NTIS
ornl

ORNL/TM-6342

**OAK
RIDGE
NATIONAL
LABORATORY**

**UNION
CARBIDE**

MASTER

**The ORNL 10-m Small-Angle
X-Ray Scattering Camera**

R. W. Hendricks

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

The following pages are an exact
representation of what is in the original
document folder.

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. 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-6342

Contract No. W-7405-eng-26

METALS AND CERAMICS DIVISION

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

R. W. Hendricks

Date Published: December 1979

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
DEPARTMENT OF ENERGY

EB
DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161.
NTIS price codes—Printed Copy: A11 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use or the results of such use of any information, apparatus, product or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

TABLE OF CONTENTS

	<u>Page</u>
*V.1 OVERVIEW	
ABSTRACT	1-1
INTRODUCTION	1-1
GEOMETRICAL CONSIDERATIONS	1-4
THE CAMERA	1-8
THE DETECTOR	1-13
THE COMPUTER	1-17
THE INTERFACE	1-19
SOFTWARE	1-23
SYSTEM PERFORMANCE	1-27
APPLICATIONS	1-30
CONCLUSIONS	1-40
V.2 X-RAY SOURCE	
V.3 CAMERA DESIGN	
V.4 TWO-DIMENSIONAL POSITION-SENSITIVE DETECTOR	
V.5 HIGH-SPEED DATA ACQUISITION INTERFACE HARDWARE	
V.6 MOTOR INTERFACE HARDWARE	
V.7 CAMAC INTERFACE	
V.8 COMPUTER OPERATING SYSTEM AND SYSTEM GENERATION SOFTWARE	
V.9 SYMBIONT DEVICES (OTI, E27, TEK, LPT, AND TIP)	
V.10 ON-LINE GRAPHICS	
*V.11 CAMERA CONTROL PROGRAMS (SAX, OVF, AND MOT)	
ABSTRACT	11-1
INTRODUCTION	11-1
DATA STRUCTURES	11-2
PROGRAM OVF	11-8
PROGRAM MOT	11-8
PROGRAM SAX	11-9
SAX OPERATOR COMMANDS	11-11
SYSTEM PROGRAMMING NOTES	11-38
APPENDIX A — PROGRAM OVF SOURCE LISTING	11-39
APPENDIX B — PROGRAM MOT SOURCE LISTING	11-43
APPENDIX C — PROGRAM SAX SOURCE LISTING	11-47
APPENDIX D — EXAMPLES OF THE USE OF PROGRAM SAX	11-67
APPENDIX E — PROCEDURES FOR COMPILING, LINK-EDITING, AND CATALOGUING PROGRAM SAX	11-71
V.12 MICROPROCESSOR DATA TRANSFER SOFTWARE	

	<u>Page</u>
*V.13 FILE MANAGEMENT PROGRAMS (MANAGER, FETCH AND SAVE)	
ABSTRACT	13-1
INTRODUCTION	13-1
DATA STRUCTURES	13-2
PROGRAM MANAGER	13-4
MANAGER OPERATOR COMMANDS	13-9
SUBROUTINE FETCH	13-25
SUBROUTINE SAVE	13-26
SYSTEM PROGRAMMING NOTES	13-27
APPENDIX A — SOURCE LISTING OF PROGRAM MANAGER	13-29
APPENDIX B — SOURCE LISTING OF SUBROUTINE FETCH	13-39
APPENDIX C — SOURCE LISTING OF SUBROUTINE SAVE	13-43
V.14 DATA ANALYSIS SOFTWARE	
V.15 IBM 360/91 BATCH DATA ANALYSIS PROGRAMS	
V.16 PDP-10 TIME-SHARING DATA ANALYSIS PROGRAMS	
V.17 SPECIAL SAMPLE HOLDERS	
V.18 DYNAMIC DEFORMATION DEVICE	
V.19 CONSTRUCTION OF LABORATORY BUILDING	
V.20 OPERATOR'S GUIDE	

ACKNOWLEDGMENTS

The ORNL 10-m small-angle x-ray scattering camera is a unique, state-of-the-art, computer-controlled facility and represents an investment of approximately \$500,000. It was developed by a team of approximately 30 engineers, programmers, technicians and craftsmen, all of whom contributed significantly to its outstanding success. This report, in which each chapter is authored by those individuals principally involved in a specific aspect of the instrument, provides complete hardware and software documentation for the facility. It is presented in a loose-leaf format with the pages numbered by chapter. In this way each volume may be updated from time to time as appropriate changes and/or enhancements to the system are made. As updated pages are prepared, the corrected text will be denoted by a solid bar in the left margin, as illustrated here. In the following Table of Contents, an asterisk (*) (left-hand side) indicates that the volume has been published. The remaining volumes will be published in the near future. The Table of Contents has been designed to serve as a subject index to the entire report.

Special thanks must go to the following: J. E. Weidley and R. H. Ward who designed the mechanical aspects of the camera and H. N. Elmore, W. L. Hylton, T. F. Roberts, J. C. Ogle, J. Adams, and D. E. Tumlin who constructed it; M. K. Kopp who designed the two-dimensional position-sensitive detector, W. T. Clay who supervised its construction and testing, and C. E. Fowler, Jr. for his great skill in assembling it; W. J. McClain, R. W. Napier, R. T. Roseberry, and J. W. Woody, Jr. for their role in specifying and procuring the computer; J. C. Twichell who conceived and designed the interface and motor driver hardware, C. C. Johnson whose knowledge of the ModComp CPU was instrumental in successful installation of the interface, R. G. Upton who supervised the construction and debugging of the interface components, J. A. Ramsey and R. R. Bentz who did all of the assembly of the various boards, M. Liccardo of Scientific Micro Systems who provided invaluable help with the microcontroller, and W. Strauss of Modular Computer Systems who helped design, code, and prepare the custom ROMs for the communications processor; and R. L. Stephenson, N. A. Betz, J. S. Lin, R. D. Carlson, and T. C. Tucker who wrote some of the special-purpose software. My colleagues, B. S. Borie, R. D. Carlson, J. S. Lin, C. J. Sparks, and H. L. Yakel provided continuous help through their discussions of geometrical optics of the system, monochromator design, and general aspects of the camera. Finally, such a project could not have been successful without the continuous support and encouragement of H. L. Yakel, C. J. McHargue, J. R. Weir, and A. Zucker. I am grateful to all of these individuals.

INTERNAL DISTRIBUTION

1-2. Central Research Library	62-63. J. S. Lin
3. Document Reference Section	64. C. J. McHargue
4-5. Laboratory Records Department	65. H. Postma
6. Laboratory Records, ORNL RC	66. R. L. Stephenson
7. ORNL Patent Office	67. J. O. Stiegler
8. H. R. Child	68. D. B. Trauger
9-58. R. W. Hendricks	69. J. R. Weir, Jr.
59. M. R. Hill	70. H. L. Yakel
60. S. P. King	71. A. Zucker
61. W. C. Koehler	

EXTERNAL DISTRIBUTION

- 72-73. USDOE, Division of Material Sciences, Washington, DC 20545
 L. C. Ianniello
 D. K. Stevens
74. USDOE, Oak Ridge Operations Office, P. O. Box E, Oak Ridge,
 TN 37830
 Assistant Manager, Energy Research and Development
- 75-101. Technical Information Center, Office of Information Services,
 P. O. Box 62, Oak Ridge, TN 37830

ORNL/TM-6342/V1

OAK
RIDGE
NATIONAL
LABORATORY

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

Vol. 1. Overview

R. W. Hendricks

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

1. OVERVIEW

ABSTRACT

A new small-angle x-ray scattering camera which utilizes a rotating anode x-ray source, crystal monochromatization of the incident beam, pinhole collimation, and a two-dimensional position-sensitive proportional counter has been developed. Because the sizes of the x-ray focal spot, the sample, and the resolution element of the detector are each approximately $1 \times 1 \text{ mm}^2$, the camera was designed so that the focal spot-to-sample and sample-to-detector distances may each be varied in 0.5-m increments up to 5 m to provide a system resolution in the range 0.5–4.0 mrad. A large, general-purpose specimen chamber has been provided into which a wide variety of special-purpose specimen holders can be mounted. The detector has an active area of $200 \times 200 \text{ mm}$ and has up to 200×200 resolution elements. The data are recorded in the memory of a mini-computer by a high-speed interface which uses a microprocessor to map the position of an incident photon into an absolute minicomputer memory address. With this interface, over 10^5 events/s can be recorded. The data recorded in the computer memory can be processed on-line by a variety of programs designed to enhance the user's interaction with the experiment. Among these are routines for background and detector sensitivity correction, contour and perspective plotting of the two-dimensional data, a variety of averaging schemes for determining circular averages and/or one-dimensional slices of the data, file management programs to handle the large quantity of data produced by a two-dimensional detector, and an interactive package for communications with a central computing facility via a dedicated hardwired link. At the highest angular resolution (0.4 mrad), the flux incident on the specimen is 1.0×10^6 photons/s with the x-ray source operating at 45 kV and 100 mA. The performance of the instrument is demonstrated with several examples, among which are scattering patterns from voids in neutron irradiated nickel, plastically deformed polyethylene, and collagen fibrils. Results from a kinetic study of the crystallization of polyethylene are also reported.

1. INTRODUCTION

Small-angle x-ray scattering (SAXS) has long been used to study microscopic levels of structures represented by fluctuations in electron

densities. Until recently, this technique suffered from the serious problem that the low intensity of available x-ray sources forced experiments to be performed in long slit collimation geometries in which angular resolution is good in one direction, but poor in the orthogonal direction. This limitation (1) required relatively large specimens which were often difficult to prepare, (2) resulted in a large parasitic background caused by multiple Bragg and multiple diffuse scattering, and (3) made the examination of specimens which showed anisotropic scattering patterns nearly impossible. Recent technological developments have removed these impediments. First, x-ray sources have become significantly more powerful as rotating anode and synchrotron systems have been developed; and second, detection efficiency has increased through the use of both one- and two-dimensional position-sensitive x-ray detectors to simultaneously record many data points. A review of the crystallographic and diffuse scattering applications of position-sensitive detectors for both x-rays and neutrons has been given.¹ An instrument utilizing pin-hole collimation and recovering the low intensity with a rotating anode x-ray source and a one-dimensional position-sensitive proportional counter was described by Schelten and Hendricks.² Other SAXS experiments which employ one-dimensional position-sensitive x-ray detectors have been reported by Dupont et al.;³ Hashizume et al.;⁴ Faruqi;⁵ and Podolsky, St. Onge, Yu, and Lymn.⁶ A review of the status of small-angle scattering instrumentation as of early 1978 has been given by Schelten and Hendricks.⁷

¹R. W. Hendricks, *Trans. Am. Crystallogr. Assoc.* 12: 103-46 (1976).

²J. Schelten and R. W. Hendricks, *J. Appl. Cryst.* 8: 421-9 (1975).

³Y. Dupont et al., *Nature* 238: 332-3 (London, 1972).

⁴H. Hashizume et al., *Annual Report of the Engineering Research Institute* 34: 227-34, Faculty of Engineering, University of Tokyo. [also *Jap. J. Appl. Phys.* 15: 2211-9 (1976).]

⁵A. R. Faruqi, *Proceedings Second Ispra Nuclear Electronics Symposium*, (Stresa, Italy, May 20-23, 1975), pp. 199-205.

⁶R. J. Podolsky, R. St. Onge, L. Yu, and R. W. Lymn, *Proc. Nat. Acad. Sci.* 73: 813-7 (1976).

⁷J. Schelten and R. W. Hendricks, *J. Appl. Cryst.* 11: 297-324 (1978).

Although two-dimensional position-sensitive proportional counters were developed some years ago,^{8,9} they have only recently been used for x-ray diffraction^{10,11} and neutron small-angle scattering.^{7,12-17} Application of such detectors to x-ray scattering has lagged because of their complex construction and the cost of the associated electronics and digital data acquisition system, and because their relatively poor spatial resolution requires the construction of large cameras. Despite these problems, the anticipated gain in count-rate over a one-dimensional system, and the ability to record data from anisotropic samples make the two-dimensional detector attractive.

It is the purpose of this series of reports to describe in detail our development of the ORNL 10-m SAXS camera which utilizes monochromatic radiation, pinhole collimation, and a two-dimensional position-sensitive

⁸C. J. Borkowski and M. K. Kopp, *IEEE Trans. Nucl. Sci.* NS-17: 340-9 (1970).

⁹C. J. Borkowski and M. K. Kopp, *IEEE Trans. Nucl. Sci.* NS-19: 161-8 (1972).

¹⁰C. Cork et al., *Proceedings of International Conference on Computers in Chemical Research and Education*, (Ljubljana-Zagreb, Yugoslavia, 1973).

¹¹C. Cork et al., *J. Appl. Cryst.* 7: 319-23 (1974).

¹²K. Ibel, W. Schmatz, and T. Springer, *Atomkernenergie* 17: 15-18 (1971).

¹³W. Schmatz, T. Springer, J. Schelten, and K. Ibel, *J. Appl. Cryst.* 7: 96-116 (1974).

¹⁴J. Alberi, *IEEE Trans. Nucl. Sci.* NS22: 255-68 (1975).

¹⁵K. Ibel, *J. Appl. Cryst.* 9: 296-309 (1976).

¹⁶B. P. Schoenborn et al., *J. Appl. Cryst.* 11: 455-460 (1978).

¹⁷H. R. Child and S. Spooner, *J. Appl. Crystallogr.* (in press).

detector. An overall view of the instrument is shown in Fig. 1. In this first volume, the design criteria for the instrument are outlined and an overview of the various components is given. In each section of this volume, appropriate references to other volumes of the series are given in order to provide the user with as complete a guide as possible to both the hardware and software which has been developed. Finally, this volume is concluded with a quantitative description of the performance of the system, and its use is illustrated with several examples taken from our current research in materials and polymer science and biophysics.

II. GEOMETRICAL CONSIDERATIONS

Apart from requirements imposed by the resolution desired in the new instrument, there are three physical constraints which must be considered: (1) the size of the focal spot, (2) the maximum specimen size, and (3) the size of the resolving element in the two-dimensional detector. For our applications, it has been determined that a resolution of 0.5 mrad (corresponding to a $\Delta K \approx 0.002 \text{ \AA}^{-1}$ with $\text{CuK}\alpha$ radiation) will be necessary. The x-ray focal spot of a 6-kW Rigaku-Denki rotating anode generator is $0.7 \times 7 \text{ mm}$ while for a 60-kW machine it is $1.0 \times 10 \text{ mm}$. Limited sizes of certain specimens dictated that the beam at the specimen should not exceed 1.5 mm in diameter. The two-dimensional position-sensitive detector currently in use in our laboratory has a grid of 100×100 wires spaced 2 mm apart in both the x- and y-directions. Although the electronics are capable of resolving the point of impact of an incident photon to within 1 mm (FWHM), giving a grid of 200×200 points, we have chosen to set the resolution at the Full Width Tenth Maximum point, or 2 mm, in order to eliminate the need for deconvoluting the data for the effects of electronic smearing, and to decrease the computer memory requirements for data storage and analysis.

The scattered power $P(\underline{\kappa})$ recorded in a detector resolving element at the point (x,y) may be expressed as

$$P(\underline{\kappa}) = \phi F_1 \Delta\Omega_1 t e^{-\mu t} \frac{d\Sigma}{d\Omega}(\underline{\kappa}) \Delta\Omega_2, \quad (1)$$

Fig. 1. The ORNL Small-Angle X-Ray Scattering Laboratory. The main spectrometer components are identified in Fig. 2. In the upper right are the analog signal processing electronics, computer system, operator and user consoles, and the graphics terminal. In the lower left is the one-dimensional position-sensitive detector system described by Schelten and Hendricks.²

where

$$|\underline{\kappa}| = \frac{4\pi}{\lambda} \sin\theta,$$

2θ = scattering angle,

λ = x-ray wavelength,

ϕ = apparent brilliance of the x-ray source at the first pinhole (photons/cm²/sterad/sec),

F_1 = area of the entrance slit (cm²),

$\Delta\Omega_1$ = solid angle subtended by the second pinhole (sterad),

t = specimen thickness (cm),

$e^{-\mu t}$ = specimen transmission, and μ its linear absorption coefficient (cm⁻¹),

$\frac{d\Sigma}{d\Omega}(\underline{\kappa})$ = specimen differential scattering cross section (cm²), and

$\Delta\Omega_2$ = solid angle subtended by a detector resolution element (sterad).

Although several different methods of "optimizing" a collimation system can be defined, in the present case the most effective procedure is to maximize the power scattered into a detector resolving element subject to the constraint of achieving a desired resolution. The resolution, R , of the collimation system is defined* as

$$R = \alpha + \beta, \quad (2)$$

where

α = maximum angular deviation of the incident beam, and

β = maximum angular deviation of the rays recorded in the detector resolution element.

For a collimation system with square pinholes of dimension $a_1 \times a_1$ and $a_2 \times a_2$ separated by a distance L_1 , and a detector with a resolving

*In small-angle x-ray scattering, a common definition of resolution is the minimum angle, $\epsilon_{\min}$, to which data can be recorded. For the collimation system defined here,

$$\epsilon_{\min} = (a_2 + 2\alpha L_2 + a_3)/2L_2,$$

which, from Eq. (3) may be shown to be identical to the resolution R defined in Eq. (2).

element of size $a_3 \times a_3$ separated by a distance L_2 from the specimen which is located immediately behind the second pinhole, we find

$$\alpha = (a_1 + a_2)/2L_1 \quad , \quad (3a)$$

$$\beta = (a_2 + a_3)/2L_2 \quad . \quad (3b)$$

Following Shaffer and Hendricks,¹⁸ the collimation system dependent parameters of Eq. (1) can be separated from those which depend exclusively on the sample, yielding a factor

$$P_0 = \frac{\phi a_1^2 a_2^2 a_3^2}{L_1^2 L_2^2} \quad , \quad (4)$$

which must be maximized subject to the constraint of Eq. (2). Since for this collimation system a_1 , a_2 , and a_3 are constrained, only L_1 and L_2 are adjustable in the maximization. By the method of Lagrange multipliers, the maximum in P_0 is found when

$$\alpha = \beta = R/2 \quad , \quad (5)$$

giving by Eq. (3)

$$L_1 = \frac{(a_1 + a_2)}{R} \quad , \quad (6a)$$

and

$$L_2 = \frac{(a_2 + a_3)}{R} \quad . \quad (6b)$$

To achieve a resolution $R (= \epsilon_{\min})$ of 0.5 mrad, if $a_1 = 0.7$ mm, $a_2 = 1.5$ mm, and $a_3 = 2.0$ mm, the optimum focal spot-to-specimen distance, L_1 , is 4.4 m, while the optimum specimen-to-detector distance, L_2 , is 7 m. With the incident beam centered in a 200×200 mm detector, a total angular range of 0.5 to 14 mrad is covered. Larger angular ranges, with an attendant

¹⁸L. B. Shaffer and R. W. Hendricks, *J. Appl. Cryst.* 7: 154-8 (1974).

increase in incident beam power but lower resolution, are obtained by decreasing L_1 and L_2 while maintaining constant pinhole sizes.

III. THE CAMERA

The considerations of the previous section indicate that to achieve the desired resolution under the constraints of the existing x-ray source and with the current design of the two-dimensional detector, the overall camera length must be about 10 m. It was also necessary to design the system such that both the resolution and the x-ray wavelength or method of x-ray monochromatization could be changed readily. All of these criteria are met in the design shown schematically in Fig. 2. Complete details and engineering drawings of all components discussed in this section are found in Vol. 3 of this series of reports.¹⁹ The supporting frame consists of a steel box-beam, 10-m-long and 30 × 30 cm in cross-section, into which a 60-cm vertical offset was built 50 cm from one end. This beam is pivoted at a point directly under the monochromator crystal which is mounted on the tube tower of the x-ray generator. Emerging from under the generator, the offset raises the beam above an access floor through a 1.2-m square hole. Two unistrut rails, 38 cm apart, are mounted on seven pedestals and are 104 cm above the access floor (Fig. 2, Sec. A-A). These rails extend over the top of the x-ray generator to a point under the monochromator to align the beam tubes such that they are coaxial with the incident x-ray beam.

The incident beam monochromator is a small, flat piece of pyrolytic graphite.²⁰ It is mounted on a stage which can be translated into the beam emerging at a -6° take-off angle from the x-ray generator, and can be tilted about a horizontal axis to make the diffracted beam horizontal.

The collimation system consists of two pinholes separated by distances of up to 5 m. The first is a 1-mm-diam hole in a lead sheet mounted on an

¹⁹R. W. Hendricks, R. W. Ward, J. E. Weidley, *The ORNL 10-m Small-Angle X-Ray Scattering Camera. 3. Camera Design*, ORNL/TM-6342/V 3.

²⁰C. J. Sparks, Jr., "Increased X-Ray Intensities by Use of Graphite as a Monochromator," *Metals and Ceramics Division Ann. Progr. Rep.* June 30, 1968, ORNL-4370, p. 58.

Fig. 2. Schematic Drawing of the new ORNL 10-m Small-Angle X-Ray Scattering Camera.

x-y positioning stage on the front of the beam line. The second consists of a set of four specially polished tungsten edges²¹ mounted to make a 1-mm square hole. These edges have surfaces parallel to the incident beam cut to make an included angle of 88° with the plane perpendicular to the incident beam. The polishing technique produces a radius of curvature of the edge which touches the beam of about $300 \text{ \AA}$ over a length of 2 mm. This hole is mounted on an x-y positioning stage located inside the specimen chamber. The path between the holes consists of 0.5- and 1.0-m lengths of 15-cm-diam vacuum pipes bolted to the unistrut rails.

A 125- μm mylar foil, tilted at 45° to the beam, mounted in a vacuum chamber immediately after the first hole, serves as an incident beam monitor. A scintillation detector records the incoherent scattering normal to the beam. This scattering has been demonstrated to be linear over a wide range of incident beam intensity (60:1).

The specimen chamber consists of a $30 \times 30 \times 35 \text{ cm}$ vacuum chamber containing externally operated x-y positioning devices for both the second pinhole and the specimen stage. Each of these positioners is driven by a stepping motor which may be controlled manually or via the computer. Five interchangeable plugs have been mounted in the baseplate of the chamber; each carries a 6-pin Kovar vacuum electrical feedthrough. This mechanism provides easily interchanged electrical leads for motors, solenoids, furnace power, thermocouples, and other specimen controlling or monitoring devices. The specimen stage consists of a $15 \times 15 \text{ cm}^2$ plate which can be moved $\pm 1.2 \text{ cm}$ in both the x and y directions, and onto which a variety of special-purpose specimen holders can be mounted. One such holder, a motor-driven rotary wheel, is shown in Fig. 3. This device permits 19 small specimens, mounted on interchangeable discs, to be rotated in turn into the beam under computer control. Several other special-purpose specimen holders, including liquid sample cells, furnaces for polymer crystallization kinetics, etc., are described in another volume of this series.²² In addition, we have installed a piece of glassy carbon on a sliding

²¹Prepared by Ion Tech., Ltd., of Teddington, England.

²²R. W. Hendricks et al., *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 17. *Special Sample Holders*, ORNL/TM-6342/V 17.

Fig. 3. Specimen Chamber. Shown are (a) slit and specimen stage positioning micrometers, (b) slit holder, (c) glassy carbon mount, (d) specimen translation stage, and (e) rotary specimen wheel.

mechanism mounted on the second pinhole stage. By use of a stepping motor equipped with an eccentric cam, the glassy carbon can be reproducibly moved into the incident beam (Fig. 3). We periodically measure the intense glassy carbon scattering as a second check on the incident beam intensity, and we measure its attenuation to determine the transmission coefficient of the specimen.²³

The flight path to the detector consists of 0.5- and 1.0-m lengths of 20- and 35-cm-diam vacuum pipes. Scattered radiation emerges from this beam line through a 30-cm-diam 0.5-mm-thick beryllium window. Since a beryllium sheet of such diameter and thickness will not support 1 atm, a rigid square grid of aluminum ribs, 8-mm-deep $\times$ 1.0-mm-thick and 40 mm apart and which is 96% transparent, is mounted on its vacuum side. The window is clamped between two "O" rings to provide sufficient radial motion to compensate for the 4-mm sag which occurs when the beam pipe is evacuated.

The detector, weighing approximately 25 kg, is mounted on an x-y-z positioning stage adapted from a small table-top milling machine. The entire mechanism is mounted on a small dolly which can travel the length of the unistrut rails. The x-y motions (perpendicular to the incident beam) are controlled by stepping motors, while the z-motion (parallel to the incident beam) is exclusively under manual control to prevent accidental penetration of the beam stop through the beryllium windows. The incident beam stop is mounted on a thin finger in the center of the 20 $\times$ 20 cm active area of the detector. The detector is positioned with the beam stop just touching the beryllium window on the flight path.

With this arrangement the resolution of the spectrometer is easily changed by inserting or removing sections of beam line and moving the detector to a new position immediately behind the exit window. To decrease the time necessary to evacuate the system after changing specimens, the specimen chamber may be isolated from the collimation and scattering beam lines by electropneumatic gate valves. Each of the three sections is separately pumped to a vacuum of $<10 \mu\text{m Hg}$.

²³This procedure is usually valid because the scattering from most samples contributes insignificantly to the intense total scattering from the glassy carbon.

IV. THE DETECTOR

The detector is a two-dimensional position-sensitive proportional counter developed by Borkowski and Kopp.⁹ Complete engineering and electronic details are given in Vol. 4 of this series of reports.²⁴ Physically, it is 40 cm in diameter and 10 cm deep, with an active area of 20×20 cm. Photons pass through a 20×20 cm sheet of 0.5-mm-thick beryllium into a 6-cm linear drift region. The drift field is developed with two parallel grids of 75- μ m wires. The wires in each grid are parallel to each other and are 5 mm apart while the grids are separated by 12 mm parallel to the beam axis. The drift region increases the photon absorption probability and maintains the electron drift in a direction perpendicular to the detector plane.

The detection portion of the chamber consists of two cathodes 12 mm apart with an anode centered between them. The cathode wires are 75 μ m Nichrome supported on bronze spring fingers by epoxy joints that electrically isolate the wires from the springs. The anode wires are 12- μ m stainless steel soldered to similar spring fingers. The springs are designed to keep sufficient tension on the wires to prevent them from sagging, but not enough to plastically deform them. The fill gas is 1 atm of Ar-5% CO₂, which is approximately 90% efficient for 8-keV x rays.

The detector signals are processed by the rise-time method.²⁵ A block diagram of the circuit is shown in Fig. 4. The sum of the pulses from the four corners of the detector is equivalent to the pulse created in a normal proportional counter, and contains both energy and absolute rise-time information. In the circuit of Fig. 4, energy analysis eliminates pulses which result from higher harmonics (λ/n) diffracted by the monochromator. Absolute rise-time analysis also reduces spurious counts resulting from partial ionization tracks of the correct energy that may be caused by gamma rays emitted by radioactive samples. The output pulse from the energy discriminator is delayed and then tested for coincidence

²⁴M. K. Kopp et al., *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 4. *Two-Dimensional Position-Sensitive Detector*, ORNL/TM-6342/V 4.

²⁵C. J. Borkowski and M. K. Kopp, *Rev. Sci. Instr.* 39: 1515-22 (1968).

Fig. 4. Block Diagram of Detector Electronics. Abbreviations are: ADC — analog to digital converter; TSCA — timing single channel analyzer; A — nuclear pulse amplifier; COINC — coincidence unit; TAC — time to amplitude converter; SCA — single channel analyzer; LED — leading edge discriminator; GDC — gate and delay generator.

with the output pulse from the rise-time analysis circuit. The coincidence pulse is used to activate a gate and delay generator which, in turn, is used as the gate input to both the x- and y-axis time to amplitude (TACs) converters. On receipt of a photon of both the correct energy and rise time, the digitized x- and y-addresses appear on the two ADC outputs for further processing by a digital computer.

We have verified that in the operating range of our detector, the gas amplification factor follows the Diethorn relationship.²⁶ Thus, a semi-logarithmic plot of $\ln A$ vs V , where A is the amplitude of the sum pulse and V is the cathode bias voltage, is linear. Two separate curves have been determined: one for CuK_α (8.05 keV) x rays and one for MnK_α (5.89 keV) x rays emitted by the radioisotope ^{55}Fe . With these curves, it is a simple matter to count either energy without any adjustments to the position, rise-time, or energy signal processing electronics; only the detector bias need be adjusted to accurately position the MnK_α signal in the CuK_α energy window.

We have mounted a 1 mCi ^{55}Fe source in a vacuum-tight holder with a 0.125-mm beryllium window. This source, when placed in the specimen position 5 m from the detector, produces an integral count rate of 300 s^{-1} . The spatial distribution of this radiation (see Fig. 5) is a measure of the non-uniformity in sensitivity of the detector, as well as the shadowing effect of the aluminum grid, which supports the exit window, and the beam stop with its supporting arm. All scattering patterns are corrected for this non-uniformity by dividing by the normalized sensitivity pattern. The x- and y-axis linearity and channel-to-position conversion factors can also be obtained from the data of Fig. 5 by determining the spacing (in channels) of the shadows of the known positions of the beryllium window support grids. We have found the detector to be linear in both x- and y-directions, and to be free from pincushion or barrel distortions within the accuracy of our measurements (1%). By adjusting the analog signal processing electronics appropriately, we can achieve the same sensitivity (mm/channel) in each direction.

