ 7 ] -
  CHANGE.2.TH.A.FM
;

: ROCKING.CURVE
 SET.INITIAL.ROCKING.CURVE
 DELETE ROCKING.DAT
 0 GAMMA :=
 ROCKING.WIDTH DEL.TH.RC / 0 DO
  918.COUNT
  OUT>FILE ROCKING.DAT
  I DEL.TH.RC * . . CR
  OUT>FILE.CLOSE
  7 1 FIX.FORMAT
  -929.0 900 / BRAGG.SEC 900 - ROCKING.WIDTH 2 / + DEL.TH.RC I 1 + * - *
   IW1.MVABS
  9 DEL.TH.RC / I GAMMA - <= IF
   1 CHANGE.2.TH.B
   9 DEL.TH.RC / GAMMA + GAMMA :=
  THEN
 LOOP
;

:  INPUT.ROCKING.CURVE.PARAMETERS
 {MAIN.MID} SCREEN.CLEAR CR
 ." Enter the Bragg angle to the nearest 1/4 deg. " #INPUT BRAGG.DEG :=
 CR
 ." Enter the Bragg angle in sec. (0-900 sec.) " #INPUT BRAGG.SEC :=
 CR
 ." Enter the full width of the rocking curve (sec.) " #INPUT ROCKING.WIDTH :=
 CR
 ." Enter the increment of the rocking curve (sec.) " #INPUT DEL.TH.RC :=
 CR
 ." Enter the preset charge (mC) " #INPUT CHARGE :=
 CR
 ROCKING.CURVE.DISPLAY
 F6 FUNCTION.KEY.DOES ROCKING.CURVE
 SCREEN.CLEAR CR
 ." Hit F6 to run the rocking curve program "
 STACK.CLEAR
;
```

ECHO.ON

INPUT.ROCKING.CURVE.PARAMETERS

APPENDIX B

PROGRAMMING WORD LIST

Mainload.utl
: gpib.init
: init.ok
: do.init

Maintcam.wrd
: crate1.f
: crate1.fand2
: crate1.in
: maint.fan
: camac.init
: maint.fand2
: set.det.hv
: read.adc
: read.adc1
: ramp.gen
: maint.f

918.wrd
: ortec.init.chk
: ortec.chk
: ortec.prt.data
: ortec.com
: ortec.show
: 918.init

IW.wrd
: IW.read
: IW.com
: IW.display
: IW.display.1
: IW.display.2
: IW.instruct
: IW.parameters
: IW.init
: IW1.status
: IW2.status
: IW.status
: IW.go.home
: IW1:step
: IW2.step
: IW1.mvabs
: IW2.mvabs

Motor.wrd
: spoll
: motor.init
: motor1.init
: motor2.init
: motor1.com
: motor2.com
: set.motor.variable
: check.l/l.th.a

: check.l/l.th.b
: check.l/l.2.th.a
: check.l/l
: 2.th.a.shtdn
: 2.th.a.lock
: 2.th.a.startup
: 2.th.a.lift
: th.a.lift
: th.b.lift
: th.a.lock
: th.b.lock
: 2.th.a.move
: 2.th.a.rotate
: change.2.th.a
: th.a.move
: th.a.rotate
: change.th.a
: th.b.move
: th.b.rotate
: change.th.b
: 2.th.b.move
: 2.th.a.fm.move
: 2.th.b.rotate
: change.2.th.b
: change.2.th.a.fm

Read.bas
: read.bas
: array>fil

Menu.wrd
: 918.count
: clear.fkeys
: check.position
: DCS.status.display
: main.menu.display
: axis.position
: scan1
: scan2
: one-xtal
: main.menu

Detector.wrd
: set.detector.parameters
: 918.count
: ev.detector
: det1
: det2

Alignment.wrd
: zero.axes
: set.zero

Scan.wrd
: scan.display
: set.initial.scan
: scan.pos
: scan
: input.scan.parameters

Scan2.wrd

: rocking.curve.display
: set.initial.rocking.curve
: rocking.curve
: input.rocking.curve.parameters

DATE FILMED

4/7/94

END