²⁶W. Diethorn, *A Methane Proportional Counter System for Natural Radiocarbon Measurements*, U.S. AEC Report NYO-6628 (1956).

Fig. 5. Perspective Plot of the Detector Sensitivity Function Obtained by Recording the Uniform Radiation Emitted by 1 mCi ^{55}Fe Placed at the Specimen Position. The integrated count rate with a source-to-detector distance of 5 m is 300 cps. Note the grooves in the pattern resulting from the aluminum supporting grid.

V. THE COMPUTER

There are several considerations in selecting an on-line computer system for the collection, manipulation, and display of the data recorded by the two-dimensional position-sensitive detector. Such a system should have (1) at least a 16-bit word in order that a minimum of 65,536 events can be recorded per channel before overflow occurs; (2) a 65-K memory which allows sufficient core for 16 K of data, a powerful real-time operating system, and a reasonable complement of one- and two-dimensional graphics display programs; (3) a hardware floating point processor to insure adequate computational speed for rapid graphical display of the data; (4) a large-screen graphics display terminal; (5) a minimum of 2.5 M words of on-line disc storage for retaining the operating system, programs, and data; (6) appropriate hardware to support a 9600 bps bisynchronous communications link to a large central computer to allow for remote processing of data with programs which are too large to run on-line, and to allow access to large-capacity disc packs for additional data storage capacity; (7) several general-purpose interface boards for rapid interfacing to detector electronics and stepping motor drivers; and (8) a full complement of software which includes a real-time operating system, an assembler, a FORTRAN compiler, a text editor, a link-editor, and a debugging program plus other diagnostic routines.

A Modular Computer Systems²⁷ ModComp 11/220 CP computing system was selected as the most cost-effective machine for our needs. The configuration is shown in Fig. 6. A unique feature of this machine is its modular bus architecture for movement of data between the central processing unit (CPU) and memory, and the possibility to add custom hardware instructions which utilize the modular bus. This aspect plays an important role in the detector interface to be discussed in the next section. A Tektronix Model 4014-1 Graphics Terminal with a 48-cm display screen was selected as the graphics display system because (1) it is a storage display terminal as opposed to a refreshing type scope, thus requiring the graphical results to be computed only once instead of repetitively; (2) it is RS-232C

²⁷Modular Computer Systems, Inc., 1650 McNab Road, Ft. Lauderdale, Florida 33309.

Fig. 6. Block Diagram of the Organization of the ModComp II Computer with Custom Macroinstructions.

compatible with any computer and (3) it has a full complement of FORTRAN driver programs, thus facilitating the development of specialized plotting routines required by the SAXS experiments.

Details of the computer hardware and the operating system configuration are found in Vol. 8 of this series of reports,²⁸ while details of other hardware and software components will be reviewed in the following sections of this report.

²⁸R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera. 8. Computer Operating System and System Generation Software*, ORNL/TM-6342/V 8.

VI. THE INTERFACE

Interfacing the digitized x-y position of the incident photon as determined by the two ADCs in Fig. 4 to the digital computer system can be accomplished in many ways, and depends strongly on the desired data rate and the architecture of the computer. The interface must accomplish two functions: (1) the variable number of bits of information obtained from each ADC must be converted into a single 16-bit word which is the absolute address of the corresponding location in the computer memory; and (2) the contents of that address must then be incremented by one and tested for overflow. The simplest possible interface which achieves these objectives connects the ADCs directly to the general-purpose interface board, using the ModComp interrupt hardware to initiate necessary software routines. Preliminary tests indicate that with such an interface 100% of the CPU time would be devoted to storing data when the data rate reached 50,000 events/s, leaving no time for the desired graphical display of the data. Since data rates of as high as 500,000 events/s could be envisioned for the instrument, this type of interface was rejected.

More sophisticated, and much higher speed, data acquisition systems have been described by Cork et al.^{10,11} and Klesse and Kostorz²⁹ in which the mapping is performed by external hardware, and the data are stored in memory external to the CPU. We have developed an interface in which each of the two major functions are also performed by separate high-speed hardware. However, unlike other systems, we use the minicomputer memory, rather than external memory, for data storage. A block diagram of the interface is shown in Fig. 7. The address mapping procedure is handled by a Scientific Micro Systems³⁰ microcontroller, which is a fast, bit-string-oriented microcomputer. The microcontroller is connected to the two detector ADCs through a high-speed buffer, and to the ModComp II through three Model 4801 general-purpose controller (GPC) interface boards. The

²⁹R. Klesse and G. Kostorz, *Crystallographic Computing* (F. R. Ahmed, K. Huml, B. Sedlecek, eds.), Munksgaard, Copenhagen (1976).

³⁰Scientific Micro Systems, Inc., 520 Clyde Ave., Mountain View, California 94043.

Fig. 7. Block Diagram of the Fast ADC Computer Interface, Scaler/Timer Interface, and Motor Driver Interface.

ADC buffer is used because our current Northern Scientific³¹ TN 1245 ADCs are Wilkinson ramp-type devices (200 MHz clock; 1.2 μ s mean conversion time), and it is necessary to synchronize the x- and y-axis data before passing them on the microcontroller. The hardware aspects of the interface are described in detail in Vol. 5 of this series of reports.³²

³¹Tracor Northern, Inc., Middleton, Wisconsin 53562.

³²J. C. Twicrell, R. G. Upton, C. C. Johnson, R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera. 5. High-Speed Data Acquisition Interface Hardware*, ORNL/TM-6342/V 5.

The operation of the interface is as follows. At the start of an experiment the computer operator downloads the origin and dimensions of the data storage array into the microcontroller. This information is used by the microcontroller to determine the absolute memory location into which an event is stored in the ModComp. The microcontroller program is designed to operate on a three-level priority basis; first, it tests to see if there are ADC data ready to read and, if so, assembles the 16-bit absolute address and places it on a last-in first-out stack; secondly, it tests the I/O lines on the ModComp general-purpose interface to see if the computer is ready to receive data and, if so, fetches an address from the stack and places it on sixteen parallel output lines; and finally, it tests if the run is complete. With this procedure, in excess of 100,000 events/s can be processed. Extension to higher rates is easily accomplished by using several microprocessors in parallel. It is to be noted that all user interaction with the microcontroller is handled by directives issued from program SAX³³ which communicates with the microcontroller code³⁴ in a handshaking mode.

The second major function, that of incrementing the appropriate memory location and storing the address in a separate buffer in the event of an overflow, was solved by installing a custom I/O macroinstruction in the computer mainframe. This has been done by recoding the read-only memory chips in the standard ModComp Communications Processor (CP). We have developed two instructions. In the first, on receipt of a high-priority interrupt from the microprocessor through the general-purpose interface board, a special source identification is recognized and complete control of the machine, through the Modular Bus Control, is passed to the CP. The recoded hardware instruction reads the memory address to be incremented from the GPC, fetches the contents of that address, increments it by one, and stores it back in memory. It also tests for overflow and, if true, stores the address in a stack for later processing. Control is then returned to the central processor. This instruction loop has an execution time of 3.8 μ s and permits data rates up to about 250 kHz. The

³³R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*.
11. *Camera Control Programs (SAX, OVF and MOT)*, ORNL/TM-6342/V 11.

³⁴R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*.
12. *Microprocessor Data Transfer Software*, ORNL/TM-6342/V 12.

second instruction is similar to the first, except that on receipt of an interrupt from the microcontroller, a block transfer of n addresses is made, thereby reducing the mean execution time to $1.8 \mu\text{s}/\text{event}$. With this instruction, over 550,000 events/s could be managed by the system, including overflow handling. Complete details of this interface are given elsewhere.^{32,35,36}

It is important to note that these custom instructions do not use any of the CPUs general-purpose registers, thus eliminating context switching during data acquisition. This leaves the computer free for system control, graphics display, and other necessary computations, or even unrelated background computations such as program editing and compilation, all of which may be performed simultaneously with data acquisition. At 100,000 events/s, the apparent machine speed for other tasks is degraded by only 25%.

A powerful feature of this data acquisition scheme, apart from its speed, is its generality. The custom hardware instructions in the ModComp CP are oblivious to the data array size and format; these are determined exclusively by software in the microprocessor. It must be recalled that the minimum conversion gain of our ADCs is 8 bits (256 channels); no increase in conversion rate can be achieved by selecting smaller conversion gains. A data array 256×256 (65 k) is clearly too large for either our machine or reasonable data handling. We have developed programs to acquire data in a variety of arrays,³⁴ including 64×64 and 128×128 arrays which are accomplished by dropping the least significant bits, a high-resolution "zoom" which selects a 64×64 region of interest from a 128×128 array, and high-resolution rectangular arrays in which the central horizontal strip of 32 or 16 channels is taken from a 128 or 256 channel y-axis to produce 32×128 and 16×256 arrays. In such procedures, which are especially useful for scattering patterns which show one-dimensional orientation effects, the detector's electronic resolution element is still square. There is no compression of data along the y-axis into a few low-resolution

³⁵J. C. Twichell and R. W. Hendricks, *Proceedings of the Second Annual ModComp Users' Exchange*, Modular Computer Systems, Inc., Ft. Lauderdale, Florida (December 6-8, 1976).

³⁶J. C. Twichell and R. W. Hendricks, *Nucl. Inst. Methods* (in preparation).

channels. Rather, the y-axis resolution remains equal to that along the x-axis and data outside the central strip are discarded.

Finally, high-speed dynamic studies can be implemented with the interface developed here. A sequence of ModComp data array origins can be stored in the microprocessor registers, and, with simple software, a new origin can be selected for address mapping in less than 2 μ s. The time at which the origin is changed can be triggered by a pulse from an external reference which is generated, for example, by a field applied to the specimen. Through programming, it would be possible to dedicate as much as 32K of the ModComp memory to data acquisition, thus allowing the simultaneous presence of eight 64×64 data arrays. The instrument is thus uniquely suited to a wide variety of dynamic experiments, such as the stretching of muscle, the rotation of non-spherical molecules in solutions in the presence of electric or magnetic fields, and composition fluctuations in binary alloys during oscillatory temperature fluctuations.

VII. SOFTWARE

The instrument described in the preceding sections requires a significant complement of software to be as efficient and as versatile as possible for the user. The special software which has been developed for this system falls broadly into five categories: operating system modifications; the data acquisition control program; the data base file management system; data plotting and three-dimensional visualization programs; and data analysis routines. Each of these is described briefly in the following paragraphs.

No modifications to the manufacturer-supplied Modular Applications Executive MAX III operating system were required. In all cases, our changes involved only straightforward additions of special interrupt handlers to process data and service interrupts from the general-purpose interface controllers as described in the last section. We have added simple custom interrupt handlers for motor limit switches and for beam monitor scaler/timer devices, and an interrupt-driven memory increment handler which serves as a backup to the custom microcoded CP instruction.

Details are found in the volume on the operating system and system generation software.³⁷

Several symbiont device programs have also been added to perform in-core format changing, to allow paging on the Tektronix graphics terminal, to allow the user to access remote high-speed line printers at the central computing facility as if they were local logical devices, to perform binary synchronous communications with the remote batch and time-sharing computers, and to allow terminal-to-terminal communications. These programs are described in Vol. 9 of this report.³⁸

All aspects of data acquisition and instrument control are the responsibility of two high-priority foreground tasks called OVF and SAX. Task OVF performs two functions. First, it loads the absolute starting address of a 400-word overflow buffer array into a dedicated memory location. Second, it tests (every 10 ms) the 390th word of the array to determine if it has been used and, if so, halts data acquisition until the buffer can be transferred to disc and the pointer reset. Otherwise, it relinquishes the CPU to SAX or other low-priority tasks. In this manner, the critical overflow address buffer is examined regularly, independent of the activity of all other programs simultaneously in execution. Without such control, at high data rates the custom hardware instruction could overwrite all of core (including the operating system) with overflow addresses. This task is automatically loaded and started into execution by SAX, and is completely transparent to the user until the buffer is full. Further details may be found in Vol. 11 of this report.³⁹

SAX is an interactive program which communicates with the user in simple English. Its various directives provide complete control of the camera functions among which are (a) select address mapping routine in the microprocessor; (b) start and stop data acquisition; (c) zero data

³⁷R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 8. *Operating System and System Generation Software*, ORNL/TM-6342/V 8.

³⁸N. A. Batz and R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 9. *Symbiont Devices (OTI, E27, TEK, LPT, and TIP)*, ORNL/TM-6342/V 9.

³⁹R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 11. *Camera Control Programs (SAX, OVF, and MOT)*, ORNL/TM-6342/V 11.

array; (d) save data on disk; (e) move motors; (f) enable and disable various interrupts; and (g) print experiment status reports. They may be entered from a user's console or, for automatic experiments where many specimens are to be run (e.g., with the specimen wheel) or for moderate speed dynamic experiments where a transient effect is to be followed (e.g., the crystallization of a polymer), the directive sequences can be written in advance and read from a disc file. In this case, it is essential to prevent the program from reading a new directive until after the current directive has been completed. The interrupt enable/disable directives can be used to select real time, a preset time or counts on the scalars, or a specific motor interrupt to cause the desired delay. No further directives will be read until the specified interrupt occurs. Complete instructions for the use of the camera control programs are found elsewhere in this series.³⁹

Program SAX performs no services in the area of data base management or data display and analysis. These functions are all accomplished at the lowest priority in a background core partition, where they are executed under control of the standard system program JOBC (job control). Without forethought, the quantity of data generated by a two-dimensional detector can rapidly increase to unmanageable levels. To alleviate this problem, a FORTRAN direct access data base management system has been developed in which independent experiments with variable data array sizes can be stored on disk, fetched from disk back to core for further analysis, deleted, and otherwise manipulated.⁴⁰ In addition, because of the limited storage available for the data base on the ModComp disk (480,000 words), we have also duplicated the file management scheme for execution on a central batch computer, where an entire disk pack (5.6×10^6 words) has been dedicated to data array storage; and on the central time-sharing PDP-10 computer where an on-line data base is maintained. Software has been written to allow individual experiments to be moved bi-directionally from one data base to the other. Thus, we can keep approximately one hundred 64×64

⁴⁰R. L. Stephenson and R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 13. *File Management Programs (MANAGER, FETCH and SAVE)*, ORNL/TM-6342/V 13.

data arrays on the ModComp system, while over thirteen hundred can be kept on the disc at the central facility.

The two-dimensional position-sensitive detector records so large a quantity of data that graphical analysis is imperative to judge the quality of an experiment prior to detailed quantitative analysis. We have developed a FORTRAN interface package which allows all plotting on the Tektronix graphics terminal to be coded with standard CALCOMP subroutines.⁴¹ Included in this package are subroutines NUMBER and SUMBOL (routines for plotting numbers and a wide variety of symbols in any orientation) that are not normally in Tektronix/Calcomp preview packages. To produce publishable quality hardcopy output, we have elected to use the hardwired link to the central facility where several CALCOMP plotters are available. To this end, we developed an efficient spooling program which saves all of the calling arguments to the screen graphics, transmits them to the central facility, and recreates the plot seen on the local screen. With this software, it has been straightforward to carry over to the ModComp machine some powerful plotting packages developed for the central facility. Among these are PERSPEX, a three-dimensional isometric and perspective plotting program,⁴² CR4CON, a two-dimensional isointensity contour plotting program,⁴³ and INTRIGUE, a general-purpose plotting package designed to make linear, logarithmic, and semi-logarithmic plots.⁴⁴

Finally, we have developed a series of background overlay tasks for the quantitative analysis of data.⁴⁵ Among these are CIRAVE, a circular averaging routine for producing intensity versus scattering angle data for isotropic scatterers, and ANGLE, a routine which computes the intensity

⁴¹R. W. Hendricks, J. S. Lin, R. L. Stephenson, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 10. *On-Line Graphics*, ORNL/TM-6342/V 10.

⁴²T. C. Tucker, *CATCH: Computer Assisted Topography, Cartography, and Hypsography Part III*, *PERSPX: A Subroutine Package for Perspective and Isometric Drawings*, ORNL-TM-3790 (February 1973).

⁴³T. C. Tucker, private communication (1975).

⁴⁴M. B. Emmett, *INTRIGUE-II: An IBM-360 Subroutine Package for Making Linear, Logarithmic, and Semi-Logarithmic Graphs Using the CALCOMP Plotter*, ORNL-4664 (March 1971).

⁴⁵J. S. Lin and R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 14. *Data Analysis Software*, ORNL/TM-6342/V 14.

along a slice of variable width at arbitrary azimuthal angles. These two programs accurately take into account intersections of circles (represented by regular polyhedra of 16 or more sides) or azimuthal slices with the finite detector resolution elements and partition counts within the elements. Data analysis codes have been developed that interactively compute the radius of gyration, Porod constant, and integrated intensity.

The significant feature of the computing system described here is its ability to accommodate not only a sophisticated, high-speed data acquisition system, but also simultaneous, on-line analyses of the results. In cases where too many data are collected for on-line analyses, it is simple to duplicate the codes and perform the analyses in batch mode on the central facility machines.

VIII. SYSTEM PERFORMANCE

Several parameters may be used to describe the performance of the new small-angle scattering camera. Among these are the incident beam intensity, electronic noise, parasitic slit-edge scattering, dead time, and stability. The incident beam intensity has been measured by the multiple foil method, with appropriate attention to the problem of non-uniform foil thickness described by Weinberg⁴⁶ and Shaffer and Hendricks.⁴⁷ At the highest angular resolution of the incident beam, 0.4×0.4 mrad, the measured power through the collimation system is $(1.00 \pm 0.05) \times 10^6$ photons/s when the x-ray source is operated at 45 kV and 100 mA. If the source were operated at 60 kV and 100 mA, the power would be approximately 2×10^6 photons/s. The scaling to other collimation geometries is easily computed from the equations of Sect. II. As a comparison, we consider the incident beam fluxes currently being reported from small-angle x-ray scattering instruments constructed at the synchrotron sources DESY in Germany and SPEAR in the USA (Table 1). Such comparisons must be made with caution. For the data of Table 1, neither of the synchrotron sources were operating under optimum conditions for the production of synchrotron light. Further,

⁴⁶D. L. Weinberg, *Rev. Sci. Instrum.* 34: 691-6 (1963).

⁴⁷L. B. Shaffer and R. W. Hendricks, *J. Appl. Cryst.* 7: 159-63 (1974).

Table 1. A Comparison of Various Small-Angle X-Ray Scattering Cameras

Property	Small-Angle Scattering Camera		
	ORNL	DESY ⁴⁸	SSRP ⁴⁹
Source	Rotating Anode	Synchrotron	Storage Ring
Operating Conditions	45 kV, 100 mA	6.5 GeV, 3.1 mA	3. GeV, 10 mA
Collimation	Graphite + Pinholes	Mirrors + Crystals	Mirrors + Crystals
Sample Size	1 × 1 mm ²	0.2 × 0.2 mm ²	0.5 × 0.5 mm ²
Angular Resolution (mrad)	0.4	3.5	3-4
Flux on Sample (photons/s)	1.0 × 10 ⁶	20 × 10 ⁶	500 × 10 ⁶
Flux on Sample Scaled to 3.5 mrad angular resolution (photons/s)	80 × 10 ⁶	20 × 10 ⁶	500 × 10 ⁶
Detector	Two-dimensional position-sensitive proportional counter	Film ⁵⁰	Film ⁵¹
Detector Gain (film = 1)	10 ¹ - 10 ³	1	1

⁴⁸A. Harmsen, R. Leberman, and G. E. Scholz, *J. Mol. Biol.* 104: 311-14 (1976).

⁴⁹N. G. Webb, S. Samsen, R. M. Stroud, R. C. Gamble, and J. D. Baldeschweiler, *J. Appl. Crystallogr.* 10: 104-110 (1977).

⁵⁰An area detector is being designed for this system.

⁵¹A linear position-sensitive proportional counter is being tested.

synchrotron radiation is considerably better collimated in the vertical direction than the angular resolution selected for operation of the small-angle facilities. Thus, they would scale more nearly linearly if the angular resolution were improved. Because it is not yet clear exactly what the scaling is, and because the angular resolution of the systems reported by Barrington-Leigh and Rosenbaum⁵² and Webb et al.⁴⁹ are not easily converted to higher resolution, we have chosen to scale the power of our system to the resolution selected at synchrotron sources. It is clear from these comparisons that, especially if a 60-kW x-ray generator should be available, conventional x-ray sources can still be highly competitive for small-angle scattering applications.

The electronic noise of our detector, measured with the x-ray source turned off, is approximately 2 cps. This background is almost uniformly distributed over the detector, with only a small increase near the edges. The total electronic noise plus parasitic slit-edge scattering is 80 cps, with most of the parasitic scattering occurring in a cross-shaped pattern (reflecting the square symmetry of the pinhole) in the vicinity of the incident beam. We have found that a significant portion of this intensity is higher order harmonics diffracted by the monochromator that are not eliminated by our electronics.

We have determined the dead time of the entire detection system by the method described by Chipman,⁵³ and found it to be 15 μ s. This is significantly longer than the 7 μ s estimated in Sect. VI, and is not yet understood. We anticipate reducing this value to <5 μ sec with new hardware shift registers operating under the control of the microprocessor.

Special precautions have been taken to assure the stability of the system. A stabilizing transformer has been added to the input mains of the x-ray source. Constant temperature, closed circuit water is used to cool the anode. The original low-capacity vacuum system on the generator has been replaced with a large, trapped diffusion pump. Filament lifetime

⁵²J. Barrington-Leigh, G. J. Rosenbaum, *J. Appl. Cryst.* 7: 117-21.

⁵³D. R. Chipman, *Acta Cryst.* A25: 209-14 (1969).

is now consistently over 4000 h. These modifications, as well as x-ray source operating procedures, are presented in Vol. 2 of this report.⁵⁴ Following several years of adverse experiences with fluctuating room temperatures and humidity, and with the presence of dust and dirt, a major renovation to install insulation, a hung ceiling, and a computer room air handling system was accomplished in 1978. Details concerning the structural, electrical, and air handling aspects of the laboratory are given in Vol. 19 of this report.⁵⁵

IX. APPLICATIONS

Our SAXS program is involved in a broad spectrum of materials science and biophysical structural research. The performance of the camera described in the previous sections will be illustrated with a few selected examples from our work.

A problem of continuing importance in the development of fission and fusion reactors is the adverse effect of neutron-irradiation-induced swelling of structural components. Small-angle x-ray and neutron scattering have been demonstrated to be powerful tools for the evaluation of the size distribution and total volume fraction (swelling) of voids created by neutron irradiation.⁵⁶⁻⁶⁰ As a part of our program to correlate the swelling induced by neutron and heavy-ion damage, we are determining void properties in neutron-irradiated nickel as a function of irradiation temperature and fluence. A typical scattering pattern, measured from a foil

⁵⁴J. S. Lin and R. W. Hendricks, *The ORNL 10-meter Small-Angle X-Ray Scattering Camera*. 2. X-Ray Source, ORNL/TM-6342/V2.

⁵⁵R. W. Hendricks, *The ORNL 10-meter Small-Angle X-Ray Scattering Camera*. 19. Construction of Laboratory Building, ORNL/TM-6342/V 19.

⁵⁶H. E. Kissinger, *J. Nucl. Mater.* 28: 118-9 (1968).

⁵⁷G. L. Kulcinski, B. Mastel, H. E. Kissinger, *Acta Metall.* 19: 27-36 (1971).

⁵⁸J. E. Epperson, R. W. Hendricks, K. Farrell, *Phil. Mag.* 30: 803-18 (1974).

⁵⁹R. W. Hendricks, J. Schelten, W. Schmatz, *Phil. Mag.* 30: 819-37 (1974).

⁶⁰R. W. Hendricks, J. Schelten, G. Lippmann, *Phil. Mag.* 36: 907-21 (1977).

only 20 μm thick, is shown in Fig. 8a. Because this pattern shows cylindrical symmetry, we have obtained the circularly averaged intensity versus scattering angle curve shown in Fig. 8b using the program CIRAVE described in Sect. VII. From these data and the absolute intensity calibration factor, we have determined the radius of gyration, $R_g = 51.4 \text{ \AA}$, and the swelling, $\frac{\Delta V}{V} = 0.61\%$. Through the use of the multiple specimen wheel, such data can be obtained rapidly and accurately as a complement to companion transmission microscopy studies. Further details will be presented elsewhere.⁶¹

There is widespread interest in the kinetics of crystallization of polymeric materials. Until recently, it has been very difficult to obtain dynamic structural information during crystallization because of the long times required to record a small-angle scattering pattern with traditional apparatus. For example, Wilkes and Emerson⁶² have approached the problem in two ways: first, they made measurements at low temperatures where the kinetics of crystallization were slow compared to their data recording speed; and secondly, at faster kinetics they recorded data at only a few fixed angles and attempted to infer the scattering curve from these data. The first truly dynamic experiment was performed by Schultz⁶³ who used a Kratky camera equipped with a one-dimensional position-sensitive detector to record the scattering curves from two polymers. With Schultz, we have extended this work using the high-speed data acquisition capabilities of the 10-m camera to further investigate the kinetics of crystallization of polyethylene. Two coaxial resistance furnaces were constructed to fit on the specimen stage shown in Fig. 3. Tantalum reflectors were used as heat shields. One furnace was mounted such that the incident beam passed through the center of the specimen with no interference from either the coils or heat shields. The specimen, mounted in a small steel block and with a thermocouple attached to it, could be moved from one furnace to the second by activating an electric solenoid. Because of the speed with

⁶¹ J. S. Lin and R. W. Hendricks, *A Small-Angle X-Ray Scattering Study of Voids in Neutron-Irradiated Nickel* (in preparation).

⁶² G. L. Wilkes and J. A. Emerson, *J. Appl. Phys.* 47: 4261-64 (1976).

⁶³ J. M. Schultz, *J. Poly. Sci.: Poly. Physics Ed.* 14: 2291-2311 (1976).

Fig. 8(a). Isointensity Contour Map of Polycrystalline Nickel Foil Thinned to $41\ \mu\text{m}$ by acid jet Polishing. Material was irradiated in the Oak Ridge Research Reactor (ORR) at 350°C to a fast fluence ($E > 0.1\ \text{MeV}$) of $0.6 \times 10^{21}\ \text{n/cm}^2$. Measurement time was 3000 s at 45 kV and 30 mA (only 17% of full power). The incident beam resolution was 0.4 mrad. (b) Circular average of the data in (a).

which data were collected, all necessary SAX directives were read from a disc file. Depending on the isothermal crystallization temperature, data were recorded for 30 to 200 s in time intervals from 50 to 600 s. In all, some 250 patterns were recorded at seven different temperatures. Preliminary examination of results was performed during the course of an experiment by displaying the isointensity contour maps, and by logging a status report which recorded data such as the total number of photons counted, the clock time when the run was completed, and other run identification data. Following this series of experiments, the data were transferred to a disc pack at the central computing facility for final analysis in which each curve was corrected for background, detector sensitivity, and the spherical average was computed. Typical results for the 118°C run are shown in Fig. 9, while the integrated intensity (which is related to the degree of crystallinity) is shown in Fig. 10. It is significant that eight experiments could be performed during the 400 s which elapsed between the start and completion of the crystallization process. Complete details of this investigation have been presented elsewhere.⁶⁴

The results presented thus far are from isotropic scatterers where all final information can be obtained from the spherically averaged plot. We turn now to anisotropic scattering, where the new camera is producing information heretofore obtained only with difficulty. As an example, some results from our investigation of the mechanism of elastic-plastic deformation of polyethylene are presented.⁶⁵ Small pieces of polyethylene, approximately $15 \times 3 \times 1$ mm, were mounted in a yoke and were stretched various amounts by adjusting a lead screw. The entire mechanism was mounted in the camera specimen chamber with the deformation direction parallel to the horizontal x-axis. Typical results are shown in Fig. 11. In the earliest stages [11(a)–(c)], the patterns show elliptical symmetry, with eccentricity increasing with deformation. We found that the peak

⁶⁴J. M. Schultz, J. S. Lin, and R. W. Hendricks, *J. Appl. Cryst.* 11: 551–57 (1978).

⁶⁵S. K. Baczek, Ph.D. Dissertation; University of Massachusetts (1977).

Fig. 9. Circular Averages of the Isotropic Scattering Data Obtained from Polyethylene Quenched to $T = 118^{\circ}\text{C}$ ($T_c = 141^{\circ}\text{C}$). Scattering patterns were recorded for 30 s in 50-s intervals, and were stored on disc during the run. The data have been corrected for parasitic scattering and detector nonuniformity. For $t < 353$ s, the scattering curves were essentially identical to the one at 353 s, while for $t > 604$ s, they were identical to the one at 604 s. All SAX directives for camera control were read from a disc file during the course of the run in which some 34 patterns were recorded in 1500 s. The data were recorded at 45 kV and 60 mA with an incident beam resolution of 0.4 mrad.

ORNL-DWG 77-5214

Fig. 10. The Absolute Integrated Intensity, Q_0 , for the Polyethylene Crystallization Run at 118°C . Data points were computed from the curves shown in Fig. 9. The first 300 s were recorded while the specimen was cooling from the melt temperature of 190°C .

Fig. 11. Isointensity Curves of Stretched Polyethylene (Stress Applied Parallel to X-Axis). (a) Unstretched; (b) stretched 10%, (c) stretched 20%; (d) stretched 30%; (e) stretched 40%; (f) stretched 50%; (g) stretched 60%, and (h) stretched 83%. Anisotropic data such as these can be correlated with the affine model of plastic deformation. Measurement time was 3000 s at 45 kV and 30 mA for each pattern.

position, obtained from one-dimensional intensity slices taken at various azimuthal angles, is consistent with the affine model of polymer deformation up to 40% stretch. The integrated intensity is also a parameter of significance in this study. For intensity patterns which show elliptical symmetry, it can be shown⁶⁶ that

$$Q_0 = 4\pi \frac{k_3}{k_2^2} (k_2^2 \sin^2 \psi + \cos^2 \psi)^{3/2} \int_0^\infty h^2 I(h) dh, \quad (7)$$

where $k_2 = b/a$, $k_3 = c/a$, (a,b,c) are the principal axes of the ellipsoid, and ψ is the azimuthal angle ($\psi = 0$ along the y -axis) at which the one-dimensional intensity slice is recorded. For the data of Fig. 11, in which $k_3 = 1$, we have found Q_0 to be essentially independent of plastic deformation up to 60%. This result suggests that no voids are being created during this stage of deformation; otherwise Q_0 should increase. From the results obtained from voids created in nickel by neutron irradiation, it was seen that the instrument is very sensitive to extremely small volume fractions of such microporosity.

Further plastic deformation followed by heat treatment results in the classical "two-point" pattern as shown in Fig. 12. For the first time it is possible to obtain quantitative intensity data for such patterns.

As a final example, a typical result from our continuing investigation of duck tendon collagen fibrils⁶⁷ is shown in Fig. 13. A powerful feature of the interactive graphics terminal are cursors that allow one to determine (x,y) coordinates of any feature displayed. With known channel-to-angle conversion factors, positions of the various Bragg maxima can be obtained within a few seconds. In the course of this study, we have found that the width of the Bragg reflections varies significantly with specimen condition and handling. Although the data collection rate is not significantly improved over that with the one-dimensional system, we have found that variations in width are large enough that with many specimens not all of the integrated intensity could be recorded with a linear detector, thus leading to erroneous results.

⁶⁶R. W. Hendricks (unpublished research).

⁶⁷R. H. Stinson, T. W. Bartlett, T. Kurg, and R. W. Hendricks, *Biophysical J.* 26: 209-222 (1979).

Fig. 12. Scattering Patterns from Plastically Deformed Polyethylene.
(a) Isointensity contour map showing classical "two-point" pattern;
(b) three-dimensional perspective plot of the data shown in (a). The central spike is incompletely corrected parasitic scattering which outlines the direct beam. Measurement time was 3000 s at 45 kV and 30 mA. Incident beam resolution was 0.4 mrad.

Fig. 13. Isointensity Scattering Pattern from Collagen Fibers (Obtained from Duck Tendons) Stained with Uranyl Acetate and Phosphotungstic acid. The first seven orders are clearly resolved. Higher orders can be recorded by decreasing L_2 , the specimen-to-detector distance, and using the "zoom" data recording microprocessor data transfer routines. Integrated intensities under the diffraction maxima have been obtained by summing the data in a band the width of the peaks. Measurement time was 400 s at 45 kV and 60 mA, with an incident beam resolution of 0.4 mrad.

X. CONCLUSIONS

The new ORNL 10-m small-angle x-ray scattering camera is a powerful instrument for structural research. Like our previously reported system which utilizes a one-dimensional position-sensitive detector, the new system records the true sample scattering cross section $\frac{d\Sigma}{d\Omega}$ without slit smearing effects. In addition, it records in a single pattern an entire plane in reciprocal space, thus making it easy to observe even small deviations from angularly isotropic scattering. On-line interactive graphics display and data analysis have significantly improved the efficiency of research. Finally, through careful collimation system optimization, fluxes comparable to those observed in small-angle scattering cameras constructed at synchrotron sources have been achieved at comparable angular resolution with a 6-kW rotating anode source.

ORNL/TM-6342/V11

OAK
RIDGE
NATIONAL
LABORATORY

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

Vol. 11. Camera Control Programs (SAX, OVF, and MOT)

R. W. Hendricks

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

11. CAMERA CONTROL PROGRAMS (SAX, OVF, AND MOT)

R. W. Hendricks

ABSTRACT

SAX and its associated programs OVF and MOT are high-priority, pre-queued, nonresident foreground tasks which run under the ModComp II MAX III operating system to provide complete user control of the ORNL 10-m small-angle x-ray scattering camera.

1. INTRODUCTION

The ORNL 10-m small-angle x-ray scattering camera is a sophisticated analytical instrument which consists of a rotating anode x-ray generator, a hot-pressed pyrolytic graphite incident beam monochromator, up to a 10-m vacuum flight path (5-m x-ray source-to-specimen and 5-m specimen-to-detector distances), a 20×20 cm² two-dimensional position-sensitive x-ray detector, and a high-speed microprocessor controlled data digitization and transfer interface to a MODCOMP II 16-bit minicomputer. In addition, the system includes several stepping motors for slit alignment and specimen movement, and a series of scalers and timers to record external devices such as incident beam monitors and the total integrated intensity incident upon the detector. Complete details of this camera have been given elsewhere¹ and in other reports of this documentation series.

This report describes the control programs SAX, OVF, and MOT which are used to control the operation and data acquisition procedures of the camera. They provide no capability for data analysis; these are performed at low priority in the background by special data analysis programs

¹R. W. Hendricks, *J. Appl. Crystallogr.* **11**, 15-30 (1978) and *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 1. Overview, ORNL/TM-6342/V 1.

which are described in another report of this series.² To fully understand the capabilities and features of the programs described here, the reader should first study the Computer Operating System and System Generation documentation described in Vol. 8 of this series.³ References to the appropriate hardware documentation will be made as required for an understanding of this report.

II. DATA STRUCTURES

Data gathered by the hardware/software control system are contained in various labeled common blocks. These commons are assigned to dedicated regions of memory by the GCOM or global common statement in the linkage editor (EDIT),⁴ and the memory is reserved for them by the GLOBAL statement during system generation.³ Four global commons are used, as indicated in Table 1. Two of these, the HEADER and DATA arrays, are used by the file management system⁵ to store experimental data in disc files for future retrieval and analysis.

The internal structure of each common is as follows.

1. OVER

NOVRF	JBUFF(200)	KBUFF(200)
-------	------------	------------

OVER is an overflow buffer which contains, sequentially, the absolute memory addresses of locations in the data array DATA which have overflowed (i.e., counted beyond #FFFF or 65,535). The overflow addresses are stored in OVER automatically by the hardware through an indirect

²J. S. Lin and R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 14. *Data Analysis Software*, ORNL/TM-6342/V. 14.

³R. W. Hendricks, *Ibid.*, 8. *Computer Operating System and System Generation*.

⁴MAX II/III *System Processors*, Chap. 11, The Link Editor, Reference Manual 210-600500-011 (1977), Modular Computer Systems, Inc., Ft. Lauderdale, Florida.

⁵R. L. Stephenson and R. W. Hendricks, *Ibid.*, 13. *File Management Programs (MANAGER, FETCH AND SAVE)*.

Table 1. Global Common Arrays

Label	Origin ⁶	Size	Function
OVER	#101	#191 (401)	Overflow address buffer
MOTR	#292	#20 (32)	Motor positions and speeds
HEADER	#2B2	#64 (100)	Data header record
DATA	#EFC	#1004 (4100)	Detector data array

addressing mode as described elsewhere.⁷ The address of the next vacant overflow location is obtained by the hardware from the contents of location #6F. The contents of #6F are automatically incremented by the hardware after each overflow. NOVRF is a flag which contains the number of overflow buffers which have been used, and JBUFF and KBUFF are dual 200-word buffers which contain the overflow addresses.

2. MOTR

MOTR is a 32-word common which contains information about the eight motors which are currently installed in the system.⁸ Its structure is

CPOS(8)	NPOS(8)	IRATE(8)	JTEST(8)
---------	---------	----------	----------

CPOS contain the current position (in motor steps) of the eight motors; NPOS their new (or desired position); IRATE, the rate (step/s) at which they move between positions; and JTEST, the flag conditions.

3. HEADER

Figure 1 depicts a 100-word record (one disc sector); it is the header record for all experimental data sets and contains all the necessary information to completely define an experiment. It is divided into three sections — 31 words of integer constants, 22 words of floating point

⁶Numbers preceded by # are *hexadecimal*; numbers not preceded by # are *decimal*.

⁷J. C. Twichell et al., *The ORNL 10-m Small Angle X-Ray Scattering Camera*. 5. *High-Speed Data Acquisition Interface Hardware*, ORNL/TM-6342/V5.

⁸R. G. Upton, J. C. Twichell, and R. W. Hendricks, *Ibid.*, 6. *Motor Interface Hardware*, ORNL/TM-6342/V6.

RESERVED					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ISEQ	LENGTH	OWNER	ISITE	O	NDA	NOVF	NCORR	NID	IORG	M	N	I1	I2	J1	J2	NU								ITIME(6)			NSUB	NBAK	
RESERVED					26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
NSNS	NMPT	O	O	O	O									SCALR(5)							XCENT	YCENT		O.		O.			
RESERVED					51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
O.	TRAN		ABSINT		WAVE		THICK		DIST 1		DIST 2		RO		R1		DXCHAN		DYCHAN		RHOP								
RESERVED					76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
PLOT TITLE AND COMMENTS																													

Fig. 11.1. Data Structure for HEADER.

constants,⁹ and 25 words (50 characters) of alphanumeric constants. All words in the array are initialized; integer constants to 0, the first ten floating point constants to 0., and the remaining twelve to 1., and the alphanumeric constants to blank (#2020). With these initialization values, additional parameters may be defined at a later time and by appropriately locating them in the array, previously acquired data can still be processed with programs containing the new parameter definitions. The variables which are currently defined are given in Table 2.

In addition to the four common blocks described in Table 1, a six-word array (labeled LIST) has been set aside within the operating system for use in handling I/O and interrupts from the two custom interfaces and the LASL CAMAC crate controller. This array is contained within the custom initialization routine M31N1 as described in the System Generation manual of this series of reports.³ The absolute address of this array is stored in memory location #7F. LIST (the first word) contains the last command to the custom system control interface, the second is normally 0 and is set to 1 when there is an interrupt from the scaler/timer arrays, the third contains flag bits indicating which of eight motors is turned on, while the fourth word is normally 0 and is set to 1 when one of the motors hits a limit switch. It is important to note that the M3ORTEC and M3MOTOR device handlers³ allow only one interrupt to be recorded in LIST +1 and LIST +3 at a time. Each time there is an interrupt from either device, the service interrupt (SI) for that device is disabled. Further details are given in the System Generation manual. The fifth word contains the last CAMAC control function issued to the CAMAC interface, while the sixth word is used to decode CAMAC LAMs. Further details are given in the System Generation manual and in the CAMAC manual.

4. DATA

DATA is a 4100-word array which contains the data generated by the two-dimensional position-sensitive detector. As described in Vol. 13, the file management system⁵ requires the data array size to be multiples

⁹Floating point words are 32 bits and thus occupy double words.

Table 2. Structure of HEADER

Location	Name	Default Value	Function
1	ISEQ	0	Experiment sequence number
2	LENGTH	0	Number of records required to store data set.
3	OWNER	0	CAN-coded integer ¹⁰ representing three ASCII characters for OWNER of data set.
4	ISITE	0	An integer indicating instrument on which data were taken (=0 for 10-m SAXS facility).
6	NDATA	0	Number of records for raw data set.
7	NOVF	0	Number of overflow records.
8	NCORR	0	Number of records for corrected data set.
9	NID	0	
10	IORG	#F000	Origin of IDATA global common.
11	M	0	X-axis dimension of data array.
12	N	0	Y-axis dimension of data array.
13	I1	0	X-axis lower limit for data of interest. ¹¹
14	I2	0	X-axis upper limit for data of interest. ¹¹
15	J1	0	Y-axis lower limit for data of interest. ¹¹
16	J2	0	Y-axis upper limit for data of interest. ¹¹
17	NU	0	Number of scaler/timer units in daisy-chain connected to ORTEC interface.
18	ITIME	0	A six-word array containing the time and date the experiment was performed.
24	NSUB	0	Microprocessor subroutine used for data acquisition.
25	NBACK	0	Sequence number of background data set.
26	NSNS	0	Sequence number of sensitivity data set.
27	NMPT	0	Sequence number of empty camera data set.
32	SCALER	0.	A five-word (real) array containing the contents of the scalers and timers.
42	XCENT	0.	X-axis coordinate of center of pattern.
44	YCENT	0.	Y-axis coordinate of center of pattern.
52	TRAN	1.	Transmission coefficient sample.
54	ABSINT	1.	Absolute intensity calibration factor.

Table 2 (continued)

Location	Name	Default Value	Function
56	WAVE	1.	X-ray wavelength
58	THICK	1.	Sample thickness (mm)
60	DISTI	1.	Source-to-sample distance (mm)
62	DISTZ	1.	Sample-to-detector distance (mm)
64	RO	1.	Size of first slit (mm)
66	RI	1.	Size of second slit (mm)
68	DXCHAN	1.	X-axis channel-to-distance conversion factor (mm/channel)
70	DYCHAN	1.	Y-axis channel-to-distance conversion factor (mm/channel)
72	RHOP	1.	Difference in electron density for two-phase systems (electrons/Ås).
76-100	TITLE	blank	Plot title and comments

¹⁰ MAX II/III Operating system Reference Manual, Publication No. 210-600303-000100, pp. 2-124, Modular Computer Systems, Inc., Ft. Lauderdale, Florida.

¹¹ These parameters are determined by a post data acquisition analysis program and are set at its execution time. They are set to the default values by SAX.

of a disc sector (100 words/sector); hence, for 4096 data points an array of 4100 words has been set aside. Because of the potential need for larger data arrays (e.g., 128×128) and because of limitations of the operating system, DATA is allocated space in an Unprivileged Global Common in upper core.³ Further, because of the manner in which the microprocessor code is written, the data array should begin at an absolute memory address whose last byte is zero (e.g., #F000). In the core of 64×64 data arrays the first four locations of DATA (#EFFF-#EFFF) are used to store -1, NSUB, M, and N, respectively (see Table 2).

III. PROGRAM OVF

This program is a high-priority (level 5) prequeued, nonresident foreground task. It functions as a watchdog over the overflow global common OVER to ensure that the data acquisition hardware does not store overflow addresses beyond the end of the buffer. This task, a listing of which is given in Appendix A, is loaded and set into execution by a REX (Request Executive service) call from the main control program SAX when SAX is loaded; it is completely transparent to the user.

When OVF is loaded, it types a message to the computer operator (on device C0) indicating it is loaded and in execution. It immediately tests JBUFF(1) to see if there has been an overflow. If not, it relinquishes time to lower priority tasks for one clock cycle (10 ms) and then checks JBUFF(1) again. When JBUFF(1) is modified, OVF sets NOVRF = 1 and then checks KBUFF(190) for non-zero values every 10 ms. When KBUFF(190) becomes non-zero, OVF halts data acquisition and types the message

OVERFLOW BUFFER FULL
TO RESUME, CLEAR BUFFER
THEN TYPE /OVF/R

The overflow buffers can be cleared by executing the directive ZERO in SAX (described in Sect. V).

When data rates become very much higher than they currently are, all of the hooks have been provided to allow OVF to double buffer the overflow buffers to disc, two sectors at a time. Such modifications will be provided when the need arises. OVF is removed from core by the KILL command described in Sect. VI.

IV. PROGRAM MOT

Occasionally, motors are placed in environments in which physical damage can occur if the motor power is left on too long. At present, motor 7 is mounted in the specimen vacuum chamber in order to raise and lower the glassy carbon standard out of or into the incident beam.

(This standard is used to periodically measure the stability of the machine and to determine the transmission coefficient of samples.) Without air cooling, the motor overheats and is damaged if the power is left on for more than several minutes. MOT is a high-priority (level 6) foreground task which is used as a watchdog to prevent such situations. MOT is loaded by SAX and types out a message to the operator to indicate its successful load and execution. Its operation is completely transparent to the user until a motor is shut off. It tests bit 14 of LIST +2 of the M31N1 block described above every 30 s to determine if the power to motor 7 is on. If so, it sets a countdown clock for 2½ min. If motor 7 has not been shut off in that interval, MOT issues a command to the motor interface to shut it off and announces its action to the operator via the CO device. MOT is removed from core by the KILL command described below. A source listing is given in Appendix B.

V. PROGRAM SAX

SAX is a high-priority (level 7) prequeued, nonresident task whose function is to control the operation of the ORNL 10-m small-angle x-ray scattering camera. It communicates with the user through a series of descriptive mnemonic commands which are described in detail in the next section. It provides no services in the area of data base management or data analysis and evaluation; these functions are performed at low priority in the background through a series of programs described in Vols. 13 and 14 of this series of reports.

SAX is designed to effectively utilize only a small amount of core, and thus operates as a multisegment overlay. The overlay structure is shown in Fig. 2. A source listing is given in Appendix C.

The task utilizes three different logical devices for I/O as indicated in Table 3. Device AI3 is the keyboard of the Lear-Siegler ADM-3A CRT user console, while A03 is its CRT screen. This device operates as an RS-232C asynchronous terminal at 9600 bps (960 characters/s). Device LP2 is one of the system spoolers and is used to spool a copy of all conversations on A03/AI3 to a printing device for permanent records.

Fig. 2. SAX Overlay Structure.

Table 3. SAX Logical Device Assignments

Logical Unit	Default Physical Device Assignment	Use
5	A03	Input user commands
6	A13	Write instructions to user
7	LP2	Spool hardcopy of commands, instructions, and information to hardcopy printer.

The user may reassign any or all of the logical devices to any other appropriate physical device, either during a run or at the initial loading of SAX by entering the appropriate MAX III operating system directives on the computer operator console. Various annotated examples are presented in Appendix D.

Although SAX, MOT, and OVF are usually left in core in an idle state, on occasion background processing tasks may require additional core which can be obtained only by removing the idle tasks from the foreground pool. This is accomplished by killing SAX via its command KILL described in the next section. KILL is coded to also abort and de-establish OVF and MOT before killing SAX.

VI. SAX OPERATOR COMMANDS

Program SAX is controlled by a series of primary commands. These are listed in Table 4. Some primary commands require secondary information, for which the user is provided the required input data format. When SAX is activated, as described in the previous section, it types (on *lu* = 6) an initial message to announce that it is operating, the current program version, the time and date, and then

CURRENT SEQUENCE NUMBER = 0
ENTER NEW SEQ. NO. (16)

At this point the user should enter the next sequential experiment number as determined from the system logbook. SAX then types out

SELECT FUNCTION

The user responds with any valid function described in Table 4. As for all user programs described in this series of reports, the mnemonic FNC will cause a list of the allowable commands, or functions, to be printed.

Table 4. Summary of SAX Commands

COMMAND ¹²	PURPOSE
<u>DELAY</u>	Prevent queueing of next read until event
<u>DISABLE</u>	Disable service interrupt from device
<u>DWN</u>	Lower glassy carbon into incident beam
<u>ENABLE</u>	Enable service interrupt from device
<u>FNC</u>	List allowable functions
<u>IDENT</u>	Enter experiment identification data
<u>INITIALIZE</u>	Initialize microprocessor and set up HEADER record
<u>INCREMENT</u>	Increment experiment sequence number
<u>KILL</u>	Exit from SAX and remove it from CPU queue
<u>LOOP</u>	Repeat previous functions
<u>LU</u>	Set logical unit for data storage
<u>MOT</u>	Run motors
<u>NLOOP</u>	Set number of times previous functions should loop
<u>PRINT</u>	Store contents of scaler/timers in HEADER; print contents
<u>RESUME</u>	Resume data acquisition; do not clear scaler/timer
<u>RUN</u>	Resume data acquisition; clear scaler/timer
<u>SAVE</u>	Save data array and HEADER record on disc
<u>SEQUENCE</u>	Initialize experiment sequence number
<u>STATUS</u>	Display camera and experiment status
<u>STOP</u>	Halt data acquisition
<u>UP</u>	Raise glassy carbon out of incident beam
<u>WAIT</u>	Suspend SAX task indefinitely
<u>ZERO</u>	Clear HEADER, DATA, and OVERFLOW buffers

¹²The command may be abbreviated to the underlined characters. All commands must be terminated by a carriage return.

Table 5(a). User Commands for Program SAX

<u>DELAY</u>	
FUNCTION:	Prevents SAX from queueing a read on $\ell u = 5$ until an event occurs by executing a REX, #1E (relinquish until bit set). The devices which can cause the delay are the scaler/timers, a motor limit switch, the computer clock, or a manual restart from the operator's console.
SAX RESPONSE:	WHICH DEVICE CAUSES DELAY?
USER RESPONSE:	<u>ORTEC</u> — for scaler/timers <u>MANUAL</u> — for operator manual restart <u>MOTOR</u> — for motors <u>TIME</u> — for system clock
SAX RESPONSE:	<u>ORTEC</u> — None <u>MANUAL</u> — SAX IN WAIT STATE <u>MOTOR</u> — None <u>TIME</u> — ENTER TIME DELAY IN MINUTES AND SECONDS (215)
USER RESPONSE:	<u>ORTEC</u> — None <u>MANUAL</u> — Hit console interrupt and enter /SAX/R when ready to resume <u>MOTOR</u> — None <u>TIME</u> — --xxx---yy, where xxx is the delay time in minutes and yy is in seconds
ERRORS:	<u>ORTEC</u> and <u>MOTOR</u> cannot be entered unless their respective service interrupts have been set (see <u>ENABLE</u> and <u>DISABLE</u>). An SI set is indicated by bit 2 set in word 1 or word 3 of the special M31N1 array described in Sect. 11. If the user selects <u>ORTEC</u> or <u>MOTOR</u> before enabling the appropriate SI, SAX responds with SI HAS NOT BEEN ENABLED FOR DEVICE and returns to SELECT FUNCTION.
COMMENT:	SAX will be inactive while the requested delay is in progress, and no user communications with the program is possible. On completion of the delay, SELECT FUNCTION will appear on $\ell u = 6$ and SAX has resumed execution.

Table 5(b). User Commands for Program SAX

<u>DISABLE</u>	
FUNCTION:	To reset Service Interrupt (SI) for scaler/timer or motor interrupts. This is accomplished by clearing bit 2 of word 1 or word 3 of the M3INI array to zero.
SAX RESPONSE:	<i>WHICH DEVICE IS TO BE DISABLED?</i>
USER RESPONSE:	$\left\{ \begin{array}{c} \text{MOTOR} \\ \text{ORTEC} \end{array} \right\}$
SAX RESPONSE:	<i>DEVICE $\left\{ \begin{array}{c} \text{MOT} \\ \text{ORT} \end{array} \right\}$ DISABLED</i>
ERRORS:	None

Table 5(c). User Commands for Program SAX

<u>DWN</u>	
FUNCTION:	To lower glassy carbon standard into the beam.
METHOD:	The power to motor 7 is turned on and pulses are output to the motor. The lower limit switch is tested after each pulse to see if it was reached, and, when so, the power to the motor is turned off.
SAX RESPONSE:	GLASSY CARBON IN BEAM
USER RESPONSE:	None.
ERRORS:	None.

Table 5(d). User Commands for Program SAX

<u>ENABLE</u>	
FUNCTION:	To enable the Service Interrupt (SI) for scaler/timer or motor interrupts. This is accomplished by setting bit 2 of word 1 or word 3 of the M3INI array.
SAX RESPONSE:	WHICH DEVICE IS TO BE ENABLED?
USER RESPONSE:	$\left\{ \begin{array}{l} \text{MOTOR} \\ \text{ORTEC} \end{array} \right\}$
SAX RESPONSE:	DEVICE $\left\{ \begin{array}{l} \text{MOT} \\ \text{ORT} \end{array} \right\}$ ENABLED.
ERRORS:	None.

Table 5(e). User Commands for Program SAX

<u>FNC</u>	
FUNCTION:	To list the primary camera operation functions on the lu = 6 output device.
SAX RESPONSE:	A listing of the functions, followed by <i>DO YOU WANT HARDCOPY OF THE INPUT COMMANDS?</i>
USER RESPONSE:	$\left\{ \begin{array}{c} \underline{\text{YES}} \\ \underline{\text{NO}} \end{array} \right\}$
COMMENTS:	The default is <i>NO</i> hardcopy. If <i>YES</i> is selected, copies of all I/O commands and responses will be spooled to the system <i>SPOOLER</i> symbiont device <i>S</i> , whose output can be routed to any printing device. Instructions for operating <i>S</i> are found elsewhere. ¹³

¹³MAX II/III Input/Output Reference Manual. Chap. 12, *Imaginary Printer Spooling Symbiont*, Publication 210-600501-000I00, Modular Computer Systems, Inc., Ft. Lauderdale, Florida.

Table 5(f). User Commands for Program SAX

<u>IDENT</u>	
FUNCTION:	To provide specimen-dependent information to <i>HEADER</i> regarding the experiment for permanent storage with the intensity array.
SAX RESPONSE:	ENTER OWNER ID
USER RESPONSE:	The user must provide his three-character job-name assigned by the facility director (e.g., RWH).
SAX RESPONSE:	ENTER TRANSMISSION COEFFICIENT (F5.0)
USER RESPONSE:	The sample transmission coefficient, which must be measured prior to entering <u>IDENT</u> , should be entered.
SAX RESPONSE:	ENTER ABSOLUTE INTENSITY CALIBRATION CONSTANT (x.xxxEyy)
USER RESPONSE:	The incident flux on the sample (photons/s), which must be determined prior to entering <u>IDENT</u> , should be entered.
SAX RESPONSE:	ENTER TITLE AND COMMENTS (25A2)
USER RESPONSE:	An identifying title and/or comments should be entered. The first 25 characters will be used as plot titles in the data analysis routines.

Table 5(g). User Commands for Program SAX

<u>INITIALIZE</u>	
FUNCTION:	To set up the microprocessor data acquisition routine, and initialize the HEADER record.
SAX RESPONSE:	ENTER ORIGIN OF DATA ARRAY (HEX). LAST BYTE MUST BE ZERO.
USER RESPONSE:	Currently, all data are acquired in an array which begins at #F000.
SAX RESPONSE:	SELECT MICROPROCESSOR DATA TRANSFER ROUTINE (12)
USER RESPONSE: ¹⁴	01 - ADC Test Program 02 - 64 × 64 Direct data transfer 03 - 64 × 64 Transfer with stack 04 - 128 × 128 Zoom (stores central 64 × 64 portion of pattern) 05 - 16 × 256 06 - 32 × 128 07 - 64 × 64 Interrupt driven (same as 02, but bypasses custom I/O Macro and increments memory with M3ADCXFR software routine at interrupt priority #7 (ref. 3)).
SAX RESPONSE:	MICROPROCESSOR INITIALIZED ENTER X-RAY WAVELENGTH (x.xxxx)
USER RESPONSE:	Provide x-ray wavelength (in angstroms)
SAX RESPONSE:	HOW MANY SCALERS AND TIMERS? (11)
USER RESPONSE:	Enter number of scalers and timers in ORTEC daisy chain. Currently, NU = 3. SCALER array is currently dimensioned 5
SAX RESPONSE:	ENTER SPECIMEN TO DETECTOR DISTANCE (IN CHANNEL UNITS) (F8.3)
USER RESPONSE:	This number is the ratio of the specimen-to-detector distance (in mm) divided by the detector spacial calibration constant (in mm/channel).

¹⁴R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*.
 12. *Microprocessor Data Transfer Software*, ORNL/TM-6342/V 12.

Table 5(h). User Commands for Program SAX

<u>INCREMENT</u>	
FUNCTION:	To increment the experiment sequence number, <i>ISEQ</i> .
SAX RESPONSE:	NEW SEQUENCE NUMBER IS <i>xx</i> .
USER RESPONSE:	None.
NOTE:	SAVE automatically increments <i>ISEQ</i> . See also <i>SEQ</i> .

Table 5(i). User Commands for Program SAX

<u>KILL</u>	
FUNCTION:	To shut down SAX gracefully, properly abort OVF and MOT, and then dequeue SAX.
SAX RESPONSE:	Halt Microprocessor.
OS RESPONSE:	!ABORT (END 7148) /OVF/OVF BY SAX !ABORT (END 3909) /MOT/MOT BY SAX

Table 5(j). User Commands for Program SAX

<u>LOOP</u>	
FUNCTION:	To allow rereading of input commands from $lu = 5$.
SAX RESPONSE:	HOW MANY STATEMENTS DO WE BACK UP (I2)?
USER RESPONSE:	Enter the number of input commands which are to be reread. In counting the statements, include the <u>LOOP</u> statement.
COMMENT:	$lu = 5$ must be assigned to a rewindable device (disc partition). The commands may be written to the disc with the SOURCE EDITOR ¹⁵ using the NO-COMPRESS option.

¹⁵MAX II/III System Processors, Chap. 18, *Source Editor*, Reference Manual 210-600500-018 (1977), Modular Computer Systems, Inc., Ft. Lauderdale, Florida.

Table 5(k). User Commands for Program SAX

<u>LU</u>	
FUNCTION:	To select a different data base for the storage of experimental data.
SAX RESPONSE:	CURRENT LU = xx ENTER NEW LU (12)
USER RESPONSE:	Enter a new logical unit.
COMMENT:	This unit must be assigned to a rewindable FORTRAN direct access I/O device (disc partition). If it is not cataloged in the system RAD file, it must be assigned to task SAX by the Operator Directive /SAX/ASS <i>lu</i> = <i>dpn</i> , where <i>lu</i> is the logical unit number and <i>dpn</i> is a disc partition name. One of the scratch partitions is usually used. <i>lu</i> must have been included in a DEFINE FILE statement in SAX at the time it was compiled.
NOTE:	USERS <u>ARE NOT</u> ENCOURAGED TO USE THIS FEATURE EXCEPT IN AN EMERGENCY.

Table 5(1). User Commands for Program SAX

<u>MOT</u>																																					
FUNCTION:	To allow computer control of the various stepping motors.																																				
SAX RESPONSE:	SELECT MOTOR FUNCTION																																				
USER RESPONSE:	<div><div>DISPLAY</div><div>EXIT</div><div>FNC</div><div>MOV</div><div>OFF</div><div>ON</div><div>POS</div><div>RATE</div></div>																																				
<u>DISPLAY</u>	To display motor status																																				
SAX RESPONSE:	<p>Lists on the LU = 6 device a table showing the current motor positions and settings, as illustrated below.</p> <table><tr><th>MOTOR</th><th>POSITION</th><th>STEPS/SEC</th><th>ON</th></tr><tr><td>1</td><td>2000</td><td>500</td><td>0</td></tr><tr><td>2</td><td>400</td><td>500</td><td>0</td></tr><tr><td>3</td><td>250</td><td>500</td><td>0</td></tr><tr><td>4</td><td>5000</td><td>500</td><td>0</td></tr><tr><td>5</td><td>100</td><td>500</td><td>0</td></tr><tr><td>6</td><td>200</td><td>500</td><td>0</td></tr><tr><td>7</td><td>2000</td><td>600</td><td>0</td></tr><tr><td>8</td><td>50</td><td>500</td><td>0</td></tr></table> <p>In the fourth column, a '0' indicates the motor power is off while '1' indicates it is on.</p>	MOTOR	POSITION	STEPS/SEC	ON	1	2000	500	0	2	400	500	0	3	250	500	0	4	5000	500	0	5	100	500	0	6	200	500	0	7	2000	600	0	8	50	500	0
MOTOR	POSITION	STEPS/SEC	ON																																		
1	2000	500	0																																		
2	400	500	0																																		
3	250	500	0																																		
4	5000	500	0																																		
5	100	500	0																																		
6	200	500	0																																		
7	2000	600	0																																		
8	50	500	0																																		
USER RESPONSE:	None.																																				
<u>EXIT</u>	To exit from motor subroutine.																																				
SAX RESPONSE:	Returns control to SAX and types SELECT FUNCTION																																				

Table 5(1) continued

<u>FNC</u>	To display motor functions.																		
SAX RESPONSE:	<p>Print a table of motor functions. This table indicates what each motor is physically connected to as well as a list of the allowable functions, as illustrated below.</p> <table> <thead> <tr> <th>MOTOR</th><th>FUNCTION</th></tr> </thead> <tbody> <tr> <td>1</td><td>SLIT X-AXIS</td></tr> <tr> <td>2</td><td>SLIT Y-AXIS</td></tr> <tr> <td>3</td><td>SPECIMEN X-AXIS</td></tr> <tr> <td>4</td><td>SPECIMEN Y-AXIS</td></tr> <tr> <td>5</td><td>DETECTOR X-AXIS</td></tr> <tr> <td>6</td><td>DETECTOR Y-AXIS</td></tr> <tr> <td>7</td><td>GLASSY CARBON</td></tr> <tr> <td>8</td><td>SPECIMEN WHEEL</td></tr> </tbody> </table> <p>THE FUNCTIONS ARE DIS = DISPLAY MOTOR POSITIONS EXIT = RETURN TO SAX CONTROL FNC = PRINT FUNCTIONS MOV = MOVE MOTORS OFF = TURN MOTORS OFF ON = TURN MOTORS ON POS = RESET CURRENT MOTOR POSITIONS RATE = RESET CURRENT MOTOR RATE</p>	MOTOR	FUNCTION	1	SLIT X-AXIS	2	SLIT Y-AXIS	3	SPECIMEN X-AXIS	4	SPECIMEN Y-AXIS	5	DETECTOR X-AXIS	6	DETECTOR Y-AXIS	7	GLASSY CARBON	8	SPECIMEN WHEEL
MOTOR	FUNCTION																		
1	SLIT X-AXIS																		
2	SLIT Y-AXIS																		
3	SPECIMEN X-AXIS																		
4	SPECIMEN Y-AXIS																		
5	DETECTOR X-AXIS																		
6	DETECTOR Y-AXIS																		
7	GLASSY CARBON																		
8	SPECIMEN WHEEL																		
USER RESPONSE:	None.																		
<u>MOVE</u>	To move motors																		
SAX RESPONSE:	ENTER MOTORS AND NEW POSITIONS																		
USER RESPONSE:	l, xxxx, m, yyyy, n, zzzz, ... , where l, m, n, ... are motor numbers and xxxx, yyyy, zzzz are their desired new positions.																		
COMMENTS:	All motors identified in the <u>MOVE</u> command will move simultaneously until each has reached its desired new position. If a limit switch is encountered before the goal is reached (a) the motor will cease moving, (b) the program will continue to "step" the motor until it "thinks" the goal is reached, and (c) the appropriate motor limit switch LED will be turned on. If the motor SI has been enabled (see <u>ENABLE</u>), SAX can be put in a WAIT state until that limit switch is reached.																		

Table 5(1) continued

<u>OFF</u>	To turn motor power supplies off.
SAX RESPONSE:	WHICH MOTORS SHOULD BE OFF?
USER RESPONSE:	$\{l, m, n, \dots\}$ ALL where $l, m, n, \dots$ is a list of motor numbers whose power supplies should be turned off
COMMENT:	By implication, motors not listed in the user response whose power is off will be turned on.
<u>ON</u>	To turn motor power supplies on.
SAX RESPONSE:	WHICH MOTORS SHOULD BE ON?
USER RESPONSE:	$l, m, n, \dots$, where $l, m, n, \dots$ is a list of the motor numbers whose power supplies should be turned on.
COMMENT:	By implication, motors not listed in the user response whose power is on will be turned off.
<u>POS</u>	To set current motor positions.
SAX RESPONSE:	SELECT MOTOR AND NEW POSITION.
USER RESPONSE:	$l, xxxx, m, yyyy, n, zzzz, \dots$, where $l, m, n, \dots$ are motor numbers and $xxxx, yyyy, zzzz, \dots$ are their current positions.
COMMENT:	The motors are <u>not</u> moved. The positions read in are accepted as the <u>current</u> positions of the motors.
<u>RATE</u>	To set motor speed (pulses/s).
SAX RESPONSE:	SELECT MOTOR AND NEW RATE.
USER RESPONSE:	$l, rrrr, m, ssss, n, tttt, \dots$, $l, m, n, \dots$ are motor numbers and $rrrr, ssss, tttt$ are motor stepping rates (in pulses/s).
COMMENT:	Motor stepping rates must lie in the range $9 \leq rrrr \leq 625$ pulses/s. The lower limit is set by the countdown clock which determines the time interval between motor pulses and which cannot be greater than $2^{16} - 1$; the upper limit is determined by the number of instructions in the loop which outputs motor pulses to the hardware and which cannot be decreased.

Table 5(m). User Commands for Program SAX

<u>NLOOP</u>	
FUNCTION:	To set a counter which determines how many times a sequence of SAX commands are executed by the <u>LOOP</u> command.
SAX RESPONSE:	HOW MANY TIMES SHOULD PROGRAM LOOP (I2)?
USER RESPONSE:	Enter the number of times the sequence of commands defined by the <u>LOOP</u> command should be executed and enter in an I2 format.
COMMENT:	<i>The rewindable device assigned to lu = 5 is back-spaced the number of records set by <u>LOOP</u>. This procedure is repeated <u>NLOOP</u> times.</i>

Table 5(n). User Commands for Program SAX

<u>PRINT</u>	
FUNCTION:	To read the contents of the scaler/timer devices and to store their contents in HEADER.
SAX RESPONSE:	<p>A table listing the contents of each scaler is printed, as illustrated below.</p> <pre> SCALER 1 = 0.5338000E+02 SCALER 2 = 0.6980600E+06 SCALER 3 = 0.1000000E+03 </pre>
NOTES:	<ol style="list-style-type: none"> 1. At the time PRI is executed the contents of the system calendar and clock are stored in HEDREC(9). 2. PRI does not clear the contents of the scalers/timers, and can be used while the units are counting.

Table 5(o). User Commands for Program SAX

<u>RESUME</u>	
FUNCTION:	To resume data acquisition. This is accomplished by setting the appropriate bits in the system control module to start (1) data transfer from the micro-processor to computer memory, and (2) the scaler/timers <u>without clearing their previous contents</u> (see also <u>RUN.</u>)
SAX RESPONSE:	MICROPROCESSOR RUNNING.
NOTE:	<u>RESUME</u> DOES NOT FUNCTION IF THE SCALER/TIMERS HAVE REACHED THEIR PRESET VALUES.

Table 5(p). User Commands for Program SAX

<u>RUN</u>	
FUNCTION:	To start data acquisition. This is accomplished by setting the appropriate bits in the system control module to (1) start data transfer from the micro-processor to computer memory, and (2) start the scaler/timers <u>after first clearing their previous contents</u> . (See also <i>RESUME</i> .)
SAX RESPONSE:	MICROPROCESSOR RUNNING.
USER RESPONSE:	None.
COMMENT:	<i><u>RUN</u> does not clear the DATA or HEADER arrays. This function is performed by <u>ZERO</u>.</i>

Table 5(q). User Commands for Program SAX

<u>SAVE</u>	
FUNCTION:	To save data on disc. The data base is a FORTRAN direct access I/O file addressed by logical unit LU (see <u>LU</u>). The data contained in the global commons <i>HEADER</i> and <i>DATA</i> (see Sect. 11) are saved by the subroutine <i>SAVE</i> described elsewhere in this series. ¹⁶ Following the <i>SAVE</i> function, the experiment sequence number, <i>ISEQ</i> , is incremented by 1.
SAX RESPONSE:	DATA WILL BE STORED UNDER THE SEQUENCE NO. xxxx.
COMMENTS:	If any errors occur during the <i>SAVE</i> operation, they will be reported to the user following the above response. The meaning of these diagnostic messages is given elsewhere. ¹⁶ Following the error, the message, <i>SAX IN WAIT STATE</i> , will be written to <i>lu = 6</i> , and <i>SAX</i> will be HELD with a REX, #11. Once the user has determined the correct course of action, and modified <i>HEADER</i> accordingly, <i>SAX</i> may be restarted with the operator directive <i>/SAX/R</i>

¹⁶ R. L. Stephenson and R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. 13. *File Management Programs*, ORNL/TM-6342/V. 13.

Table 5(r). User Commands for Program SAX

<u>SEQ</u>	
FUNCTION:	To enter a new experiment sequence number.
SAX RESPONSE:	CURRENT SEQUENCE NUMBER = xxxxxx ENTER NEW SEQ NO. (I6).
USER RESPONSE:	Enter sequence number (up to five digits right justified in an I6 field).
SAX RESPONSE:	NEW SEQUENCE NUMBER IS xxxxx.
COMMENTS:	<p>1. ISEQ is stored in HEDREC(1) as indicated in Sect. 11.3. On first activation of SAX, SEQ is called to initialize the system. Following this initialization, ISEQ is incremented by 1 each time a data set is <u>SAVED</u>. SEQ should be used only when data sets have been erroneously saved, and have been deleted by MANAGER.¹⁶</p> <p>2. When first activated, SAX automatically calls SEQ prior to any other function.</p>

Table 5(s). User Commands for Program SAX

<u>STATUS</u>	
FUNCTION:	To display camera and experiment status. Only data stored in <i>HEADER</i> will be displayed; thus <i>ID</i> , <i>INI</i> , and <i>PRI</i> should be called first.
SAX RESPONSE:	<i>A status table of the form shown below is generated. The time and date are inserted at the time the PRI command is executed.</i>
USER RESPONSE:	None.
	<pre> STATUS INFORMATION FOR OWNER ID SEQUENCE NO. 7000 RWH TITLE: NEUTRON IRRADIATED NI (#7-15) EXPERIMENT PERFORMED ON: 4-11-78 THIS EXPERIMENT WAS COMPLETED AT 16:21:13 TSAMP = 0.375 ABSINT = 0.15460E+07 THE DATA ARRAY DIMENSIONS ARE 64 BY 64 A TOTAL OF 0.8250E+05 PHOTONS HAVE BEEN RECORDED USING DATA TRANSFER ROUTINE 2 0 OVERFLOW BUFFERS WERE WRITTEN TO DISC BEAM MONITOR COUNTS = 9952. INTEGRATED INTENSITY = 83672. ELAPSED TIME = 100.0 SECONDS </pre>

Table 5(t). User Commands for Program SAX

<u>STOP</u>	
FUNCTION:	To stop data transfer from the microprocessor to computer memory.
SAX RESPONSE:	HALT MICROPROCESSOR.
USER RESPONSE:	None.

Table 5(u). User Commands for Program SAX

<u>UP</u>	
FUNCTION:	To raise glassy carbon standard out of the beam.
METHOD:	The power to motor 7 is turned on and pulses are output to the motor. The upper limit switch is tested after each pulse to see if it was reached, and when so, the power to the motor is turned off.
SAX RESPONSE:	GLASSY CARBON OUT OF BEAM.
USER RESPONSE:	None.
ERRORS:	None.

Table 5(v). User Commands for Program SAX

<u>WAIT</u> ,	
FUNCTION:	To put SAX in a wait, or hold, state.
METHOD:	A REX, #11 is executed. Following this instruction, the TASKMASTER suspends the execution of SAX indefinitely.
SAX RESPONSE:	SAX IN WAIT STATE.
USER RESPONSE:	SAX begins execution at the next instruction following the REX, #11 when the operator directive /SAX/R is entered.

Table 5(w). User Commands for Program SAX

<u>ZERO</u>	
FUNCTION:	To reinitialize the <i>HEADER</i> and <i>DATA</i> global commons.
METHOD:	The record as set up in <i>INI</i> is copied into <i>HEADER</i> and the current value of <i>ISEQ</i> is stored in the first word. The <i>DATA</i> array is zeroed, and the first four words are set up as in <i>INI</i> .
SAX RESPONSE:	DO YOU WISH TO ZERO THE DATA ARRAY AND OVERFLOW BUFFERS?
USER RESPONSE:	$\left\{ \begin{array}{c} \underline{\text{YES}} \\ \underline{\text{NO}} \end{array} \right\}$
SAX RESPONSE:	DATA ARRAY AND OVERFLOW BUFFERS CLEARED. (If <u>YES</u> above). DO YOU WISH TO CLEAR THE HEADER RECORD?
USER RESPONSE:	$\left\{ \begin{array}{c} \underline{\text{YES}} \\ \underline{\text{NO}} \end{array} \right\}$
SAX RESPONSE:	HEADER RECORD CLEARED. (If <u>YES</u> above).
ERRORS:	INITIALIZE MICROPROCESSOR BEFORE ZEROING ARRAYS. The dimensions of the data array are unknown to SAX before an <i>INI</i> is issued. Hence, garbage indices, which may halt the computer or cause it to perform erroneous executions, can occur.

VII. SYSTEM PROGRAMMING NOTES

Program OVF uses the global common OVER and thus must be link-edited with the statement

```
      GCOM  OVER  #101  #191 .
```

Note that OVF is a FORTRAN program with in-line assembly language code. The program uses only the global file C0 as an output device, and thus no TFILES need be assigned in the Task Work Area (TWA). The completed load module, with TWA, is catalogued in the LM file as a Privileged Task (PTASK).

Program MOT does not use any global commons and does not make any external references. Since it is written completely in assembly language, it, along with its TWA, can be assembled in one pass with the MACRO catalogued procedure. The completed load module is catalogued in the LM file as a PTASK.

Program SAX is a FORTRAN program with in-line assembly language for the custom interfaces on the General Purpose Controllers (GPCs). Because it contains both global commons and load-on-cell overlays, it must be link-edited separately after compilation and assembly. Following this, the program is catalogued in the LM file in four steps, one for the root segment and three for the three overlay segments. In cataloguing the root, it is essential (1) to add sufficient space to hold the largest overlay segment (the additional size is determined by subtracting the HIGH for the root from the HIGH for the longest overlay as determined from the link-edit map), and (2) to include two buffers for the FORTRAN direct access I/O file. A complete example is given in Appendix E. In the example, it is assumed that no changes in the TWA have been made. Thus, the TWA for the currently catalogued program (W\$SAX) can be used again. It has been fetched by the GET directive to the DISC CATALOGUER. Note that W\$SAX must follow the root segment (without a file mark) and the overlays then follow.

APPENDIX A
PROGRAM OV F SOURCE LISTING

*C MODCOMP SOURCE EDITOR

DATE 10/08/79 08:49:25

```

1      PROGRAM OVF
2      DIMENSION JBUFF(200),KBUFF(200)
3      COMMON/OVER/IPT,JBUFF,KBUFF
4      INLINE
5      LDI,2      UFTCO
6      REX,1
7      DFC      MSG1,26
8      START1 LDI,3      JBUFF
9      STM,3      #6F      SET ADDRESS OF OVERFLOW BUFFER FOR CP
10     ZRR,3      0
11     STM,3      IPT      CLEAR OVERFLOW FLAG
12     STM,3      JBUFF
13     LDI,2      JBUFF
14     REX,#1E      WAIT FOR FIRTS OVERFLOW
15     DFC      #8000+#4000+#2000+3
16     ABMM,15      IPT
17     LDI,2      KBUFF+190
18     REX,#1E      WAIT FOR 390TH OVERFLOW
19     DFC      #8000+#4000+#2000+3
20     LDI,2      #40A8
21     OCA,2,6
22     LDM,1
23     STS,2,1      #7F      TELL SI HANDLER LIST
24     AGAIN LDI,2      UFTCO
25     REX,1
26     DFC      MSG2,126
27     OBMM,0      IPT
28     REX,#11      HOLD FOR OPERATOR RESPONSE
29     ZRR,3
30     CRMB,3      IPT,START,AGAIN
31     BRU      AGAIN
32     *
33     *      CONSTANTS, UFTS, AND MESSAGES
34     *
35     UFTCO EQU      $
36     DFC      0,ECO ,#A000,0,0,0
37     MSG1 EQU      $
38     DFC      " OVF TASK NOW EXECUTING "
39     DFC      #0D0A,#2000
40     MSG2 EQU      $
41     DFC      "*****",#0D0A
42     DFC      " OVERFLOW BUFFER FULL ",#0D0A
43     DFC      " TO RESUME, CLLAR BUFFER",#0D0A
44     DFC      " THEN TYPE /OVF/R ",#0D0A
45     DFC      "*****",#0D0A,#2000
46     FINI
47     END
EXIT
$AVR C1 4
$END DO

```

*C MODCOMP SOURCE EDITOR DATE 10/08/79 08:32:03
 RESOURCE BLOCK FOR OVERFLO (OVF) TASK

```

1      TTL RESOURCE BLOCK FOR OVERFLO (OVF) TASK
2      INS SO,M3TWA
3      RESOURCES OVF,NON,1,0,50,48
4      TFILES 1
5      ENDRSOURCES
EXIT
$AVR C1 4
$END DO
$LIST MO1
$NOP
$EXE SED

```

APPENDIX B

PROGRAM MOT SOURCE LISTING

*C MODCOMP SOURCE EDITOR

DATE 10/08/79 08:46:33

```

1      PGM MOT
2      *
3      * THIS PROGRAM KEEPS TRACK OF HOW LONG POWER TO MOTOR 7 IS ON
4      *
5      LDI,2      UFT
6      REX,#1
7      DFC      MSG1,26
8      LDM,1      #7F
9      START1    LDS,2,3      GET MOTOR STATUS
10     TBRB,2,14  ON          MOTOR 7 ON?
11     REX,#14    NO
12     DFC      #8000,3000    WAIT 30 SEC
13     BRU      START1      TEST AGAIN
14     ON       REX,#14      MOTOR 7 ON
15     DFC      #8002,0      WAIT 2 MINUTES
16     LDS,2,3      GET MOTOR STATUS
17     TBRB,2,14  OFF       MOTOR 7 STILL ON?
18     BRU      START
19     OFF      ZBR,2,14      CLEAR MOTOR 7 OFF FLAG
20     STS,2,3      SET SI
21     UCA,2,7      TURN MOTOR 7 OFF
22     LDI,2      COUFT
23     REX,1      INFORM OPERATOR
24     DFC      MSG2,52
25     BRU      START1      START OVER
26     *
27     * CONSTANTS, UFTS, AND MESSAGES
28     *
29     UFT      EQU      $
30     DFC      0,@OUT,#A000,0,0,0
31     COUFT    EQU      $
32     DFC      0,@CD,#A000,0,0,0
33     MSG1     EQU      $
34     DFC      " MOT TASK NOW EXECUTING ",#0D0A,#2000
35     MSG2     EQU      $
36     DFC      " MOTOR 7 HAS BEEN ON OVER 2 MIN ",#0D0A
37     DFC      " NOW TURNED OFF ",#0D0A,#2000
38     END

```

*C MODCOMP SOURCE EDITOR
 RESOURCE BLOCK FOR MOTOR (MOT) TASK

DATE 10/08/79 08:46:33

```

39     TTL      RESOURCE BLOCK FOR MOTOR (MOT) TASK
40     INS      SU,M3TWA
41     RESOURCES MOT,NON,1,0,50,48
42     TFILE    OUT,TY
43     ENDRESOURCES
EXIT
$AVR CI 4
$END DO

```

APPENDIX C

PROGRAM SAX SOURCE LISTING*

*In order to avoid the necessity of photographically reducing the listing, some lines of code have been cut and moved to the next line. This is easily identified by a gap in the line numbers.

LIST

*C MODCOMP SOURCE EDITOR

DATE 10/08/79 09:10:41

```
1-----
1      PROGRAM SAX
264     END
265     SUBROUTINE DELAY
339     END
340     SUBROUTINE DISABLE
372     END
373     SUBROUTINE DOWN
410     END
411     SUBROUTINE DRIVER
442     END
443     SUBROUTINE ENABLE
475     END
476     SUBROUTINE FNC
511     END
512     SUBROUTINE HRDCPY
530     END
531     SUBROUTINE IDENT
570     END
571     SUBROUTINE INCR
583     END
584     SUBROUTINE INITL
723     END
724     SUBROUTINE LOOP
737     END
738     SUBROUTINE MOTOR
885     END
886     SUBROUTINE OV1LY(N,NAME,LENGTH,IADD,NERR)
908     END
909     SUBROUTINE OV2LY(N,ISPEC)
921     END
922     SUBROUTINE OV3LY(N)
940     END
941     SUBROUTINE PRINT
980     END
981     SUBROUTINE RESUME
997     END
998     SUBROUTINE RUN
1019    END
1020    SUBROUTINE SETLOOP
1027    END
1028    SUBROUTINE SEQU
1045    END
1046    SUBROUTINE SPEC(ISPEC)
1106    END
1107    SUBROUTINE STATUS
1170    END
1171    SUBROUTINE STOP
1184    END
1185    SUBROUTINE TRANS(ISPEC)
1278    END
1279    SUBROUTINE UP
1316    END
1317    SUBROUTINE ZERO
1375    END
```

*C MODCOMP SOURCE EDITOR DATE 10/08/79 09:53:42

```

1  PROGRAM SAX
2  C *** VERSION A.02, 24 SEPTEMBER 1979 *****
3  DIMENSION IDATA(64,64),JBUFF(200),KBUFF(200),ITIME(6)
4  DIMENSION JDATA(4100)E(8),JTEST(8),IRUF(5)
5  DIMENSION ISETUP(100)
6  INTEGER ANS(3),CPOS(8),HEDREC(100)
7  LOGICAL HDCPY,INIFLG
8  COMMON/DATA/JDATA
9  COMMON/HEADER/HEDREC
10 COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
11 COMMON/OVER/NOVRF,JBUFF,KBUFF
12 COMMON/HARD/HDCPY
13 COMMON/LOOPS/ILOOP,ISEQ
14 COMMON/INITL/INIFLG,ISETUP
15 COMMON/MANAGE/MAXREC,MXWRDS,LU,IDEM
16 EQUIVALENCE (M,JDATA(3)),(N,JDATA(4)),(JDATA(5),IDATA(1,1))
17 EQUIVALENCE (HEDREC(18),ITIME(1)),(HEDREC(54),ABSINT)
18 DEFINE FILE 21(4480,200,L,NXTREC)
19 CALL TIMDAT(ITIME)
20 CALL CORE(IRUF,5)
21 IMIN=MOD(ITIME(1),60)
22 IHR=(ITIME(1)-IMIN)/60
23 ITC=MOD(ITIME(2),ITIME(6))
24 ISEC=(ITIME(2)-ITC)/ITIME(6)
25 WRITE(6,8000) ITIME(4),ITIME(3),ITIME(5),IHR,IMIN,ISEC
26 8000 FORMAT(1H,'THE ORNL SMALL-ANGLE SCATTERING MONITOR')
27 X1H,'VERSION A.02 24 SEPTEMBER 1979'
28 X1H,'IS NOW OPERATING'///
29 X1H,'THE DATE IS ',I2,'-',I2,'-',I2/
30 X1H,'THE TIME IS ',I2,':',I2,':',I2//
31 INLINE
32 REX,#16
33 DFC ERR1,@OVF,0,0
34 BRU GO1
35 EQU $
36 LDI,2 SOUFT
37 REX,1
38 DFC MSG1,28
39 BRU GO1
40 MSG1 EQU $
41 DFC " PROGRAM 'OVF' NOT FOUND"
42 DFC #0D0A,$2000
43 SOUFT DFC 0,@CD,$A000,0,0,0
44 GO1 EQU $
45 REX,#16
46 DFC ERR2,@MOT,0,0
47 BRU GO2
48 ERR2 EQU $
49 LDI,2 SOUFT
50 REX,1
51 DFC MSG2,28
52 BRU GO2
53 MSG2 EQU $
54 DFC " PROGRAM 'MOT' NOT FOUND"
55 DFC #0D0A,$2000
56 GO2 EQU $
57 FINI
58 C...
59 C... SET DEFAULT PARAMETERS
60 C...
61 INIFLG=.FALSE.
62 HDCPY=.FALSE.
63 LU = 21
64 MAXREC=4480
65 MXWRDS=100
66 IDEM=4100
67 DO 4 I=1,8
68 JTEST(I)=0
69 CPOS(I)=0
70 NPOS(I)=0
71 4 IRATE(I)=500
72 IRATE(7)=2000
73 ILOOP=0
74 ISEQ=0
75 CALL OV3LY(8)
76 C...
77 C... START MONITOR
78 C...
79 30 WRITE(6,8020)
80 8020 FORMAT(1H,'SELECT FUNCTION')

```

```

81      35 READ(5,8010) ANS
82      8010 FORMAT(3A1)
83      IF(HDCPY) WRITE(7,8011) ANS
84      8011 FORMAT(1H,3A1)
85      IF(ANS(1).EQ.'D') GO TO 400
86      IF(ANS(1).EQ.'E') GO TO 500
87      IF(ANS(1).EQ.'F') GO TO 600
88      IF(ANS(1).EQ.'H') GO TO 800
89      IF(ANS(1).EQ.'I') GO TO 900
90      IF(ANS(1).EQ.'K') GO TO 1100
91      IF(ANS(1).EQ.'L') GO TO 1200
92      IF(ANS(1).EQ.'M') GO TO 1300
93      IF(ANS(1).EQ.'N') GO TO 1400
94      IF(ANS(1).EQ.'P') GO TO 1600
95      IF(ANS(1).EQ.'R') GO TO 1800
96      IF(ANS(1).EQ.'S') GO TO 1900
97      IF(ANS(1).EQ.'T') GO TO 2000
98      IF(ANS(1).EQ.'U') GO TO 2100
99      IF(ANS(1).EQ.'W') GO TO 2300
100     IF(ANS(1).EQ.'Z') GO TO 2600
101     WRITE(6,8030)
102     8030 FORMAT(1H,'INVALID... PLEASE TRY AGAIN'//)
103     GO TO 30
104     400 IF(ANS(2).EQ.'E') GO TO 405
105     IF(ANS(2).EQ.'I') GO TO 410
106     IF(ANS(2).EQ.'W') GO TO 415
107     WRITE(6,8030)
108     GO TO 30
109     405 CALL OV1LY(5)
110     GO TO 30
111     410 CALL OV1LY(7)
112     GO TO 30
113     415 CALL OV2LY(3,0)
114     GO TO 30
115     500 CALL OV1LY(6)
116     GO TO 30
117     600 CALL OV1LY(9)
118     GO TO 30
119     800 CALL OV1LY(10)
120     GO TO 30
121     900 IF(ANS(2).EQ.'D') GO TO 905
122     IF(ANS(2).EQ.'N') GO TO 910
123     WRITE(6,8030)
124     GO TO 30
125     905 CALL OV3LY(3)
126     GO TO 30
127     910 IF(ANS(3).EQ.'I') GO TO 915
128     IF(ANS(3).EQ.'C') GO TO 920
129     WRITE(6,8030)
130     GO TO 30
131     915 CALL OV3LY(4)
132     GO TO 30
133     920 CALL OV3LY(5)
134     GO TO 30
135     1100 CONTINUE
136     CALL OV1LY(3)
137     INLINE
138     REX,#17      ABORT(MOT)
139     DFC          ERR3,@MOT,@END
140     BRU          GO3
141     ERR3 EQU     $
142     LDI,2       SOUFT
143     REX,1
144     DFC          MSG3,28
145     BRU          GO3
146     MSG3 EQU    $
147     DFC          " TASK 'MOT' NOT IN CPUQ ",#0D0A,#2000
148     GO3 EQU     $
149     REX,#17      ABORT(OVF)
150     DFC          ERR4,@OVF,@END
151     BRU          GO4
152     ERR4 EQU    $
153     LDI,2       SOUFT
154     REX,1
155     DFC          MSG4,28
156     BRU          GO4
157     MSG4 EQU    $
158     DFC          " TASK 'OVF' NOT IN CPUQ ",#0D0A,#2000
159     GO4 EQU     $
160     REX,#12      EXIT
161     FINI
162     1200 IF(ANS(2).EQ.'O') GO TO 1210
163     IF(ANS(2).EQ.'U') GO TO 1205
164     WRITE(6,8030)
165     GO TO 30
166     1205 WRITE(6,8065) LU

```

```

167 8065 FORMAT( ' CURRENT LU = ',I3// ' ENTER NEW LU (I2)')
168 READ(5,8060) LU
169 IF(HDCPY) WRITE(7,8066) LU
170 8066 FORMAT( ' NEW LU = ',I3)
171 GO TO 30
172 1210 IF(ILOOP.EQ. 0) GO TO 30
173 CALL OV3LY(6)
174 GO TO 30
175 1300 CALL OV2LY(1,0)
176 GO TO 30
177 1400 CALL OV3LY(7)
178 GO TO 30
179 1600 CALL OV1LY(4)
180 GO TO 30
181 1800 IF(ANS(2).EQ. 'E') GO TO 1830
182 IF(ANS(2).EQ. 'U') GO TO 1820
183 WRITE(6,8030)
184 GO TO 30
185 1820 CALL OV1LY(1)
186 GO TO 30
187 1830 IF(ANS(3).EQ. 'S') GO TO 1840
188 WRITE(6,8030)
189 GO TO 30
190 1840 CALL OV1LY(2)
191 GO TO 30
192 1900 IF(ANS(2).EQ. 'E') GO TO 1970
193 IF(ANS(2).EQ. 'T') GO TO 1920
194 IF(ANS(2).EQ. 'A') GO TO 1950
195 ISPEC = (ANS(2)/256)*256 + ANS(3)/256
196 WRITE(20,8055) ISPEC
197 8055 FORMAT(A2)
198 READ(20,8060) ISPEC
199 8060 FORMAT(I2)
200 IF(ISPEC.GE. 0 .AND. ISPEC.LE. 19) GO TO 1905
201 WRITE(6,8030)
202 GO TO 30
203 1905 CALL OV2LY(4,ISPEC)
204 GO TO 30
205 1920 IF(ANS(3).EQ. 'A') GO TO 1930
206 IF(ANS(3).EQ. 'O') GO TO 1940
207 WRITE(6,8030)
208 GO TO 30
209 1930 CALL OV3LY(1)
210 GO TO 30
211 1940 CALL OV1LY(3)
212 GO TO 30
213 1950 NAME=HEDREC(1)
214 WRITE(6,8040) HEDREC(1),LU
215 8040 FORMAT( ' DATA WILL BE STORED UNDER THE SEQUENCE NO. ',I3,I6/
216 ' ON LOGICAL UNIT = ',I3)
217 IF(HDCPY) WRITE(7,8040) HEDREC(1),LU
218 LENGTH=4100
219 IADD=0
220 HEDREC(7) = NOVRF
221 CALL OV1LY(8,NAME,LENGTH,IADD,NERR)
222 HEDREC(1)=ISEQ
223 IF(NERR.EQ. 0) GO TO 1955
224 WRITE(6,8045) NERR
225 8045 FORMAT( ' ERROR IN SAVE ... NERR = ',I5)
226 IF(HDCPY) WRITE(7,8045) NERR
227 GO TO 2300
228 1955 IF(NOVRF.EQ. 0) GO TO 1966
229 LENGTH=400
230 IADD=-1
231 DO 1960 I=1,200
232 JDATA(1)=JBUFF(I)
233 1960 JDATA(1+200)=KBUFF(I)
234 CALL OV1LY(8,NAME,LENGTH,IADD,NERR)
235 IF(NERR.EQ. 0) GO TO 1965
236 WRITE(6,8045) NERR
237 IF(HDCPY) WRITE(7,8045) NERR
238 GO TO 2300
239 1965 NOVRF=0
240 1966 ISEQ = ISEQ + 1
241 GO TO 30
242 1970 CALL OV3LY(8)
243 GO TO 30
244 2000 ISPEC=(ANS(2)/256)*256+ANS(3)/256
245 WRITE(20,8055) ISPEC
246 READ(20,8060) ISPEC
247 IF(ISPEC.GE. 0 .AND. ISPEC.LT. 19) GO TO 2005
248 WRITE(6,8030)
249 GO TO 30
250 2005 CALL OV2LY(5,ISPEC)
251 GO TO 30
252 2100 CALL OV2LY(2,0)
253 GO TO 30

```



```

254 2300 CONTINUE
255     WRITE(6,8035)
256 8035 FORMAT('SAX IN WAIT STATE')
257     IF(HDCPY) WRITE(7,8035)
258     INLINE
259     REX,#11
260     FINI
261     GO TO 30
262 2600 CALL OV3LY(2)
263     GO TO 30
264     END
265     SUBROUTINE DELAY
266     LOGICAL HDCPY
267     INTEGER ANS(5),ELAPSE
268     COMMON/HARD/HDCPY
269     DATA ELAPSE/428000/
270     5 WRITE(6,8000)
271 8000 FORMAT('WHICH DEVICE CAUSES DELAY?')
272     READ(5,8005) ANS
273 8005 FORMAT(5A1)
274     IF(HDCPY) WRITE(7,8006) ANS
275 8006 FORMAT(1H,' DEVICE ',5A1,' CAUSES DELAY')
276     IF(ANS(1).EQ.'O') GO TO 10
277     IF(ANS(1).EQ.'M') GO TO 15
278     IF(ANS(1).EQ.'T') GO TO 25
279     WRITE(6,8010)
280 8010 FORMAT('INVALID ... TRY AGAIN')
281     GO TO 5
282     10 CONTINUE
283     INLINE
284     LDM,1 #7F
285     TBSB,3,1 NERR1
286     BRU ERR
287 NERR1 TRR,2,1
288     REX,#1E RELINQUISH TILL BIT 15 SET
289     DFC #400F
290     ZBSM,15,0 CLEAR BIT
291     FINI
292     GO TO 20
293     15 IF(ANS(2).EQ.'A') GO TO 30
294     IF(ANS(2).EQ.'O') GO TO 35
295     30 WRITE(6,8035)
296 8035 FORMAT('SAX IN WAIT STATE')
297     IF(HDCPY) WRITE(7,8035)
298     INLINE
299     REX,#11
300     FINI
301     GO TO 20
302     35 CONTINUE
303     INLINE
304     LDM,1 #7F
305     TBSB,3,3 NERR2
306     BRU ERR
307 NERR2 ABR,2,14
308     REX,#1E RELINQUISH TILL BIT 15 SET
309     DFC #400F
310     ZBSM,15,2 CLEAR BIT
311     BRU CONT
312 ERR EQU $
313     FINI
314     WRITE(6,8015)
315 8015 FORMAT('SI HAS NOT BEEN ENABLED FOR DEVICE')
316     IF(HDCPY) WRITE(7,8015)
317     INLINE
318 CONT EQU $
319     FINI
320     20 RETURN
321     25 WRITE(6,8020)
322 8020 FORMAT('ENTER TIME DELAY IN MINUTES AND SECONDS (215)')
323     READ(5,8025) MIN, ISEC
324 8025 FORMAT(2I5)
325     IF(HDCPY) WRITE(7,8030) MIN, ISEC
326 8030 FORMAT('DELAY IS ',I5,' MIN ',I5,' SEC')
327     MIN=MIN+ELAPSE
328     ISEC=ISEC*100
329     INLINE
330     LDM,2 MIN
331     STM,2 AMIN
332     LDM,2 ISEC
333     STM,2 ASEC
334     REX,#14
335 AMIN DFC 0
336 ASEC DFC 0
337     FINI
338     GO TO 20
339     END

```

```

340      SUBROUTINE DISABLE
341      LOGICAL HDCPY
342      INTEGER ANS(5)
343      COMMON/HARD/HDCPY
344      5 WRITE(6,8000)
345      8000 FORMAT(' WHICH DEVICE IS TO BE DISABLED?')
346      READ(5,8005) ANS
347      8005 FORMAT(5A1)
348      IF(ANS(1).EQ.'O') GO TO 10
349      IF(ANS(1).EQ.'M') GO TO 15
350      WRITE(6,8010)
351      8010 FORMAT(' INVALID .... PLEASE TRY AGAIN ')
352      GO TO 5
353      10 CONTINUE
354      INLINE
355      LDM,1 #7F
356      ZBSM,3,1
357      LDS,2,1
358      OCA,2,6
359      FINI
360      GO TO 20
361      15 CONTINUE
362      INLINE
363      LDM,1 #7F
364      ZBSM,3,3
365      LDS,2,3
366      OCA,2,7
367      FINI
368      20 WRITE(6,8015) ANS
369      8015 FORMAT(' DEVICE ',5A1,' DISABLED')
370      IF(HDCPY) WRITE(7,8015) ANS
371      RETURN
372      END
373      SUBROUTINE DOWN
374      DIMENSION NPOS(8),IRATE(8),JTEST(8)
375      INTEGER CPOS(8)
376      LOGICAL HDCPY
377      COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
378      COMMON/HARD/HDCPY
379      INLINE
380      LDM,1 #7F
381      OBSM,14,3      TURN ON MOTOR #7
382      LDS,2,3
383      OCA,2,7
384      REX,14      WAIT FOR MOTOR POWER
385      DFC #8000
386      DFC 100
387      FINI
388      NPOS(7) = -2
389      CALL DRIVER
390      STEP
391      LBR,2,14
392      ODA,2,7
393      ISA,2,7
394      TBRB,2,14      THERE
395      REX,14
396      DFC #8000
397      DFC 1
398      BRU STEP
399      THERE EQU $
400      LDM,1 #7F
401      ZBSM,14,3      TURN OFF MOTOR 7
402      LDS,2,3
403      OCA,2,7
404      FINI
405      CPOS(7) = 0
406      WRITE(6,8000)
407      8000 FORMAT(' GLASSY CARBON IN BEAM')
408      IF(HDCPY) WRITE(7,8000)
409      RETURN
410      END
411      SUBROUTINE DRIVER
412      INTEGER CPOS(8),PULSE
413      DIMENSION NPOS(8),IRATE(8),JTEST(8)
414      COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
415      5 MRATE=600
416      DO 30 I=1,8
417      IF(CPOS(I)-NPOS(I)) 20,15,10
418      10 CALL SETBF(PULSE,I)
419      CALL SETBT(PULSE,I+8)
420      CPOS(I)=CPOS(I)-1
421      GO TO 25
422      15 CALL SETBT(PULSE,I)
423      CALL SETBT(PULSE,I+8)

```

```

424      GO TO 25
425      20 CALL SETBT(PULSE,I)
426      CALL SETBF(PULSE,I+8)
427      CPOS(I)=CPOS(I)+1
428      25 CONTINUE
429      IF(IRATE(I) .LT. MRATE .AND. CPOS(1) .NE. NPOS(1)) MRATE=IRATE(1)
430      30 CONTINUE
431      IF(PULSE .EQ. 0) GO TO 35
432      NCNT=INT(607903./FLOAT(MRATE))-973
433      INLINE
434      LDM,2,PULSE
435      ODA,2,7
436      LDM,2,NCNT
437      LOOP EQU $
438      SRRB,2,15 LOOP
439      FINI
440      GO TO 5
441      35 RETURN
442      END
443      SUBROUTINE ENABLE
444      LOGICAL HDCPY
445      INTEGER ANS(5)
446      COMMON/HARD/HDCPY
447      5 WRITE(6,8000)
448      8000 FORMAT(' WHICH DEVICE IS TO BE ENABLED?')
449      READ(5,8005) ANS
450      8005 FORMAT(5A1)
451      IF(ANS(1) .EQ. '0') GO TO 10
452      IF(ANS(1) .EQ. 'M') GO TO 15
453      WRITE(6,8010)
454      8010 FORMAT(' INVALID ... TRY AGAIN')
455      GO TO 5
456      10 CONTINUE
457      INLINE
458      LDM,1,#7F
459      OBSM,3,1
460      LDS,2,1
461      OCA,2,6
462      FINI
463      GO TO 20
464      15 CONTINUE
465      INLINE
466      LDM,1,#7F
467      OBSM,3,3
468      LDS,2,3
469      OCA,2,7
470      FINI
471      20 WRITE(6,8015) ANS
472      8015 FORMAT(' DEVICE ',5A1,' ENABLED')
473      IF(HDCPY) WRITE(6,8015) ANS
474      RETURN
475      END
476      SUBROUTINE FNC
477      INTEGER ANS(3)
478      WRITE(6,8025)
479      8025 FORMAT(1H,'THE FUNCTIONS ARE')
480      X1H,'DEL = WAIT FOR INTERRUPT'
481      X1H,'DIS = DISABLE SI'
482      X1H,'DWN = PUT GLASSY CARBON IN THE BEAM'
483      X1H,'ENA = ENABLE SI'
484      X1H,'HC = PRINT HARD COPY OF COMMANDS'
485      X1H,'ID = EXPERIMENT IDENTIFICATION'
486      X1H,'FNC = TYPE FUNCTIONS'
487      X1H,'INI = INITIALIZE EXPERIMENT'
488      X1H,'INC = INCREMENT EXPERIMENT SEQUENCE NUMBER'
489      X1H,'KIL = KILL PROGRAM'
490      X1H,'LOO = LOOP INPUT INSTRUCTIONS'
491      X1H,'LU = CHANGE LOGICAL UNIT FOR DATA STORAGE'
492      X1H,'MOT = CALL MOTOR DRIVERS'
493      X1H,'ENTER ANY CHARACTER TO RESUME LIST')
494      READ(5,8010) ANS
495      8010 FORMAT(3A1)
496      WRITE(6,8026)
497      8026 FORMAT(1H,'NLO = SET NUMBER OF TIMES PROGRAM IS TO LOOP')
498      X1H,'PRI = PRINT ORTEC SCALERS AND TIMERS'
499      X1H,'RES = RESUME RUN'
500      X1H,'RUN = START DATA COLLECTION'
501      X1H,'SAV = SAVE THE CONTENTS OF CORE ON DISC'
502      X1H,'SEQ = SET EXPERIMENT SEQUENCE NUMBER'
503      X1H,'SXX = MOVE SPECIMEN XX ON ROTARY HOLDER'
504      X1H,'STA = DISPLAY THE MACHINE STATUS'
505      X1H,'STO = STOP RUN'
506      X1H,'TXX = MEASURE TRANSMISSION COEFFICIENT OF SPECIMEN XX'
507      X1H,'UP = TAKE GLASSY CARBON OUT OF THE BEAM'
508      X1H,'WAI = PUT THIS PROGRAM INTO A WAIT STATE'
509      X1H,'ZER = ZERO MEMORY ARRAY AND OVERFLOW BUFFERS')
510      RETURN
511      END

```

```

512 SUBROUTINE HRDCPY
513 LOGICAL HDCPY
514 INTEGER ANS(3), ITIME(6), HEDREC(100)
515 COMMON/HEADER/HEDREC
516 EQUIVALENCE (ITIME(1), HEDREC(18))
517 IMIN=MOD(ITIME(1), 60)
518 IHR=(ITIME(1)-IMIN)/60
519 ITIC=MOD(ITIME(2), ITIME(6))
520 ISEC=(ITIME(2)-ITIC)/ITIME(6)
521 WRITE(6, 8050)
522 8050 FORMAT(1H, 'DO YOU WANT HARD COPY OF THE INPUT COMMANDS?')
523 READ(5, 8010) ANS
524 8010 FORMAT(3A1)
525 IF(ANS(1).EQ.'N') HDCPY = .FALSE.
526 IF(ANS(1).EQ.'Y') HDCPY = .TRUE.
527 IF (HDCPY) WRITE(7, 8000) ITIME(4), ITIME(3), ITIME(5),
528 XIHR, IMIN, ISEC
529 8000 FORMAT(1H, 'THE ORNL SMALL-ANGLE SCATTERING MONITOR'/
530 X1H, 'VERSION A.02 24 SEPTEMBER 1979'/
531 X1H, 'IS NOW OPERATING'///
532 X1H, 'THE DATE IS ', I2, '-', I2, '-', I2, /
533 X1H, 'THE TIME IS ', I2, ':', I2, ':', I2, //)
534 RETURN
535 END
536 SUBROUTINE IDENT
537 DIMENSION JBUFF(200), KBUFF(200), ITIME(6)
538 DIMENSION IPARM(5), IKIND(5), JDATA(4100)
539 INTEGER HEDREC(100), ANS(3)
540 DIMENSION SCALER(5)
541 LOGICAL HDCPY
542 COMMON/HEADER/HEDREC
543 COMMON/DATA/JDATA
544 COMMON/OVER/NOVRF, JBUFF, KBUFF
545 COMMON/HARD/HDCPY
546 COMMON/LOOPS/ILOOP, ISEQ
547 EQUIVALENCE (HEDREC(52), TSAMP), (HEDREC(18), ITIME(1)),
548 X(HEDREC(32), SCALER(1)), (HEDREC(54), ABSINT)
549 EQUIVALENCE (M, JDATA(3)), (N, JDATA(4))
550 5 WRITE(6, 8000)
551 8000 FORMAT(1H, 'ENTER OWNER ID')
552 READ(5, 8005) ANS
553 NAME = IACAN(ANS)
554 HEDREC(3) = NAME
555 8005 FORMAT(3A1)
556 WRITE(6, 8010)
557 8010 FORMAT(1H, 'ENTER TRANSMISSION COEFFICIENT (F5.0)')
558 READ(5, 8015) TSAMP
559 8015 FORMAT(F5.0)
560 WRITE(6, 8016)
561 8016 FORMAT(1H, 'ENTER ABSOLUTE INTENSITY FACTOR (X.XXXEYY)')
562 READ(5, 8017) ABSINT
563 8017 FORMAT(E8.3)
564 WRITE(6, 8020)
565 8020 FORMAT(1H, 'ENTER TITLE AND COMMENTS (25A2)')
566 READ(5, 8025) (HEDREC(I), I=76, 100)
567 8025 FORMAT(25A2)
568 CALL CTA(NAME, ANS)
569 HEDREC(1) = ISEQ
570 HEDREC(3) = NAME
571 IF(HDCPY) WRITE(7, 8030) ISEQ, ANS, TSAMP, (HEDREC(1), I=76, 100)
572 8030 FORMAT(' SEQ. NO. = ', I4, '1X, I6, 2X, 'OWNER ID = ', 3A1, 5X, 'TSAMP = ',
573 X, F6.3, ' TITLE : ', 25A2)
574 RETURN
575 END
576 SUBROUTINE INCR
577 INTEGER HEDREC(100), ANS(3)
578 LOGICAL HDCPY
579 COMMON/HARD/HDCPY
580 COMMON/HEADER/HEDREC
581 COMMON/LOOPS/ILOOP, ISEQ
582 HEDREC(1)=HEDREC(1)+1
583 ISEQ=ISEQ+1
584 WRITE(6, 8000) HEDREC(1)
585 8000 FORMAT(' NEW SEQUENCE NUMBER IS ', I6)
586 IF(HDCPY) WRITE(7, 8000) HEDREC(1)
587 RETURN
588 END
589 SUBROUTINE INILL
590 DIMENSION JDATA(4100), ISETUP(100), SETUP(22)
591 INTEGER OUTWRD, BLNK, HEDREC(100)
592 LOGICAL HDCPY, INIFLG
593 COMMON/DATA/JDATA
594 COMMON/HARD/HDCPY
595 COMMON/HEADER/HEDREC

```

```

596 COMMON/INITL/INIFLG, ISETUP
597 EQUIVALENCE (ISETUP(32), SETUP(1)), (ABSINT, ISETUP(54))
598 EQUIVALENCE (NU, ISETUP(17)), (NSUR, ISETUP(24))
599 EQUIVALENCE (WAVE, ISETUP(56))
600 EQUIVALENCE (DXCHAN, ISETUP(68)), (DYCHAN, ISETUP(70))
601 EQUIVALENCE (DIST2, ISETUP(62)), (DIST1, ISETUP(60))
602 DATA BLNK// '/'
603 DO 1 I=1,31
604   ISETUP(I) = 0
605 1 CONTINUE
606 DO 2 I=76,100
607 2 ISETUP(I) = BLNK
608 DO 6 I=1,10
609 6 SETUP(I) = 0.0
610 DO 7 I=11,22
611 7 SETUP(I) = 1.0
612 WAVE = 1.5417
613 NU = 3
614 INLINE
615 LDI,2 #4030
616 OCA,2,6
617 REX, #14          DELAY 1 TICK
618 DFC #8000
619 DFC 1
620 LDI,2 #40A0
621 OCA,2,6
622 FINI
623 WRITE(6,8000)
624 8000 FORMAT(1H, 'ENTER ORIGIN OF DATA ARRAY (HEX)//'
625 X1H, 'LAST BYTE MUST BE ZERO//')
626 READ(5,8005) OUTWRD
627 8005 FORMAT(Z4)
628 IF(HDCPY) WRITE(7,8006) OUTWRD
629 8006 FORMAT(' ORIGIN OF DATA ARRAY IS #',Z4)
630 INLINE
631 LDM,2 OUTWRD
632 ODA,2,6
633 REX, #14          DELAY 1 TICK
634 DFC #8000
635 DFC 1
636 LDI,2 #40AC
637 OCA,2,6
638 FINI
639 WRITE(6,8010)
640 8010 FORMAT(1H, 'SELECT MICROPROCESSOR DATA TRANSFER ROUTINE(12)')
641 IF(.NOT.HDCPY) GO TO 3
642 WRITE(6,8012)
643 8012 FORMAT(1H, ' 1 = ADC TEST//'
644 X1H, ' 2 = 64X64 DIRECT DATA TRANSFER//'
645 X1H, ' 3 = 64X64 TRANSFER WITH STACK//'
646 X1H, ' 4 = 128X128 ZOOM//'
647 X1H, ' 5 = 16X256//'
648 X1H, ' 6 = 32X128//'
649 X1H, ' 7 = 64X64 INTERRUPT DRIVEN//')
650 WRITE(7,8010)
651 3 READ(5,8013) OUTWRD
652 8013 FORMAT(I2)
653 IF(HDCPY) WRITE(7,8011) OUTWRD
654 8011 FORMAT(' DATA TRANSFER ROUTINE ',I2,' SELECTED')
655 JDATA(1)=-1
656 JDATA(2)=OUTWRD
657 GO TO (5,10,10,10,25,30,40),OUTWRD
658 5 JDATA(3)=0
659 JDATA(4)=0
660 GO TO 35
661 10 JDATA(3)=64
662 JDATA(4)=64
663 GO TO 35
664 25 JDATA(3)=256
665 JDATA(4)=16
666 GO TO 35
667 30 JDATA(3)=128
668 JDATA(4)=32
669 GO TO 35
670 40 JDATA(3)=64
671 JDATA(4)=64
672 INLINE
673 SIE, #7
674 FINI
675 35 CONTINUE
676 INLINE
677 LDM,2 OUTWRD
678 ODA,2,6
679 REX, #14          DELAY 1 TICK
680 DFC #8000

```

```

681      DFC      1
682      LDI,2    #40AB
683      OCA,2,6
684      LDM,1    #7F
685      STS,2,1
686      FIN1
687      WRITE(6,8015)
688 8015 FORMAT(1H,'MICROPROCESSOR INITIALIZED'//)
689      INIFLG=.TRUE.
690      ISETUP(11) = JDATA(3)
691      ISETUP(12) = JDATA(4)
692      ISETUP(24) = OUTWRD
693      WRITE(6,8020)
694 8020 FORMAT(' ENTER X-RAY WAVELENGTH (X.XXXX)')
695      READ(5,8025) WAVE
696 8025 FORMAT(F6.4)
697      IF(HDCPY) WRITE(7,8026) WAVE
698 8026 FORMAT(' X-RAY WAVELENGTH = ',F6.4)
699      WRITE(6,8030)
700 8030 FORMAT(' HOW MANY SCALERS/TIMERS ? (11)')
701      READ(5,8035) NU
702 8035 FORMAT(I1)
703      IF(HDCPY) WRITE(7,8036) NU
704 8036 FORMAT(' THERE ARE ',I1,' SCALERS AND TIMERS')
705      WRITE(6,8040)
706 8040 FORMAT(' ENTER SPECIMEN TO DETECTOR DISTANCE')
707      X, '(IN MM) (F8.3)')
708      READ(5,8045) DIST2
709 8045 FORMAT(F8.3)
710      WRITE(6,8050)
711 8050 FORMAT(' ENTER SPECIMEN TO SOURCE DISTANCE (IN MM),(F8.3)')
712      READ(5,8045) DIST1
713      WRITE(6,8055)
714 8055 FORMAT(' ENTER X-AXIS CHANNELS TO MM CONVERSION FACTOR(F8.3)')
715      READ(5,8045) DXCHAN
716      DYCHAN = DXCHAN
717      IF(HDCPY) WRITE(7,8047) DIST2
718 8047 FORMAT(' THE SPECIMEN TO DETECTOR DISTANCE IS ',F8.2
719      X, ' MM')
720      IF(HDCPY) WRITE(7,8056) DIST1
721 8056 FORMAT(' THE SPECIMEN TO SOURCE DISTANCE IS ',F8.2,' MM')
722      IF(HDCPY) WRITE(7,8057) DXCHAN
723 8057 FORMAT(' THE X-AXIS CHANNELS TO MM CONVERSION FACTOR IS'
724      X, F8.2, ' CHANNELS/MM')
725      DO 400 I=1,100
726 400 HEDREC(I) = ISETUP(I)
727      RETURN
728      END
729      SUBROUTINE LOOP
730      INTEGER UFT(6)
731      COMMON/LOOPS/ILOOP,ISEQ
732      DATA UFT/0,ZC800,ZA000,3*0/
733      ILOOP=ILOOP-1
734      WRITE(6,8000)
735 8000 FORMAT(' HOW MANY STATEMENTS DO WE BACK UP (I2)?')
736      READ(5,8005) NSTAT
737 8005 FORMAT(I2)
738      DO 5 I=1,NSTAT
739      CALL BKREC(UFT)
740      5 CONTINUE
741      RETURN
742      END
743      SUBROUTINE MOTOR
744      DIMENSION NPOS(8),IRATE(8),JTEST(8),IPARM(16),IKIND(16)
745      LOGICAL HDCPY
746      INTEGER CPOS(8),ANS(3),PULSE
747      COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
748      COMMON/HARD/HDCPY
749      1 IF(HDCPY) WRITE(6,8000)
750 8000 FORMAT(1H,'MOTOR FUNCTION'//
751      X1H,' 1 SLIT X-AXIS'//
752      X1H,' 2 SLIT Y-AXIS'//
753      X1H,' 3 SPECIMEN X-AXIS'//
754      X1H,' 4 SPECIMEN Y-AXIS'//
755      X1H,' 5 DETECTOR X-AXIS'//
756      X1H,' 6 DETECTOR Y-AXIS'//
757      X1H,' 7 GLASSY CARBON'//
758      X1H,' 8 SPECIMEN WHEEL'//)
759      IF(HDCPY) WRITE(6,8010)
760 8010 FORMAT(1H,'THE FUNCTIONS ARE'//
761      X1H,'DIS = DISPLAY MOTOR POSITIONS'//
762      X1H,'EXIT = RETURN TO SAX CONTROL'//
763      X1H,'FNC = PRINT FUNCTIONS'//
764      X1H,'MOV = MOVE MOTORS'//
765      X1H,'OFF = TURN MOTORS OFF'//
766      X1H,'ON = TURN MOTORS ON'//

```

```

767      X1H , 'POS = RESET CURRENT MOTOR POSITION'//
768      X1H , 'RATE = RESET CURRENT MOTOR RATE'///)
769      S WRITE(6,8005)
770      8005 FORMAT(1H , 'SELECT MOTOR FUNCTION'//)
771      READ(5,3015) ANS
772      8015 FORMAT(3A1)
773      IF(ANS(1) .EQ. 'D') GO TO 400
774      IF(ANS(1) .EQ. 'E') RETURN
775      IF(ANS(1) .EQ. 'F') GO TO 600
776      IF(ANS(1) .EQ. 'M') GO TO 1300
777      IF(ANS(1) .EQ. 'O') GO TO 1400
778      IF(ANS(1) .EQ. 'P') GO TO 1500
779      IF(ANS(1) .EQ. 'R') GO TO 1800
780      WRITE(6,8020)
781      8020 FORMAT(1H , 'INVALID.. PLEASE TRY AGAIN'///)
782      GO TO 5
783      400 WRITE(6,8025)
784      8025 FORMAT(1H , 'MOTOR POSITION STEPS/SEC ON'//)
785      WRITE(6,8030)(I,CPOS(I),IRATE(I),JTEST(I),I=1,8)
786      8030 FORMAT(2X,I1,6X,I5,7X,I5,5X,I1)
787      IF(HDCPY) WRITE(7,8030) (I,CPOS(I),IRATE(I),JTEST(I),I=1,8)
788      GO TO 5
789      600 HDCPY=.TRUE.
790      GO TO 1
791      1300 WRITE(6,8035)
792      8035 FORMAT(' ENTER MOTORS AND NEW POSITIONS')
793      NUM=16
794      CALL PARM(IPARM,IKIND,5,NUM)
795      IF(NUM .GT. 0 .AND. NUM .LE. 16) GO TO 1302
796      1301 WRITE(6,8020)
797      GO TO 1300
798      1302 IF(MOD(NUM,2)) 1301,1303,1301
799      1303 DO 1304 I=1,NUM,2
800      IF( (IKIND(I) .NE. 1 .OR. IKIND(I+1) .NE. 1) GO TO 1301
801      M=IPARM(I)
802      1304 NPOS(M)=IPARM(I+1)
803      DO 1305 I=1,8
804      IF(IRATE(I) .GT. 600) IRATE(I)=600
805      1305 IF(IRATE(I) .LT. 25) IRATE(I)=25
806      CALL DRIVER
807      GO TO 5
808      1400 IF(ANS(2) .EQ. 'N') GO TO 1405
809      IF(ANS(2) .EQ. 'F') GO TO 1420
810      WRITE(6,8020)
811      GO TO 5
812      1405 WRITE(6,8055)
813      8055 FORMAT(' WHICH MOTORS SHOULD BE ON?')
814      NUM=0
815      CALL PARM(IPARM,IKIND,5,NUM)
816      IF(NUM .GT. 0 .AND. NUM .LE. 8) GO TO 1410
817      1409 WRITE(6,8020)
818      GO TO 1405
819      1410 PULSE=16384
820      DO 1411 I=1,8
821      1411 JTEST(I)=0
822      DO 1415 I=1,NUM
823      IF( (IKIND(I) .NE. 1) GO TO 1409
824      M=IPARM(I)
825      JTEST(M) = 1
826      CALL SETBF(PULSE,M+8)
827      1415 CONTINUE
828      INLINE
829      LDM,2 PULSE
830      OCA,2,7
831      LDM,1 #7F
832      STS,2,3 TELL SI STATUS WORD
833      FINI
834      GO TO 5
835      1420 PULSE=16639
836      WRITE(6,8050)
837      8050 FORMAT(' WHICH MOTORS SHOULD BE OFF?')
838      NUM=0
839      CALL PARM(IPARM,IKIND,5,NUM)
840      IF(NUM .GT. 0 .AND. NUM .LE. 8) GO TO 1430
841      1425 WRITE(6,8020)
842      GO TO 1420
843      1430 IF( (IKIND(1) .NE. 2) GO TO 1431
844      PULSE=16384
845      DO 1434 I=1,8
846      1434 JTEST(I) = 0
847      GO TO 1436
848      1431 DO 1432 I=1,8
849      1432 JTEST(I)=1
850      DO 1435 I=1,NUM
851      IF( (IKIND(I) .NE. 1) GO TO 1425

```



```

852      M=IPARM(I)
853      JTEST(M) = 0
854      CALL SETBT(PULSE,M+8)
855 1435 CONTINUE
856 1436 CONTINUE
857      INLINE
858          LDM,2          PULSE
859          OCA,2,7
860          LDM,1          #7F
861          STS,2,3        TELL SI STATUS WORD
862      FINI
863      GO TO 5
864 1500 WRITE(6,8040)
865 8040 FORMAT(1H,'SELECT MOTOR AND NEW POSITION')
866      NUM=16
867      CALL PARM(IPARM,IKIND,5,NUM)
868      IF(NUM.GT. 0 .AND. NUM.LE. 16) GO TO 1502
869 1501 WRITE(6,8020)
870      GO TO 1500
871 1502 IF(MOD(NUM,2)) 1501,1503,1501
872 1503 DO 1504 I=1,NUM,2
873      IF(IKIND(I).NE. 1 .OR. IKIND(I+1).NE. 1) GO TO 1501
874      M=IPARM(I)
875 1504 CPOS(M)=IPARM(I+1)
876      GO TO 5
877 1800 WRITE(6,8045)
878 8045 FORMAT(1H,'SELECT MOTOR AND NEW RATE')
879      NUM=16
880      CALL PARM(IPARM,IKIND,5,NUM)
881      IF(NUM.GT. 0 .AND. NUM.LE. 16) GO TO 1802
882 1801 WRITE(6,8020)
883      GO TO 1800
884 1802 IF(MOD(NUM,2)) 1801,1803,1801
885 1803 DO 1804 I=1,NUM,2
886      IF(IKIND(I).NE. 1 .OR. IKIND(I+1).NE. 1) GO TO 1801
887      M=IPARM(I)
888 1804 IRATE(M)=IPARM(I+1)
889      GO TO 5
890      END
891      SUBROUTINE OV1LY(N,NAME,LENGTH,IADD,NERR)
892      GO TO (5,10,15,20,25,30,35,40,45,50),N
893      5 CALL RUN
894      GO TO 100
895      10 CALL RESUME
896      GO TO 100
897      15 CALL STOP
898      GO TO 100
899      20 CALL PRINT
900      GO TO 100
901      25 CALL DELAY
902      GO TO 100
903      30 CALL ENABLE
904      GO TO 100
905      35 CALL DISABLE
906      GO TO 100
907      40 CALL SAVE(NAME,LENGTH,IADD,NERR)
908      GO TO 100
909      45 CALL FNC
910      GO TO 100
911      50 CALL HRDCPY
912 100 RETURN
913      END
914      SUBROUTINE OV2LY(N,ISPEC)
915      GO TO (5,10,15,20,25), N
916      5 CALL MOTOR
917      GO TO 100
918      10 CALL UP
919      GO TO 100
920      15 CALL DOWN
921      GO TO 100
922      20 CALL SPEC(ISPEC)
923      GO TO 100
924      25 CALL TRANS(ISPEC)
925 100 RETURN
926      END
927      SUBROUTINE OV3LY(N)
928      GO TO (5,10,15,20,25,30,35,40), N

```

```

929      5 CALL STATUS
930      GO TO 100
931      10 CALL ZERO
932      GO TO 100
933      15 CALL IDENT
934      GO TO 100
935      20 CALL INITL
936      GO TO 100
937      25 CALL INCR
938      GO TO 100
939      30 CALL LOOP
940      GO TO 100
941      35 CALL SE'LOOP
942      GO TO 100
943      40 CALL SEQU
944      100 RETURN
945      END
946      SUBROUTINE PRINT
947      DIMENSION SCALER(5)
948      INTEGER HEDREC(100),ITIME(6)
949      LOGICAL HDCPY
950      COMMON/HEADER/HEDREC
951      COMMON/HARD/HDCPY
952      EQUIVALENCE (HEDREC(32),SCALER(1)),(HEDREC(18),ITIME(1))
953      CALL TIMDAT(ITIME)
954      NU=HEDREC(17)
955      C...
956      C... OUTPUT STATUS WORD
957      C...
958      INLINE
959      LDI,2 #41A8
960      OCA,2,6
961      LDM,1 #7F
962      STS,2,1
963      FINI
964      ND=6
965      DO 20 I=1,NU
966      IF(I.EQ. 2) ND=7
967      SCALER(I)=0.
968      DO 10 J=1,ND
969      INLINE
970      REX,#14 DELAY 1 TICK
971      DFC #8000
972      DFC 1
973      IDA,2,6
974      STM,2 IDATA
975      FINI
976      SCALER(I)=FLOAT(IDATA)*(10.)**(ND-J)+SCALER(I)
977      10 CONTINUE
978      ND=6
979      20 CONTINUE
980      SCALER(3)=SCALER(3)/10.
981      WRITE(6,8000)(I,SCALER(I),I=1,NU)
982      8000 FORMAT(1H 'SCALER ',I3,' = ',E20,7)
983      IF(HDCPY) WRITE(7,8000)(I,SCALER(I),I=1,NU)
984      RETURN
985      END
986      SUBROUTINE RESUME
987      LOGICAL HDCPY
988      COMMON/HARD/HDCPY
989      INLINE
990      LDM,1 #7F
991      TRSB,3,1 SET
992      LDI,2 #40C0
993      HOP, OUT
994      SET LDI,2 #50C0
995      OUT OCA,2,6
996      STS,2,1
997      FINI
998      WRITE(6,8000)
999      8000 FORMAT(1H 'MICROPROCESSOR RUNNING'//)
1000      IF(HDCPY) WRITE(7,8000)
1001      RETURN
1002      END
1003      SUBROUTINE RUN
1004      LOGICAL HDCPY
1005      COMMON/HARD/HDCPY
1006      INLINE
1007      LDI,2 #4028
1008      OCA,2,6
1009      REX,#14 DELAY 1 TICK
1010      DFC #8000
1011      DFC 1
1012      LDM,1 #7F
1013      TRSB,3,1 SET

```

```

1014      LDI,2      #40C0
1015      HOP,      OUT
1016      SET      LDI,2      #50C0
1017      OUT      OCA,2,6
1018      STS,2,1
1019      FINI
1020      WRITE(6,8000)
1021      8000      FORMAT(1H,'MICROPROCESSOR RUNNING'//)
1022      IF(HDCPY) WRITE(7,8000)
1023      RETURN
1024      END
1025      SUBROUTINE SETLOOP
1026      COMMON/LOOPS/ILOOP,ISEQ
1027      WRITE(6,8000)
1028      8000      FORMAT(' HOW MANY TIMES SHOULD PROGRAM LOOP (I2)?')
1029      READ(5,8005) ILOOP
1030      8005      FORMAT(I2)
1031      RETURN
1032      END
1033      SUBROUTINE SEQU
1034      INTEGER HEDREC(100)
1035      LOGICAL HDCPY
1036      COMMON/HEADER/HEDREC
1037      COMMON/HARD/HDCPY
1038      COMMON/LOOPS/ILOOP,ISEQ
1039      WRITE(6,8000) ISEQ
1040      8000      FORMAT(' CURRENT SEQUENCE NUMBER = ',1X,I6)
1041      IF(HDCPY) WRITE(7,8000) ISEQ
1042      WRITE(6,8005)
1043      8005      FORMAT(' ENTER NEW SEQ. NO. (I6)')
1044      READ(5,8010) ISEQ
1045      8010      FORMAT(I6)
1046      IF(HDCPY) WRITE(7,8015) ISEQ
1047      8015      FORMAT(1H,'NEW SEQ. NO. = ',I6)
1048      HEDREC(1)=ISEQ
1049      RETURN
1050      END
1051      SUBROUTINE SPEC(ISPEC)
1052      DIMENSION NPOS(8),IRATE(8),JTEST(8)
1053      INTEGER CPOS(8),ANS(3),PULSE
1054      LOGICAL IENTRY,HDCPY
1055      COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
1056      COMMON/HARD/HDCPY
1057      DATA IENTRY/.FALSE./
1058      IF(ISPEC.LT. 0) GO TO 4
1059      IF(IENTRY) GO TO 5
1060      4      WRITE(6,8000)
1061      8000      FORMAT(' SET SPECIMEN WHEEL TO POSITION "0" '//
1062      X1H,' TYPE "GO" WHEN READY ')
1063      READ(5,8005) ANS
1064      8005      FORMAT(3A1)
1065      IENTRY = .TRUE.
1066      INLINE
1067      LDM,1      #7F
1068      OBSM,15,3      TURN ON MOTOR 8
1069      LDS,2,3
1070      OCA,2,7
1071      LBR,2,15      GO
1072      ODA,2,7      STEP MOTOR TILL LIMIT SWITCH SET
1073      ISA,2,7
1074      TBRB,2,15      ZERO
1075      REX,14
1076      DFC,1      #8000
1077      DFC,1
1078      BRU,1      GO
1079      ZERO      EQU,1      $
1080      FINI
1081      CPOS(8) = 0
1082      5      CONTINUE
1083      INLINE
1084      LDM,1      #7F
1085      OBSM,15,3      TURN ON MOTOR 8
1086      LDS,2,3
1087      OCA,2,7
1088      FINI
1089      NPOS(8) = -ISPEC*10 - 2
1090      CALL DRIVER
1091      INLINE
1092      STEP      LBR,2,15
1093      ODA,2,7
1094      ISA,2,7
1095      TBRB,2,15      THERE
1096      REX,14      DELAY 1 TIC
1097      DFC,1      #8000
1098      DFC,1
1099      BRU,1      STEP

```

```

1100 THERE EQU $
1101 LDM,1 #7F
1102 ZBSM,15,3 TURN OFF MOTOR 8
1103 LDS,2,3
1104 OCA,2,7
1105 FINI
1106 NPOS(8)=NPOS(8)+2
1107 WRITE(6,8015) ISPEC
1108 8015 FORMAT(' SPECIMEN # ',I3,' SELECTED ')
1109 IF(HDCPY) WRITE(7,8015) ISPEC
1110 RETURN
1111 END
1112 SUBROUTINE STATUS
1113 DIMENSION IDATA(64,64),JBUFF(200),KBUFF(200)
1114 DIMENSION JDATA(4100),ITIME(6)
1115 DIMENSION SCALER(5)
1116 INTEGER HEDREC(100)
1117 INTEGER ANS(3)
1118 LOGICAL HDCPY
1119 COMMON/DATA/JDATA
1120 COMMON/HEADER/HEDREC
1121 COMMON/OVER/NOVRF,JBUFF,KBUFF
1122 COMMON/HARD/HDCPY
1123 EQUIVALENCE (HEDREC(52),TSAMP),(HEDREC(32),SCALER(1))
1124 EQUIVALENCE (M,HEDREC(11)),(N,HEDREC(12)),(JDATA(5),IDATA(1,1))
1125 EQUIVALENCE (HEDREC(18),ITIME(1)),(HEDREC(54),ABSINT)
1126 EQUIVALENCE (WAVE,HEDREC(56)),(NSUB,HEDREC(24)),(NU,HEDREC(17))
1127 TOT=0
1128 IF(ITIME(6).EQ.0) CALL TIMDAT(ITIME)
1129 IMIN=MOD(ITIME(1),60)
1130 IHR=(ITIME(1)-IMIN)/60
1131 ITIC=MOD(ITIME(2),ITIME(6))
1132 ISEC=(ITIME(2)-ITIC)/ITIME(6)
1133 DO 5 J=1,N
1134 DO 5 I=1,M
1135 TEMP=FLOAT(IDATA(I,J))
1136 IF(TEMP.LT.0.) TEMP=65536.+TEMP
1137 5 TOT=TOT+TEMP
1138 WRITE(6,8001)
1139 8001 FORMAT(1H1)
1140 NAME = HEDREC(3)
1141 CALL CTA(NAME,ANS)
1142 WRITE(6,8000) HEDREC(1),ANS
1143 8000 FORMAT(' STATUS INFORMATION FOR',10X,'OWNER ID'/
1144 X' SEQUENCE NO.',2X,I6,14X,3A1)
1145 WRITE(6,8005)(HEDREC(1),I=76,100)
1146 8005 FORMAT(1H,'TITLE:',25A2)
1147 WRITE(6,8006) HEDREC(20),HEDREC(21),HEDREC(22)
1148 8006 FORMAT(1H,'EXPERIMENT PERFORMED ON:',I2,'-',I2,'-',I2)
1149 WRITE(6,8041) IHR,IMIN,ISEC
1150 8041 FORMAT(1H,'THE EXPERIMENT WAS COMPLETED AT ',I2,':',I2,':',I2//)
1151 WRITE(6,8007) TSAMP,ABSINT
1152 8007 FORMAT(1H,'TSAMP = ',F6.3,5X,'ABSINT = ',E15.5)
1153 WRITE(6,8010) M,N
1154 8010 FORMAT(1H,'THE DATA ARRAY DIMENSIONS ARE',I4,' BY',I4)
1155 WRITE(6,8015) TOT,NSUB
1156 8015 FORMAT(1H,'A TOTAL OF ',E10.4,' PHOTONS HAVE BEEN RECORDED'/
1157 X1H,'USING DATA TRANSFER ROUTINE ',I3)
1158 HEDREC(7)=NOVRF
1159 WRITE(6,8020) NOVRF
1160 8020 FORMAT(1H,'I3',' OVERFLOW BUFFERS WERE WRITTEN TO DISC')
1161 WRITE(6,8025)(SCALER(I),I=1,NU)
1162 8025 FORMAT(1H,'BEAM MONITOR COUNTS = ',F10.0/
1163 X1H,'INTEGRATED INTENSITY = ',F10.0/
1164 X1H,'ELAPSED TIME = ',F10.1,' SECONDS'//)
1165 IF (HDCPY) WRITE(7,8000) HEDREC(1),ANS
1166 IF (HDCPY) WRITE(7,8005) (HEDREC(1),I=76,100)
1167 IF (HDCPY) WRITE(7,8006) HEDREC(20),HEDREC(21),HEDREC(22)
1168 IF (HDCPY) WRITE(7,8041) IHR,IMIN,ISEC
1169 IF (HDCPY) WRITE(7,8007) TSAMP,ABSINT
1170 IF (HDCPY) WRITE(7,8010) M,N
1171 IF (HDCPY) WRITE(7,8015) TOT,NSUB
1172 IF (HDCPY) WRITE(7,8020) NOVRF
1173 IF (HDCPY) WRITE(7,8025) (SCALER(I),I=1,NU)
1174 RETURN
1175 END
1176 SUBROUTINE STOP
1177 LOGICAL HDCPY
1178 COMMON/HARD/HDCPY
1179 INLINE
1180 LDI,2 #40A8
1181 OCA,2,6
1182 LDM,1 #7F
1183 STS,2,1
1184 FINI

```

```

1185 WRITE(6,8000)
1186 8000 FORMAT(1H , 'HALT MICROPROCESSOR'//)
1187 IF(HDCPY) WRITE(7,8000)
1188 RETURN
1189 END
1190 SUBROUTINE TRANS(1SPEC)
1191 DIMENSION SCAL1(3),SCAL2(3),NPOS(8),IRATE(8)
1192 DIMENSION JTEST(8)
1193 INTEGER CPOS(8),HEDREC(100)
1194 LOGICAL HDCPY
1195 COMMON/HEADER/HEDREC
1196 COMMON/MUTR/CPOS,NPOS,IRATE,JTEST
1197 COMMON/HARD/HDCPY
1198 EQUIVALENCE (TSAMP,HEDREC(52)),(HEDREC(32),SCAL1(1))
1199 CALL DOWN
1200 CALL SPEC(0)
1201 NU = 3
1202 INLINE
1203 LDI,2 #4028
1204 OCA,2,6
1205 LDI,2 #40E8 START SCALERS
1206 OCA,2,6
1207 REX,#14
1208 DFC #8003
1209 DFC #7D0
1210 LDI,2 #40A8 STOP SCALERS
1211 OCA,2,6
1212 LDI,2 #41A8
1213 OCA,2,6
1214 LDM,1 #7F
1215 STS,2,1
1216 FINI
1217 ND=6
1218 DO 20 I=1,NU
1219 IF(I.EQ. 2) ND=7
1220 SCAL1(I) = 0.0
1221 DO 10 J=1,ND
1222 INLINE
1223 REX,#14 DELAY 1 TICK
1224 DFC #8000
1225 DFC 1
1226 IDA,2,6
1227 STM,2 IDATA
1228 FINI
1229 SCAL1(I) = FLOAT(IDATA)*(10.)**(ND-J) + SCAL1(I)
1230 10 CONTINUE
1231 ND=6
1232 20 CONTINUE
1233 SCAL1(3) = SCAL1(3)/10.
1234 WRITE(6,8000)
1235 8000 FORMAT(1H , 'SCALER VALUE')
1236 IF(HDCPY) WRITE(7,8000)
1237 WRITE(6,8001) (I,SCAL1(I),I=1,NU)
1238 8001 FORMAT(1H ,13,5X,F11.1)
1239 IF(HDCPY) WRITE(7,8001) (I,SCAL1(I),I=1,NU)
1240 CALL SPEC(1SPEC)
1241 INLINE
1242 LDI,2 #4028
1243 OCA,2,6
1244 LDI,2 #40E8 START SCALERS
1245 OCA,2,6
1246 REX,#14
1247 DFC #8003
1248 DFC #7D0
1249 LDI,2 #40A8 STOP SCALERS
1250 OCA,2,6
1251 LDI,2 #41A8
1252 OCA,2,6
1253 LDM,1 #7F
1254 STS,2,1
1255 FINI
1256 ND=6
1257 DO 30 I=1,NU
1258 IF(I.EQ. 2) ND=7
1259 SCAL2(I) = 0.0
1260 DO 40 J=1,ND
1261 INLINE
1262 REX,#14 DELAY 1 TICK
1263 DFC #8000
1264 DFC 1
1265 IDA,2,6
1266 STM,2 IDATA
1267 FINI
1268 SCAL2(I) = FLOAT(IDATA)*(10.)**(ND-J) + SCAL2(I)
1269 40 CONTINUE

```

```

1270 ND=6
1271 30 CONTINUE
1272 SCAL2(3) = SCAL2(3)/10.
1273 WRITE(6,8000)
1274 IF(HDCPY) WRITE(7,8000)
1275 WRITE(6,8001) (I,SCAL2(I),I=1,NU)
1276 IF(HDCPY) WRITE(7,8001) (I,SCAL2(I),I=1,NU)
1277 CALL UP
1278 TSAMP=SCAL2(2)*SCAL1(1)/(SCAL2(1)*SCAL1(2))
1279 WRITE(6,8002) ISPEC,TSAMP
1280 8002 FORMAT(1H,' THE TRANSMISSION OF SPECIMEN #',I3,' IS',F6.3)
1281 IF(HDCPY) WRITE(7,8002) ISPEC,TSAMP
1282 RETURN
1283 END
1284 SUBROUTINE UP
1285 DIMENSION NPOS(8),IRATE(8),JTEST(8)
1286 INTEGER CPOS(8)
1287 LOGICAL HDCPY
1288 COMMON/MOTR/CPOS,NPOS,IRATE,JTEST
1289 COMMON/HARD/HDCPY
1290 INLINE
1291 LDM,1 #7F
1292 ORSM,14,3 TURN ON MOTOR 7
1293 LDS,2,3
1294 OCA,2,7
1295 REX,#14 WAIT FOR MOTOR POWER
1296 DFC #8000
1297 DFC 100
1298 FINI
1299 NPOS(7) = 200
1300 CALL DRIVER
1301 INLINE
1302 STEP LBR,2,6
1303 ODA,2,7
1304 ISA,2,7
1305 TBR,2,6 THERE
1306 REX,#14
1307 DFC #8000
1308 DFC 1
1309 BRU STEP
1310 THERE EQU $
1311 LDM,1 #7F
1312 ZBSM,14,3 TURN OFF MOTOR 7
1313 LDS,2,3
1314 OCA,2,7
1315 FINI
1316 CPOS(7) = 200
1317 WRITE(6,8000)
1318 8000 FORMAT(' GLASSY CARBON OUT OF BEAM')
1319 IF(HDCPY) WRITE(7,8000)
1320 RETURN
1321 END
1322 SUBROUTINE ZERO
1323 DIMENSION SCALER(5)
1324 DIMENSION IDATA(64,64),JDATA(4100),JBUFF(200),KBUFF(200)
1325 INTEGER ANS(3),BLNK,HEDREC(100)
1326 DIMENSION RHEAD(25),ISETUP(100)
1327 LOGICAL HDCPY,INIFLG
1328 COMMON/DATA/JDATA
1329 COMMON/OVER/NOVRF,JBUFF,KBUFF
1330 COMMON/HEADER/HEDREC
1331 COMMON/HARD/HDCPY
1332 COMMON/INITL/INIFLG,ISETUP
1333 EQUIVALENCE (HEDREC(32),RHEAD(1)),(M,HEDREC(11)),(N,HEDREC(12))
1334 EQUIVALENCE (JDATA(5),IDATA(1,1))
1335 IF(INIFLG) GO TO 25
1336 WRITE(6,8030)
1337 8030 FORMAT(' INITIALIZE MICROPROCESSOR BEFORE ZEROING ARRAYS')
1338 IF(HDCPY) WRITE(7,8030)
1339 GO TO 40
1340 25 WRITE(6,8020)
1341 8020 FORMAT(1H,' DO YOU WISH TO CLEAR THE HEADER RECORD?')
1342 READ(5,8010) ANS
1343 IF(ANS(1).EQ.'Y') GO TO 35
1344 IF(ANS(1).EQ.'N') GO TO 30
1345 WRITE(6,8015)
1346 GO TO 25
1347 35 DO 45 I=1,100
1348 45 HEDREC(I)=ISETUP(I)
1349 8025 FORMAT(1H,' HEADER RECORD INITIALIZED')
1350 IF(HDCPY) WRITE(7,8025)
1351 30 WRITE(6,8005)
1352 8005 FORMAT(1H,' DO YOU WISH TO ZERO THE DATA ARRAY')
1353 X' AND OVERFLOW BUFFERS?')
1354 READ(5,8010) ANS
1355 8010 FORMAT(3A1)

```

```

1356      IF(ANS(1) .EQ. 'Y') GO TO 20
1357      IF(ANS(1) .EQ. 'N') GO TO 40
1358      WRITE(6,8015)
1359      8015 FORMAT(1H,'INVALID ... PLEASE TRY AGAIN')
1360      GO TO 30
1361      20 DO 5 J=1,N
1362         DO 5 I=1,M
1363           5 IDATA(I,J)=0
1364           JDATA(1) = -1
1365           JDATA(2) = HEDREC(24)
1366           JDATA(3) = HEDREC(11)
1367           JDATA(4) = HEDREC(12)
1368           INLINE
1369             LDI,3      JBUFF
1370             STM,J      #6F
1371             FINI
1372           DO 10 I=1,200
1373             JBUFF(I)=0
1374           10 KBUFF(I)=0
1375             NOVRF=0
1376             WRITE(6,8000)
1377           8000 FORMAT(1H,'DATA ARRAY AND OVERFLOW BUFFER CLEARED')
1378             IF(HDCPY) WRITE(7,8000)
1379           40 RETURN
1380           END
EXIT
$AUR CI 4
$END DO

```

*C MODCOMP SOURCE EDITOR
RESOURCE BLOCK FOR SAX TASK

```

1      ITL      RESOURCE BLOCK FOR SAX TASK
2      INS      SO,M3TWA
3      RESOURCES SAX,NON,3,0,100,128
4      TFILE    5,A13
5      TFILE    6,A03
6      TFILE    7,LP2
7      TFILE    RAD,ARA
8      TFILE    DAT,XDA
9      TFILE    20,OTI
10     TFILE    OTI,OTI
11     TFILES 2
12     ENDRESOURCES
EXIT
$AUR CI 4
$END DO

```


APPENDIX D

EXAMPLES OF THE USE OF PROGRAM SAX

III/SAX/
 XSAX/EST
 XASS 5=TY 6=TY
 XACT
 THE ORNL SMALL-ANGLE SCATTERING MONITOR
 VERSION A.02 24 SEPTEMBER 1979
 IS NOW OPERATING

THE DATE IS 10- 8-79
 THE TIME IS 11:45:41

DVF TASK NOW EXECUTING

MOT TASK NOW EXECUTING

CURRENT SEQUENCE NUMBER, = 0
 ENTER NE
 ENTER NEW SEQ. NO. (I6)
 2896
 SELECT FUNCTION

FNC
 THE FUNCTIONS ARE
 DEL = WAIT FOR INTERRUPT
 DIS = DISABLE SI
 DWN = PUT GLASSY CARBON IN THE BEAM
 ENA = ENABLE SI
 HC = PRINT HARD COPY OF COMMANDS
 ID = EXPERIMENT IDENTIFICATION
 FNC = TYPE FUNCTIONS
 INI = INITIALIZE EXPERIMENT
 INC = INCREMENT EXPERIMENT SEQUENCE NUMBER
 KIL = KILL PROGRAM
 LOO = LOOP INPUT INSTRUCTIONS
 LU = CHANGE LOGICAL UNIT FOR DATA STORAGE
 MOT = CALL MOTOR DRIVERS

ENTER ANY CHARACTER TO RESUME LIST
 GO
 NLO = SET NUMBER OF TIMES PROGRAM IS TO LOOP
 PRI = PRINT ORTEC SCALERS AND TIMERS
 RES = RESUME RUN
 RUN = START DATA COLLECTION
 SAV = SAVE THE CONTENTS OF CORE ON DISC
 SEQ = SET EXPERIMENT SEQUENCE NUMBER
 SXX = MOVE SPECIMEN XX ON ROTARY HOLDER
 STA = DISPLAY THE MACHINE STATUS
 STO = STOP RUN
 TXX = MEASURE TRANSMISSION COEFFICIENT OF SPECIMEN XX
 UP = TAKE GLASSY CARBON OUT OF THE BEAM
 WAI = PUT THIS PROGRAM INTO A WAIT STATE
 ZER = ZERO MEMORY ARRAY AND OVERFLOW BUFFERS

SELECT FUNCTION

INI
 ENTER ORIGIN OF DATA ARRAY (HEX)
 LAST BYTE MUST BE ZERO

F000

SELECT MICROPROCESSOR DATA TRANSFER ROUTINE(I2)

02

MICROPROCESSOR INITIALIZED

ENTER X-RAY WAVELENGTH (X.XXXX)

1.5417

HOW MANY SCALERS/TIMERS ? (I1)

3

ENTER SPECIMEN TO DETECTOR DISTANCE

(IN MM) (F8.3)

2126.

ENTER SPECIMEN TO SOURCE DISTANCE (IN MM),(F8.3)

3650.

ENTER X-AXIS CHANNELS TO MM CONVERSION FACTOR(F8.3)

0.324

SELECT FUNCTION

ZERO

DO YOU WISH TO CLEAR THE HEADER RECORD?

YES

DO YOU WISH TO ZERO THE DATA ARRAY
AND OVERFLOW BUFFERS?

YES

DATA ARRAY AND OVERFLOW BUFFER CLEARED

SELECT FUNCTION

ID

ENTER OWNER ID

RWH

ENTER TRANSMISSION COEFFICIENT (F5.0)

0.376

ENTER ABSOLUTE INTENSITY FACTOR (X.XXXEYY)

1.0

ENTER TITLE AND COMMENTS (25A2)

TEST EXAMPLE FOR USERS

SELECT FUNCTION

RUN

MICROPROCESSOR RUNNING

SELECT FUNCTION

PRINT

SCALER 1 = 0.3200000E+02

SCALER 2 = 0.1906900E+05

SCALER 3 = 0.1000000E+02

SELECT FUNCTION

STATUS

STATUS INFORMATION FOR OWNER ID

SEQUENCE NO. 2896 RWH

TITLE: TEST EXAMPLE FOR USERS

EXPERIMENT PERFORMED ON: 8-10-79

THE EXPERIMENT WAS COMPLETED AT 11:49:14

TSAMP = 0.376 ABSINT = 0.10000E+01

THE DATA ARRAY DIMENSIONS ARE 64 BY 64

A TOTAL OF 0.1906E+05 PHOTONS HAVE BEEN RECORDED
USING DATA TRANSFER ROUTINE 2

0 OVERFLOW BUFFERS WERE WRITTEN TO DISC

BEAM MONITOR COUNTS = 32.

INTEGRATED INTENSITY = 19069.

ELAPSED TIME = 10.0 SECONDS

SELECT FUNCTION

SAVE

DATA WILL BE STORED UNDER THE SEQUENCE NO. 2896

ON LOGICAL UNIT = 21

SELECT FUNCTION

APPENDIX E

PROCEDURES FOR COMPILING, LINK-EDITING,
AND CATALOGUING PROGRAM SAX

```

JOB
$
ASS LO=NO
$
COMPILE SAX
$
JOB
$
ASS BI=BSA BO=CSA
$
REW BI BO
$
EXE EDIT
//LINK EDITOR// /B /EDI
GCOM OVF #101 #191
GCOM MOTR #292 #20
GCOM HEADER #2B2 #64
GCOM DATA #EFFF #1004
LOCALL OV1LY,OV2LY,OV3LY
EDIT MAIN,BI

```

*N LINK EDITOR MEMORY MAP DATE 10/08/79

```

C OVF      0101 A
C MOTR     0292 A
C HEADER   02B2 A
C DATA    EFFF A
P SAX      0000      028D
C HARD     0000
C OVER     0001
C MANAGE   0192
C INITL    0196
C LOOPS    01FB
C AAAS$    01FD
C LBF:     020D
  RTP::    029A
  BRX::    02A7
LC OV1LY   0838
  OV1LY    0838
LC OV2LY   083D
  OV2LY    083D
LC OV3LY   0842
  OV3LY    0842
P CORE     0847
  CORE     0847
P TIMDAT   0873
  TIMDAT   0873
P LOCAL:   0881
  LOCAL:   0881
P CAO:     08F4
  CAO:     08F4
P FPA:     092D
  FPA:     092D
P CFM:G    0951
  CFM:G    0951
P MOD:     0960
  MOD:     0960
P NII:     0971
  NII:     0971
P L:ERR    0973
  L:ERR    0973
P RDM:G    09C1
  TRV:     09D5

```

TRS: 09C1
 TRO: 09D3
 TRN: 09D1
 IWU: 09D9
 IWF: 09D7
 RFI: 09DB
 RRU: 09DD
 RFW: 09C9
 RRF: 09DF
 EOF: 09C7
 TAR: 09CF
 RWU: 09E1
 RWF: 09E3
 BRX: 09C3
 IRU: 09CD
 BKS: 09C5
 IRF: 09CB
 IRD: 09E5
 *HIGH 09E6
 *XFER 02CC

*N LINK EDITOR MEMORY MAP DATE 10/08/79

L LOCALL 09E7
 P OV1LY 09E7
 OV1LY 09E9
 P PRINT 0A47
 PRINT 0A49
 P RESUME 0B1C
 RESUME 0B1E
 P RUN 0B54
 RUN 0B56
 P STOP 0B92
 STOP 0B94
 P SAVE 0BC4
 SAVE 0C2A
 P PRI: 0F5B
 PRI: 0F5B
 P CIR: 0FA6
 FLO: 0FA6
 CIR: 0FA6
 P T\$OVR 0FAF
 C\$OVR 0FAF
 T\$OVR 0FBB
 P DELAY 0FD3
 DELAY 0FDB
 P DISABL 114B
 DISABL 1152
 P ENABLE 11EA
 ENABLE 11F1
 P FNC 1284
 FNC 1289
 P HRDCPY 14FE
 HRDCPY 1503
 *HIGH 161B
 *XFER 09E7

*N LINK EDITOR MEMORY MAP DATE 10/08/79

L	LOCALL	09E7
P	OV2LY	09E7
	OV2LY	09E9
P	SPEC	0A1C
	SPEC	0A22
P	TRANS	0AED
	TRANS	0AF5
P	UP	0D03
	UP	0D05
P	PRI:	0D54
	PRI:	0D54
P	CIR:	0D9F
	FLO:	0D9F
	CIR:	0D9F
P	T\$OVR	0DA8
	C\$OVR	0DA8
	T\$OVR	0DB4
P	DOWN	0DCC
	DOWN	0DCE
P	DRIVER	0E1C
	DRIVER	0E1E
P	MOTOR	0ECE
	MOTOR	0EF3
P	PARM	1458
	PARM	148B
P	COLLEC	15AD
	COLLEC	15AD
P	IACAN	15EF
	IACAN	15EF
P	IABIN	160F
	IABIN	160F
P	TESTB	165A
	TESTB	165A
P	SETBT	1679
	SETBT	1679
P	SETBF	1696
	SETBF	1696
P	IFI:	16B3
	INT:	16B3
	CRI:	16B3
	IFI:	16B3
	*HIGH	16DD
	*XFER	09E7

*N LINK EDITOR MEMORY MAP DATE 10/08/79

L	LOCALL	09E7
P	OV3LY	09E7
	OV3LY	09E9
P	SETLOO	0A23
	SETLOO	0A25
P	SEQU	0A5C
	SEQU	0A5E
P	STATUS	0AE1
	STATUS	0AE6
P	ZERO	0E40
	ZERO	0E4F
P	CTA	0FF4
	CTA	0FF4
P	RLT:	1006

```

      RLT:      1006
P  CIR:      1017
      FLO:      1017
      CIR:      1017
P  IDENT     1020
      IDENT     102F
P  INCR      117A
      INCR      117F
P  INITL     11C1
      INITL     11C4
P  LOOP      15E1
      LOOP      15E9
P  IACAN     1638
      IACAN     1638
P  MOVE      1658
      AVRECO    1662
      BKRECO    165F
      REWIND    1658
      WEOF      1668
      AVFILE    1665
      HOME      166B
      BKFILE    165C
P  GTUFT:    1682
      GTUFT:    1682

```

```
*HIGH 168F
```

```
*XFER 09E7
```

```
WEOF B0
```

```
EXIT
```

```
$
```

```
JOB
```

```
$
```

```
ASS LO=NO
```

```
$
```

```
ASS SO=MC
```

```
$
```

```
MACRO SAXRES
```

```
$
```

```
JOB
```

```
$
```

```
EXE LIB
```

```
LIBRARY UPDATE /B /LIB
```

```
ASS BI=CSA SI=BSA BO=CSB
```

```
REW BI BO SI
```

```
ADD 1
```


11-77

*E.1 MODCOMP LIBRARY UPDATE DATE 10/08/79 09:49:25 PAGE 1

1	P SAX						
	COUNTER 01	LAST 09E7	BOUNDARY 0001	SIZE 09E7	2535		
	TRANSFER ADDRESS 01 02CC		COMMON NONE				
2	P W\$SAX						
	COUNTER 01	LAST 016F	BOUNDARY 0001	SIZE 016F	367		
	TRANSFER ADDRESS NONE		COMMON NONE				
COPY							
3	P OV1LY						
	COUNTER 01	LAST 161C	BOUNDARY 0001	SIZE 161C	5660		
	TRANSFER ADDRESS 01 09E7		COMMON NONE				
4	P OV2LY						
	COUNTER 01	LAST 16DE	BOUNDARY 0001	SIZE 16DE	5854		
	TRANSFER ADDRESS 01 09E7		COMMON NONE				
5	P OV3LY						
	COUNTER 01	LAST 1690	BOUNDARY 0001	SIZE 1690	5776		
	TRANSFER ADDRESS 01 09E7		COMMON NONE				
TOTAL RECORDS WRITTEN = 138							
EXIT							

JOB
\$
EXE CAT
*I.1 DISC CATALOGER /B /CAT
FILE LM
ASS BI=CSB
REW BI
NOV
NO VERIFICATION MODE ENTERED /B /CAT
CAT SAX,SAX,PTASK,A.02,BUF=2,ADD=#CF7
CAT OV1LY,OV1,POVER,A.02
CAT OV2LY,OV2,POVER,A.02
CAT OV3LY,OV3,POVER,A.02
EXIT
\$

ORNL/TM-6342/V13

OAK
RIDGE
NATIONAL
LABORATORY

THE ORNL 10-m SMALL-ANGLE X-RAY SCATTERING CAMERA

Vol. 13. File Management Programs
(MANAGER, FETCH and SAVE)

R. L. Stephenson and R. W. Hendricks

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

THE ORNL 10-m SMALL ANGLE X-RAY SCATTERING CAMERA

13. FILE MANAGEMENT PROGRAMS (MANAGER, FETCH AND SAVE)

R. L. Stephenson¹ and R. W. Hendricks

ABSTRACT

FORTRAN IV software is described which permits the storage and retrieval of variable-length data sets in a self-defining, direct-access data base. A utility program provides data base management functions such as renaming or deleting entries, file compression, backup copies of either individual entries or an entire data base, and archival storage of data sets on other media.

1. INTRODUCTION

The ORNL 10-m small-angle x-ray scattering camera, described elsewhere² and in other volumes of this report, is capable of producing large quantities of variable-length data arrays via its two-dimensional position-sensitive detector and associated data acquisition hardware system. It is necessary to store these data sets as they are acquired and to retrieve them in random order for subsequent analysis and processing.

This report describes a general-purpose file management system which has been created to allow the user to store and retrieve information sets of varying length, without regard to their content, in a self-defining data base. Storage and retrieval are accomplished by two FORTRAN subroutines (FETCH and SAVE) which are called by the user's data collection and/or data processing software. A separate program (MANAGER) provides utility functions such as data set renaming and deletion, data base compression, data set migration, data base directories, and archival

¹Computer Sciences Division.

²R. W. Hendricks, *J. Appl. Crystallogr.* 11, 15-30 (1978).

storage and retrieval from off-line disc packs. Certain of these operations are privileged and may be executed only by the owner of a data set or by authorized systems personnel.

Although it might be more desirable to pass data between the program modules and the data base via argument lists, for reasons associated with the hardware of the small-angle camera, our experimental data must reside in global common arrays. It would be a very simple task to modify the software described here to permit this, thus making our data base management system attractive for application in a wide range of similar on-line data acquisition and analysis programs.

II. DATA STRUCTURES

The data base is a self-defining, direct-access data-base having an organization as shown schematically in Fig. 1. The storage of actual data begins with the first record of the data base and proceeds forward while the index begins with the last record of the data base and proceeds backward. This is done so that one need not allocate a specific quantity of space to the index. As a result, the data and the index are extended until they meet without wasted space regardless of whether a large number of short data sets require a long index or vice versa.

An index record is shown schematically in Fig. 1. Fundamentally, each data set is represented by a two-word entry in the index consisting of a data set name and the starting record number of the data set. For the purpose of SAX³ the data set names are sequential integer numbers. However, it is straightforward to utilize three-character names and generate their CAN-code⁴ integer representation. The data base is initialized by inserting -9999 into the first word of the index. When the storage of a data set is attempted, this word is checked. If the data base has not been initialized, the subroutine does the initialization, inserts

³R. W. Hendricks, *The ORNL 10-m Small-Angle X-Ray Scattering Camera*. V. 11. *Camera Control Programs (SAX, OVF, and MOT)*, ORNL/TM-6342.

⁴Compressed Alpha Numeric, a selected set of 40 characters which are a subset of the ASCII characters. An algorithm allows the packing of three characters per 16-bit word, as opposed to only two ASCII characters.

Fig. 1. Schematic Diagram of the Direct-Access Data Base.

the data set name and its starting record number in words three and four of the index record, and stores the data beginning at the first record in the data base.

The integer -8888 is used as a terminator for the index. Each time a data set is stored, its data set name and starting address are stored in two consecutive words of the index, after which -8888 is inserted as a data set name in the succeeding word followed by the address of the next free record which will, of course, be the starting address of the next data set to be stored. When a data base has been initialized, the SAVE subroutine searches for the -8888 which it replaces with the name of the incoming data set. Since the next free record will be the starting address of the new data set, the subroutine simply transfers the data and inserts -8888 followed by the address of the next free record in the next two words of the index.

Each data set begins with a header record similar to the one shown in Fig. 1. The first five words of the header are reserved for use by the file management system; the remainder can be used to characterize

the data set in any way the user desires. The structure of HEADER as used by the SAX program is given in detail elsewhere.³ The five reserved words are used as follows. The first word contains the data set name (or sequence number), the second word contains the length of the data set in records, the third word contains the identification of the data set owner (in CAN-code),⁴ the fourth word is used to identify the instrument on which the data were collected,⁵ and the fifth word is reserved for future expansion. The remainder of the record is available for any descriptive commentary. The user may nest subsets to any level by using the fifth word as a pointer or counter. It is necessary that the size of the array in global common, where the header record is stored, be the same length as the logical record length of the direct access file, while the size of the data array must be an integer multiple of that length.

III. PROGRAM MANAGER

MANAGER is a utility program which provides for housekeeping chores such as the listing of the data base table of contents, deletion of unneeded data sets, compression to recover unused space, and the creation and recovery of archival backup files.

MANAGER is a low-priority program which operates in the background under the control of the batch processor task B. The load module is stored on the background load module file (LMB) and is activated by the operator directive

\$EXE MAN .

(Before entering this directive, the user should be certain that he has exited from any previously executing background task!) MANAGER will respond with its name and current revision level and ask for the USER ID. The user should enter his three-character identification code which has been previously assigned. MANAGER then asks for the logical unit to be

⁵The instrument identification codes are as follows: 0 — 10-m SAXS camera; 1 — 30-m SANS camera; 2 — 5-m SANS camera; and 3 — Kratky SAXS camera.

processed. In the code for MANAGER, four direct access I/O logical units have been defined in order to allow for separate data bases for different cameras and for archival storage of data. These are described in Table 1.

Table 1. Defined FORTRAN Direct-Access I/O Files

Logical Unit	Record Size (words)	Number of Records	Use
21	100	4480	10-m SAXS camera; on-line data
22	100	576	Kratky camera; on-line data
31	100	4480	10-m SAXS camera; off-line data
32	100	576	Kratky camera; off-line data

The user should enter the logical unit of the data base to be examined. If the selected data base has not been initialized (first word of index = -9999) an appropriate message is printed and the program is terminated. Otherwise, MANAGER responds with

SELECT FUNCTION (? FOR PROMPT)

at which point the user may enter any of the allowable MANAGER directives. The following paragraphs give a narrative description of each command, its purpose and operation, while detailed information for user use is given in the tables of the next section.

SCAN — The scan directive produces a listing of the contents of the data base in the form illustrated as example 1. Each data set name is listed along with its starting record number and length. The last column lists the cumulative number of free tracks preceding the beginning of each data set. (Free tracks result when data sets have been moved or deleted.) This free space can be recovered with the COMPRESS command.

LIST — This directive produces a listing of all header records in the data base as illustrated in example 2.

SCAN OF DATA BASE NUMBER 22

SEQ. NO.	STARTING ADDRESS	LENGTH (TRACKS)	CUMULATIVE FREE TRACKS
102	1	11	0
113	13	11	0
86	25	11	0
104	37	11	0
105	49	11	0
106	61	11	0
107	73	11	0
108	85	11	0
109	97	11	0
110	109	11	0
111	121	11	0
112	133	11	0
114	145	11	0
116	157	11	0
117	169	11	0
118	181	11	0
119	193	11	0
101	205	11	0
102	217	11	0
136	229	11	0
137	241	11	0
138	253	11	0
139	265	11	0
142	301	11	22
143	313	11	22
144	325	11	22

Example 1

DSN	OWNER	LENGTH	TITLE
108	SWD	11	P-9 3.3% DATE: 9-25-79 TIME: 21: 2:42 TSAMP = 0.561 ABSINT = -0.25729E-66 MON = 0. TOT = 240075. TIME = 16840.0
109	SWD	11	P-8 VOL FRACT-.66 DATE: 9-26-79 TIME: 0:16:50 TSAMP = 0.565 ABSINT = -0.25729E-66 MON = 0. TOT = 51193. TIME = 2978.0
110	SWD	11	P-8 .66 DATE: 9-26-79 TIME: 8: 7: 8 TSAMP = 0.565 ABSINT = -0.25729E-66 MON = 0. TOT = 496156. TIME = 30585.0
111	SWD	11	P-11 1% DATE: 9-26-79 TIME: 11:37:31 TSAMP = 0.578 ABSINT = -0.25729E-66 MON = 0. TOT = 96618. TIME = 9278.0
112	SWD	11	P-10 1.2% PS IN MEK DATE: 9-26-79 TIME: 12:44:25 TSAMP = 0.578 ABSINT = -0.25729E-66 MON = 0. TOT = 41696. TIME = 3980.0

Example 2

This allows the user to read the commentary (example 2) and certain critical data he has included in the header records. It is the only command which produces an output which depends on the data beyond word 5 in the header record. The format of this printout can, of course, be modified to suit the user's data structure.

DELETE — This directive allows the user to delete any data set. When the command is entered, the program will ask for the data set name. The program simply replaces the data set name in the index with -7777, thus rendering this data set inaccessible under the old name. No overwriting or movement of data sets occurs at this point. Space occupied by the deleted data will be left vacant until a COMPRESS, described below, is done.

RENAME — A data set can be renamed by entering this directive. The program will prompt for the old name, then for the new name after which it updates both the index and the header record.

MOVE — The SAVE subroutine makes no attempt to use fragmented space; it simply adds each new data set sequentially at the end of the list. Hence, if new information is to be added to an existing data set which is not last in the file, that data set must be moved to the last position. In response to the MOVE directive, the program will copy the appropriate data set at the end of all existing data sets, delete the data set name in the index, and then add it at the end of the index.

DUMP DATA BASE — It is desirable to dump the entire data base to create a backup. The program responds to this directive by asking for the logical unit number of the device which is to receive the copy and then copies the data base. This command writes a duplicate FORTRAN direct access I/O data set on the output logical unit.

DUMP DATA SET — This command causes the program to print the contents of a particular data set on the standard output device ($lu = 6$). It also writes the data set to $lu = 7$ with an unformatted write (i.e., a direct copy of the data set is made). Thus, $lu = 7$ must be assigned to a disc partition.

COMPRESS — After several data sets have been deleted, considerable vacant space may accumulate. By using the SCAN option the user can determine the amount of unused space. This space can be recovered by

using the COMPRESS directive. When unused space is encountered, all succeeding data sets will be moved forward to fill this space and accumulate all empty space at the end of the data, where it can be used for storing subsequent data sets. The compress is done in place.

OFF-LOAD and (ON-LOAD) — In experimental situations where large amounts of data are generated, it is rarely possible (or desirable) to maintain all the data on-line at any one time. These commands allow the user to automatically move data from an on-line to a mountable off-line disc, or vice versa. For convenience, we have chosen to make the off-line data bases the same size as the on-line ones (see Table 1). Thus, a dedicated mountable (off-line) disc pack contains many disc partitions of the appropriate sizes for each data base (see SYSGEN manual, Vol. 8 of this series) and *IT IS THE USER'S RESPONSIBILITY TO BE SURE* that the destination (OFF-LOAD) or source (ON-LOAD) disc partition has been assigned prior to entering MANAGER.

RESET LOGICAL UNIT — When program MANAGER begins execution, it asks the user to enter the logical unit number of the data base to be accessed. The RESET command causes the program to prompt for a new logical unit number. The program can, therefore, be used to maintain any number of separate data bases. A DEFINE FILE statement must be included in MANAGER for each file which it is to maintain.

RESTORE — This directive enables the user to recover a backup copy made with the DUMP command. The program asks for the logical unit number of the backup copy and copies it to the data base.

ZAP — If the user is finished with all information in a data base, he may destroy it by using the ZAP directive and begin anew. When this directive is entered, the program destroys the -9999 in the first word of the index which causes subroutine SAVE to reinitialize the data base and overwrite all previous information. Use of this directive writes a message to the operator and puts the program in HOLD (REX, HOLD) until the operator verifies the ZAP.

EXIT — The EXIT directive terminates the execution of program MANAGER (returns to B).

These MANAGER commands are summarized in Table 2 and are described in detail in Table 3. A source listing of the program is given in Appendix A.

IV. MANAGER OPERATOR COMMANDS

Table 2. Summary of MANAGER Commands

COMMAND*	PURPOSE
?	Prompt. A list of the table of commands is printed
<u>COMPRESS</u>	Compress the data base
<u>DBASE</u>	Dump data base
<u>DELETE</u>	Delete a data set
<u>DSET</u>	Dump data set
<u>EXIT</u>	Exit from MANAGER
<u>LIST</u>	List header records
<u>LUNIT</u>	Reset logical unit
<u>MOVE</u>	Move data set to last
<u>OFFLOAD</u>	Off-load data sets to archival disc
<u>ONLOAD</u>	On-load data sets from archival disc
<u>RENAME</u>	Rename data set
<u>RESTORE</u>	Restore data set
<u>SCAN</u>	Scan (display available space)
<u>ZAP</u>	Clear and initialize data base

*The commands may be abbreviated to the underlined characters. All commands must be terminated by a carriage return.

Table 3(a). User Commands for Program MANAGER

?	PROMPT
FUNCTION	To prompt the user by listing the primary MANAGER functions on the $\ell u = 6$ output device
MAN RESPONSE	A listing of the user commands is printed
USER RESPONSE	None
ERRORS	None

Table 3(b). User Commands for Program MANAGER

<u>COMPRESS</u>	
FUNCTION	Compress the data base in place to recover available space from deleted data sets
MAN RESPONSE	Compresses data base in place. On completion, types "SELECT FUNCTION"
NOTE	If the data base is nearly full and if a data set is deleted from near the beginning, the compress operation will take about 20 min
USER RESPONSE	Patience!
ERRORS	None

Table 3(c). User Commands for Program MANAGER

DBA (DUMP DATA BASE)	
FUNCTION	Copies entire data base to another disc partition for archival storage
MAN RESPONSE	'ENTER LOGICAL UNIT NUMBER TO RECEIVE DUMP'
USER RESPONSE	Enter logical unit (free format). Data are written unformatted. Thus, lu must be assigned to a disc partition.
NOTE	A disc-to-disc copy requires about 5 min.
ERRORS	None

Table 3(d). User Commands for Program MANAGER

<u>DELETE</u>	
FUNCTION	Deletes a data set from the data base. This is accomplished by changing the data set sequence number (ISEQ) to -7777 in both HEDREC and in the directory
MAN RESPONSE	'ENTER DATA SET NAME'
USER RESPONSE	Type in the experiment sequence number (free format)
ERRORS	<p>'DATA SET NOT FOUND' indicates that the desired data set was not found in the index. Check input against index listing with either LIST or SCAN.</p> <p>'DATA SET NOT YOURS.' Data sets may be deleted only by authorized system programmers.</p>

Table 3(e). User Commands for Program MANAGER

<u>DSE (DUMP DATA SET)</u>	
FUNCTION	Copy a single data set to another output device
MAN RESPONSE	'ENTER DATA SET NAME'
USER RESPONSE	Enter data set sequence number for data set to be dumped
MAN RESPONSE	'ENTER LOGICAL UNIT TO RECEIVE DUMP'
USER RESPONSE	<p>Enter logical unit (free format). Currently, only $lu = 6$ and 23 are defined. If $lu = 23$ is to be used, user must first assign it to an appropriate disc partition via operating system directive</p> <p style="text-align: center;">/B/ASS 23 = xxx</p> <p>where xxx is a disc partition large enough to receive the data set</p>
ERRORS	None

Table 3(f). User Commands for Program MANAGER.

<u>EXIT</u>	
FUNCTION	Exit from MANAGER
B RESPONSE	\$ (indicates B TASK is ready to accept a new command)

Table 3(g). User Commands for Program MANAGER

<u>LIST</u>	
FUNCTION	To list the header records
MAN RESPONSE	A listing of the header records in the form shown in Example 2 (p. 13-6) is produced.
USER RESPONSE	None
ERRORS	None

Table 3(h). User Commands for Program MANAGER

<u>LUNIT</u>	
FUNCTION	To select the FORTRAN direct-access data base which is to be examined
MAN RESPONSE	'WHICH LOGICAL UNIT IS TO BE PROCESSED?'
USER RESPONSE	Select logical unit
NOTES	The currently allowable units are 21, 22, 31, and 32 as given in Table 1. Units 21 and 22 are permanently assigned to disc partitions XDA and XDB, respectively, and <u>must not</u> be reassigned. Units 31 and 32 are unassigned, and must be assigned to appropriate disc partitions before use. These units are generally used for accessing the archival data bases.
ERRORS	None

Table 3(i). User Commands for Program MANAGER

<u>MOVE</u>	
FUNCTION	To move a data set to last (this permits concatenation of new data – see text)
MAN RESPONSE	'ENTER DATA SET NAME'
USER RESPONSE	Enter data set name (sequence number) of data to be moved
ERRORS	'DATA BASE FULL' insufficient room to accept data set at end of data base. A <u>SCAN</u> will indicate if enough space can be recovered with a <u>COMPRESS</u> .

Table 3(j). User Commands for Program MANAGER

OFFLOAD	
FUNCTION	To off-load data from the on-line data base to an off-line archival data base.
MAN RESPONSE	'ENTER SEQUENCE NUMBER OF FIRST AND LAST DATA SETS TO BE OFF-LOADED'
USER RESPONSE	<p>ISTART{, ISTOP} {, ALL} where ISTART is first sequence number to be off-loaded, ISTOP is optional and is the last sequence number to be off-loaded, and ALL is optional and determines if all data sets between ISTART and ISTOP or only those owed by USER are to be off-loaded. If ISTOP is missing, only data set ISTART is off-loaded.</p>
MAN RESPONSE	'MOUNT DATA DISC AND ASSIGN LU. THEN TYPE "/B/R" TO RESUME'
USER RESPONSE	<p>FORTTRAN direct-access logical unit LU + 10 must be assigned to the disc partition for the archival data base, and the archival disc pack must be mounted in device ML. <u>NOTE!</u> There is no scratch space on the system at this point, and <u>any</u> operation which uses scratch space will destroy the archival pack.</p> <p>Hit console interrupt and center /B/R</p>
MAN RESPONSE	<p>'O.K.!' Dismount data disc, then type "/B/R" to resume</p>
USER RESPONSE	<p>Dismount the archival data disc and remount the scratch pack. Hit console interrupt and center /B/R</p>
ERRORS	Any error code from subroutines FETCH and SAVE (see Sections V and VI)
NOTE	If a data set (sequence number) is not found, it is flagged, no data are transferred, and processing continues. All other errors are fatal.

Table 3(k). User Commands for Program MANAGER

ONLOAD	
FUNCTION	To on-load data from the archival off-line data base to the on-line data base.
MAN RESPONSE	'ENTER SEQUENCE NUMBER OF FIRST AND LAST DATA SETS TO BE ON-LOADED'
USER RESPONSE	<p>ISTART{, ISTOP} {,ALL} where ISTART is first sequence number to be on-loaded, ISTOP is optional and is the last sequence number to be on-loaded, and ALL is optional and determines if all data sets between ISTART and ISTOP or only those owed by USER are to be on-loaded. If ISTOP is missing, only data set ISTART is on-loaded.</p>
MAN RESPONSE	'MOUNT DATA DISC AND ASSIGN LU. THEN TYPE "/B/R" TO RESUME'
USER RESPONSE	<p>FORTTRAN direct-access logical unit LU + 10 must be assigned to the disc pack for the archival data base, and the archival disc pack must be mounted in device ML. <u>NOTE!</u> There is no scratch space on the system at this point, and <u>any</u> operation which uses scratch space will destroy the archival pack.</p> <p>Hit console interrupt and center /B/R</p>
MAN RESPONSE	<p>'O.K.!' Dismount data disc, then type "/B/R" to resume</p>
USER RESPONSE	<p>Dismount the archival data disc and remount the scratch pack. Hit console interrupt and center /B/R</p>
ERROR	Any error code from subroutines FETCH and SAVE (see Sections V and VI)
NOTE	If a data set (sequence number) is not found, it is flagged, no data are transferred, and processing continues. All other errors are fatal.

Table 3(1). User Commands for Program MANAGER

RENAME	
FUNCTION	To rename a data set
MAN RESPONSE	'ENTER DATA SET NAME'
USER RESPONSE	Enter data set sequence number (free format)
MAN RESPONSE	'ENTER NEW NAME'
USER RESPONSE	Enter new data set sequence number (free format)
ERRORS	'DATA SET NOT FOUND' 'DATA SET NOT YOURS' Data may be renamed <u>only</u> by authorized systems programmers.

Table 3(m). User Commands for Program MANAGER

<u>RESTORE</u>	
FUNCTION	Restore data base from a previously dumped data base
MAN RESPONSE	'ENTER LOGICAL UNIT NUMBER TO SUPPLY INPUT'
USER RESPONSE	Enter one of the assignable FORTRAN direct-access I/O logical units defined in Table 1.
NOTE	User must be sure the input logical unit is assigned to the disc partition containing the dumped data base before proceeding
ERRORS	None

Table 3(n). User Commands for Program MANAGER

SCAN	
FUNCTION	To produce a listing of the contents of the data base and to indicate unused space
MAN RESPONSE	A listing of the form shown in Example 1 (p. 13-6) is produced
USER RESPONSE	None
ERRORS	None

Table 3(o). User Commands for Program MANAGER

<u>ZAP</u>	
FUNCTION	Reinitialize data base (puts -9999 in first word of index)
MAN RESPONSE	'ONLY RWH CAN ZAP DATA BASE'
USER RESPONSE	<u>You must</u> hit console interrupt and enter /B/A

V. SUBROUTINE FETCH

Subroutine FETCH provides the calling program with the capability of retrieving data sets from the data base and loading them into memory. The calling sequence is

CALL FETCH (NAME, NERR)

where

NAME is the data set name, or sequence number, and may be any integer number except -7777, -8888, or -9999,

and

NERR is an error return code, and may have the following values:

0 — no error; data retrieval was successful,

2 — desired data set not found,

3 — data base not initialized (no data sets in file),

5 — data set longer than array provided for its return (see discussion below),

6 — reserved word used as data set name (-7777, -8888, or -9999).

In addition, information concerning the sizes of the data base and the data sets must be provided by the main program in three labeled commons as follows:

COMMON/MANAGE/MAXREC, MXWRDS, LU, IDEM
COMMON/HEADER/HEDREC
COMMON/DATA/IDATA

where

MAXREC is the maximum number of records in the direct-access file.

(Must match the first parameter in the DEFINE FILE statement for the logical unit in question.)

MXWRDS is the number of words per direct access record.

LU is the logical unit for the direct access file.

IDEM is the length of the array DATA, and is not necessarily the same as LENGTH (defined in the next section).

HEDREC is the header record and must be dimensioned MXWRDS;

and

IDATA is the data array to be stored and must be dimensioned IDEM.

The calling program must contain a *DEFINE FILE* statement for each logical unit which is to be accessed. This is of the form

DEFINE FILE 21(4480,200,L,K21).

The return code (*NERR*) should be examined by the calling program to determine the success of the attempted retrieval, and appropriate action should be taken. Large data sets can be retrieved in small sections if desired. The subroutine will fill the dimensions given (*IDEM*) and return an error code of 5 if the data set is not complete. The calling program can then move or otherwise process the first segment of data and then call *FETCH* again with *NERR* still equal to 5. The subroutine will then retrieve the next *IDEM* words of the data provided no other calls have been made to *FETCH* between these two calls. In all other cases it is important to ensure that *NERR* is not equal to 5 prior to the call.

A program listing of subroutine *FETCH* appears in Appendix B.

VI. SUBROUTINE SAVE

The storage of data sets in the data base is handled by subroutine *SAVE* which is called from the user's data collection and/or file management program. The calling sequence is

CALL SAVE(NAME,LENGTH,IADD,NERR)

where

NAME is the name, or sequence number, of the data set and may be any integer number allowed by the machine except -7777, -8888, or -9999,

LENGTH is the length of the data set (not including the *HEADER*) in words,

IADD is a flag. If $IADD \geq 0$, the data set is to be added to the data base and the directory is to be updated; if $IADD < 0$, the data are to be added to the last existing data set (without including a header record). The value of *LENGTH* in the header record will be increased by the length of the additional data,

and

NERR is an error return code; it may have the following values:

- 0 — No error; data storage was successful,
- 1 — data base is full; insufficient space between that last successful storage and the index,
- 2 — data set not found (when attempting to add to existing data set),
- 3 — no data sets in file (when attempting to add to existing data set),
- 4 — duplicate data set name (when attempting to create a new data set),
- 6 — reserved work used as data set name (-7777, -8888, or -9999).
- 7 — data set to be concatenated is not last.

The same common blocks and *DEFINE FILE* statements necessary for subroutine *FETCH* are also required to use subroutine *SAVE*.

The calling program should test the error flag, *NERR*, after calling the subroutine and take appropriate action. New information can be added to an existing data set by concatenation provided that the data set is last in the file. Program *MANAGER*, discussed in Sect. III, provides the capability to move any data set to last.

A source listing of subroutine *SAVE* appears in Appendix C.

VII. SYSTEM PROGRAMMING NOTES

There are no special systems considerations for *MANAGER*, *FETCH*, or *SAVE*. *MANAGER* is compiled and link-edited with the catalogued procedure

```
$JOB
$ASS BO = CS
$COMPILE MANAGER,,,,LINK
```

and the load module may be catalogued as follows

```
$JOB
$ASS BI=CS
$EXE CAT
FILE LMB
CAT MANAGER,MAN,POVER,B.00,BUF=4
EXIT
```

It is seen that the task is catalogued in the *LMB* library as a privileged overlay of Task B. The *BUF=4* is required to provide space for the *FORTTRAN* direct-access I/O buffers (one buffer for each *DEFINE FILE* statement).

APPENDIX A

SOURCE LISTING OF PROGRAM MANAGER

```

1 C *****
2 C *
3 C *
4 C *
5 C *
6 C * COMWRD - ARRAY STORING ALL ALLOWABLE COMMANDS
7 C * COMAND - COMMAND RECEIVED AS INPUT
8 C * NCOM - MAXIMUM NUMBER OF COMMANDS
9 C *****
10 C PROGRAM MANAGER
11 C *** VERSION B.00 25 SEPTEMBER 1979 *****
12 DIMENSION HEDREC(100), INDEX(2,50), I(16), N(16), OLDDHDR(100)
13 DIMENSION NEWNDX(2,50), OLDNDX(2,50), ITIME(6)
14 DIMENSION NINDX1(2,50), NINDX2(2,50)
15 DIMENSION SCALAR(5), COMAND(3,20)
16 DIMENSION IDARAY(3,20), ID(3)
17 DIMENSION JDATA(4100)
18 INTEGER REINDX(2,100), OTUNIT, DPUNIT, COMWRD(3), STR(10)
19 INTEGER HEDREC, FRETKS, OPTION, INAME, OLDNDX, OLDDHDR, OLDPTR
20 INTEGER COMAND, OWNER(3), RNAME, QMARK
21 LOGICAL GAP, END, MASTER, ALL
22 COMMON/HEADER/HEDREC
23 COMMON/DATA/JDATA
24 COMMON/MANAGE/MAXREC, MXWRDS, LLU, IDEM
25 EQUIVALENCE (REINDX(1,1), NINDX1(1,1)), (REINDX(1,51), NINDX2(1,1))
26 EQUIVALENCE (HEDREC(52), TSAMP), (HEDREC(32), SCALAR(1))
27 EQUIVALENCE (HEDREC(18), ITIME(1)), (HEDREC(54), ABSINT)
28 EQUIVALENCE (RNAME, I(1))
29 DEFINE FILE 21(4480,200,L,K21)
30 DEFINE FILE 22(576,200,L,K22)
31 DEFINE FILE 31(4480,200,L,K31)
32 DEFINE FILE 32(576,200,L,K32)
33 DATA NOPT/14/, INUNIT /5/, OTUNIT /6/, NIDS /20/, MASTER/.FALSE./
34 DATA COMAND/'S','C','A','L','I','S','D','E','L','R',
35 X'E','N','M','O','O','D','S','D','B','A','C',
36 X'O','M','L','U','N','R','E','S','Z','A','P','E',
37 X'X','I','O','N','L','O','F','F','18*X'//,
38 X QMARK /?/, NCOM /20/
39 WRITE(OTUNIT,907)
40 907 FORMAT(1H,'PROGRAM MANAGER'/
41 X1H,'VERSION B.0 25 SEPTEMBER 1979')
42 C
43 C DETERMINE USER ID
44 C
45 WRITE(OTUNIT,903)
46 903 FORMAT(' ENTER USER ID. ')
47 80 READ(INUNIT,904) ID
48 904 FORMAT(3A1)
49 IF(ID(1).EQ.'R'.AND.ID(2).EQ.'W'.AND.ID(3).EQ.'H'
50 X) MASTER = .TRUE.
51 IF(ID(1).EQ.'J'.AND.ID(2).EQ.'S'.AND.ID(3).EQ.'L'
52 X) MASTER = .TRUE.
53 C...
54 C... DETERMINE LOGICAL UNIT
55 C...
56 90 WRITE(OTUNIT,900)
57 900 FORMAT(' WHICH LOGICAL UNIT IS TO BE PROCESSED?')
58 100 READ(5,971) STR
59 971 FORMAT(10A2)
60 IO=0
61 CALL ASCTOF(STR,IO,X,IFLAG)
62 IF(IFLAG.GE.1.AND.IFLAG.LE.2) GO TO 110
63 WRITE(OTUNIT,901)
64 901 FORMAT(' INVALID ENTRY...PLEASE TRY AGAIN')
65 GO TO 100
66 110 LU=IFIX(X)
67 MXWRDS=100
68 NDXMAX=MXWRDS/2
69 IF(LU.EQ.21.OR.LU.EQ.31) MAXREC=4480
70 IF(LU.EQ.22.OR.LU.EQ.32) MAXREC=576
71 READ(LU,MAXREC) INDEX
72 IF(INDEX(1,1).EQ.-9999) GO TO 120
73 WRITE(OTUNIT,960)
74 960 FORMAT(' *****'
75 X '23X,'DATA BASE HAS NOT BEEN INITIALIZED'//
76 X '*****'//)
77 GO TO 212
78 111 WRITE(OTUNIT,910)
79 910 FORMAT(' THE FOLLOWING OPTIONS ARE AVAILABLE:'//5X,
80 X 'SCA - SCAN (DISPLAY AVAILABLE SPACE)'//5X,
81 X 'LIS - LIST HEADER RECORDS')
82 IF(MASTER) WRITE(OTUNIT,912)

```



```

82 912 FORMAT(' ',4X,'DEL - DELETE DATA SET')
83 WRITE(OTUNIT,913)
84 913 FORMAT(' ',4X,'REN - RENAME DATA SET')
85 IF(MASTER) WRITE(OTUNIT,914)
86 914 FORMAT(' ',4X,'MOV - MOVE DATA SET TO LAST')
87 WRITE(OTUNIT,915)
88 915 FORMAT(' ',4X,'DSE - DUMP DATA SET'/5X,
89 X 'DBA - DUMP ENTIRE DATA BASE')
90 IF(MASTER) WRITE(OTUNIT,916)
91 916 FORMAT(' ',4X,'COM - COMPRESS')
92 WRITE(OTUNIT,917)
93 917 FORMAT(' ',4X,'LUN - RESET LOGICAL UNIT NO.')
94 WRITE(OTUNIT,963)
95 963 FORMAT(' ',4X,'ONL - ON-LOAD DATA')
96 WRITE(OTUNIT,964)
97 964 FORMAT(' ',4X,'OFF - OFF-LOAD DATA')
98 IF(MASTER) WRITE(OTUNIT,918)
99 918 FORMAT(' ',4X,'RES - RESTORE DATA BASE'/5X
100 X 'ZAP - ZAP DATA BASE')
101 WRITE(OTUNIT,919)
102 919 FORMAT(' ',4X,'EXI - STOP')
103 GO TO 130
104 C...
105 C... READ OPTION
106 C...
107 120 WRITE(OTUNIT,920)
108 920 FORMAT(' SELECT OPTION ( ? FOR PROMT)')
109 130 READ(5,970) COMWRD
110 970 FORMAT(3A1)
111 IF(COMWRD(1) .EQ. QMARK) GO TO 111
112 DO 131 M=1,NCOM
113 IF(COMWRD(1) .NE. COMAND(1,M) .OR. COMWRD(2) .NE. COMAND(2,M)
114 X .OR. COMWRD(3) .NE. COMAND(3,M)) GO TO 131
115 OPTION=M
116 GO TO 140
117 131 CONTINUE
118 WRITE(OTUNIT,901)
119 GO TO 130
120 140 IF(.NOT. MASTER .AND.
121 3 OPTION .EQ. 5 .OR.
122 4 OPTION .EQ. 8 .OR.
123 5 OPTION .EQ. 10 .OR.
124 6 OPTION .EQ. 11)) GO TO 120
125 GO TO (201,202,203,203,203,203,207,208,90,210,211,212,214,
126 X 214),OPTION
127 C...
128 C... SCAN OPTION
129 C...
130 201 NDXPTR=MAXREC
131 FRETCS=0
132 WRITE(OTUNIT,930) LU
133 930 FORMAT(' SCAN OF DATA BASE NUMBER ',I2//
134 X 'SEQ.',6X,'STARTING',5X,'LENGTH' CUMULATIVE'/' NO.',6X,
135 X ' ADDRESS (TRACKS) FREE TRACKS'//)
136 READ(LU,NDXPTR) NINDX1
137 240 NDXPTR=NDXPTR-1
138 READ(LU,NDXPTR) NINDX2
139 DO 311 J=1,NDXMAX
140 IF(REINDX(1,J) .EQ. -8888) GO TO 120
141 IF(REINDX(1,J) .EQ. -9999) GO TO 311
142 NTRKS=REINDX(2,J+1)-REINDX(2,J)-1
143 IF(REINDX(1,J) .NE. -7777) GO TO 230
144 FRETCS=FRETCS+REINDX(2,J+1)-REINDX(2,J)-1
145 GO TO 311
146 230 WRITE(OTUNIT,931) REINDX(1,J), REINDX(2,J),NTRKS,FRETCS
147 931 FORMAT(' ',I5,3(5X,I7))
148 311 CONTINUE
149 DO 235 K=1,NDXMAX
150 NINDX1(1,K)=NINDX2(1,K)
151 NINDX1(2,K)=NINDX2(2,K)
152 235 CONTINUE
153 WRITE(OTUNIT,936)
154 936 FORMAT(1H1)
155 GO TO 240
156 C...
157 C... LIST, OPTION = 2
158 C...
159 202 NDXPTR=MAXREC
160 260 READ(LU,NDXPTR) INDEX
161 WRITE(OTUNIT,933)
162 933 FORMAT(1H1,2X,'DSN',3X,'OWNER',4X,'LENGTH',8X,'TITLE'//)
163 DO 250 J=1,NDXMAX
164 IF(INDEX(1,J) .EQ. -9999) GO TO 250
165 IF(INDEX(1,J) .EQ. -8888) GO TO 120
166 IF(INDEX(1,J) .EQ. -7777) GO TO 250
167 READ(LU,INDEX(2,J)) HEDREC
168 NAME = HEDREC(3)
169 CALL CTA(NAME,OWNER)

```

```

170 WRITE(OTUNIT,932) HEDREC(1),OWNER,HEDREC(2),(HEDREC(3),K=76,
171 X MXWRDS)
172 932 FORMAT(' ',15,4X,3A1,4X,15,6X,25A2)
173 IF(1TIME(6).EQ.0) 1TIME(6)=32000
174 IMIN=MOD(1TIME(1),60)
175 IHR=(1TIME(1)-IMIN)/60
176 ITIC=MOD(1TIME(2),1TIME(6))
177 ISEC=(1TIME(2)-ITIC)/1TIME(6)
178 WRITE(OTUNIT,934) (HEDREC(K),K=20,22),1HR,IMIN,ISEC
179 934 FORMAT(1H,25X,'DATE: ',12,'-',12,'-',12,' ',TIME: ',
180 X12,' ',12,' ',12)
181 WRITE(OTUNIT,935) 1SAMP,ABSINT
182 935 FORMAT(1H,25X,'TSAMP = ',F6.3,10X,'ABSINT = ',E12.5)
183 WRITE(OTUNIT,937)(SCALER(K),K=1,3)
184 937 FORMAT(1H,25X,'MON = ',F7.0,3X,'TOT = ',F9.0,3X,'TIME = ',F7.1//)
185 250 CONTINUE
186 NDXPTR=NDXPTR-1
187 GO TO 260
188
189 C... DELETE,RENAME,MOVE, OR DUMP OPTION = 3, 4, 5, OR 6
190 C...
191 203 CONTINUE
192 WRITE(OTUNIT,940)
193 940 FORMAT(' ENTER DATA SET NAME')
194 264 READ(INUNIT,971) STR
195 IO = 0
196 CALL ASCTOF(STR,10,X,IFLAG)
197 IF(IFLAG.GE.1.AND. IFLAG.LE.2) GO TO 265
198 WRITE(OTUNIT,901)
199 GO TO 264
200 265 RNAME = IFIX(X)
201 NDXPTR=MAXREC
202 270 READ(LU'NDXPTR) INDEX
203 DO 280 J=1,NDXMAX
204 IF(INDEX(1,J).EQ. -9999) GO TO 280
205 IF(INDEX(1,J).NE. -8888) GO TO 290
206 WRITE(OTUNIT,950)
207 950 FORMAT(' DATA SET NOT FOUND')
208 GO TO 120
209 290 IF(INDEX(1,J).NE. I(1)) GO TO 280
210 C... IF CONTROL PASSES 290, DATA SET IS FOUND
211 READ(LU'INDEX(2,J)) HEDREC
212 NAME=HEDREC(3)
213 CALL CTA(NAME,OWNER)
214 IF(MASTER.OR.(OWNER(1).EQ.ID(1).AND.OWNER(2).EQ.ID(2).
215 XAND.OWNER(3).EQ.ID(3))) GO TO 293
216 WRITE(OTUNIT,906)
217 906 FORMAT(' DATA SET NOT YOURS')
218 GO TO 120
219 293 JPTR=INDEX(2,J)
220 IF(OPTION.EQ. 6) GO TO 206
221 IF(OPTION.NE. 4) GO TO 300
222 WRITE(OTUNIT,902)
223 902 FORMAT(' ENTER NEW NAME')
224 294 READ(5,971) STR
225 IO = 0
226 CALL ASCTOF(STR,10,X,IFLAG)
227 IF(IFLAG.GE.1.AND. IFLAG.LE.2) GO TO 295
228 WRITE(OTUNIT,901)
229 GO TO 294
230 295 RNAME = IFIX(X)
231 INDEX(1,J)=I(1)
232 HEDREC(1)=I(1)
233 GO TO 310
234 C...
235 C... DELETE FROM INDEX AND HEDREC, OPTION = 3 OR 5
236 C...
237 300 READ(LU'NDXPTR) OLDNDX
238 READ(LU'INDEX(2,J)) OLDDHDR
239 OLDPTR=NDXPTR
240 INDEX(1,J)=-7777
241 HEDREC(1)=-7777
242 C... REWRITE INDEX AND HEDREC, OPTION = 3, 4, OR 5
243 310 WRITE(LU'NDXPTR) INDEX
244 WRITE(LU'INDEX(2,J)) HEDREC
245 IF(OPTION.NE. 5) GO TO 120
246 JJ=J
247 C... MOVE DATA SET IF OPTION = 5
248 C... SEARCH FOR TERMINATOR
249 350 DO 340 K=JJ,NDXMAX
250 IF(INDEX(1,K).NE. -8888) GO TO 340
251 NSTRT=INDEX(2,K)
252 NRECS=HEDREC(2)
253 KK=K
254 C... KK IS THE INDEX OF THE ELEMENT CONTAINING THE TERMINATOR
255 GO TO 360
256 340 CONTINUE
257 NDXPTR=NDXPTR-1

```

```

258 READ(LU'NDXPTR) INDEX
259 JJ=1
260 GO TO 350
261 HEDREC(1)=I(1)
262 C... UPDATE AND REWRITE INDEX AND HEDREC
263 INDEX(1, KK)=I(1)
264 IF(KK .LT. NDXMAX) GO TO 361
265 WRITE(LU'NDXPTR) INDEX
266 NDXPTR=NDXPTR-1
267 KK=0
268 361 INDEX(2, KK+1)=INDEX(2, KK)+HEDREC(2)+1
269 INDEX(1, KK+1)=-8888
270 WRITE(LU'NDXPTR) INDEX
271 WRITE(LU'NSTRT) HEDREC
272 C... MOVE RECORDS
273 DO 370 K=1, NRECS
274 READ(LU'JPTR+K) HEDREC
275 IF(NSTRT+K .LT. NDXPTR-1) GO TO 369
276 WRITE(OTUNIT, 998)
277 998 FORMAT(' DATA BASE FULL')
278 WRITE(LU'OLDPTR) OLDNDX
279 WRITE(LU'OLDNDX(2, J)) OLDHDR
280 GO TO 120
281 369 WRITE(LU'NSTRT+K) HEDREC
282 370 CONTINUE
283 GO TO 120
284 280 CONTINUE
285 NDXPTR=NDXPTR-1
286 GO TO 270
287 C
288 C... DUMP DATA SET, OPTION = 6
289 C
290 C... DATA SET HAS BEEN LOCATED AT 203
291 206 NRECS=HEDREC(2)
292 WRITE(OTUNIT, 941)
293 213 READ(INUNIT, 971) STR
294 IO=0
295 CALL ASCTOF(STR, IO, X, IFLAG)
296 IF(IFLAG .GE. 1 .AND. IFLAG .LE. 2) GO TO 209
297 WRITE(OTUNIT, 901)
298 GO TO 213
299 209 DPUNIT=IFIX(X)
300 WRITE(OTUNIT, 962)
301 962 FORMAT(1H 'ASSIGN DUMP UNIT'//
302 X1H 'THEN TYPE "/B/R" TO RESUME')
303 CALL HOLD
304 WRITE(DPUNIT) HEDREC
305 DO 430 K=1, NRECS
306 READ(LU'JPTR+K) HEDREC
307 WRITE(DPUNIT) HEDREC
308 430 CONTINUE
309 GO TO 120
310 C
311 C... DUMP DATA BASE, OPTION = 7
312 C
313 207 WRITE(OTUNIT, 941)
314 941 FORMAT(' ENTER LOGICAL UNIT NO. TO RECEIVE DUMP')
315 400 READ(INUNIT, 971) STR
316 IO=0
317 CALL ASCTOF(STR, IO, X, IFLAG)
318 IF(IFLAG .GE. 1 .AND. IFLAG .LE. 2) GO TO 410
319 WRITE(OTUNIT, 901)
320 GO TO 400
321 410 LDUMP=IFIX(X)
322 DO 420 K=1, MAXREC
323 READ(LU'K) HEDREC
324 IF(LDUMP .NE. 6) GO TO 409
325 WRITE(LDUMP, 951) HEDREC
326 951 FORMAT(' ', 20I5)
327 GO TO 420
328 409 WRITE(LDUMP, K) HEDREC
329 420 CONTINUE
330 GO TO 120
331 C
332 C... COMPRESS, OPTION 8
333 C
334 208 NDXPTR=MAXREC
335 NDXWRT = MAXREC
336 READ(LU'NDXPTR) NINDX1
337 J=1
338 439 NDXPTR=NDXPTR-1
339 READ(LU'NDXPTR) NINDX2
340 C FIND FIRST DELETED ENTRY, IF ANY
341 440 IF(REINDEX(1, J) .NE. -8888) GO TO 445
342 WRITE(OTUNIT, 991)
343 991 FORMAT(' NO VACANT SPACE')
344 GO TO 120
345 445 IF(REINDEX(1, J) .NE. -7777) GO TO 450

```

```

346      IPTR=J
347 C      IPTR KEEPS TRACK OF WHERE INDEX ENTRIES ARE BEING MOVED TOO
348      NWRITE=REINDEX(2,J)
349 C      NWRITE KEEPS TRACK OF WHERE DATA IS BEING MOVED TOO
350      GO TO 455
351 450 J=J+1
352      IF(J .LE. NDXMAX) GO TO 440
353      DO 451 K=1,NDXMAX
354      NINDEX1(1,K)=NINDEX2(1,K)
355      NINDEX1(2,K)=NINDEX2(2,K)
356 451 CONTINUE
357      NDXWRT = NDXWRT - 1
358      J = 1
359      GO TO 439
360 C      FIRST VACANT SPACE FOUND
361 455 IF(J .LE. NDXMAX) GO TO 459
362      DO 456 K=1,NDXMAX
363      NINDEX1(1,K)=NINDEX2(1,K)
364      NINDEX1(2,K)=NINDEX2(2,K)
365 456 CONTINUE
366      NDXPTR=NDXPTR-1
367      READ(LU'NDXPTR) NINDEX2
368      J=J-NDXMAX
369      IPTR = IPTR - NDXMAX + 1
370      J = J + 1
371 460 IF(REINDEX(1,J) .NE. -8888) GO TO 465
372      REINDEX(1,IPTR)=-8888
373      WRITE(LU'NDXPTR+1) NINDEX1
374      GO TO 420
375 465 IF(REINDEX(1,J) .EQ. -7777) GO TO 470
376      IREAD=REINDEX(2,J)
377 C      IREAD DEEPS TRACK OF WHERE DATA IS BEING MOVED FROM
378 C      READ HEADER RECORD & FIND NRECS
379      READ(LU'IREAD) HEDREC
380      NRECS=HEDREC(2)
381 C      REWRITE HEADER RECORD
382      WRITE(LU'NWRITE) HEDREC
383 C      MOVE DATA
384      DO 466 K=1,NRECS
385      READ(LU'IREAD+K) HEDREC
386      WRITE(LU'NWRITE+K) HEDREC
387      NBUG = NWRITE + K
388 466 CONTINUE
389 C      UPDATE INDEX
390      REINDEX(1,IPTR)=REINDEX(1,J)
391      NWRITE=NWRITE+1+NRECS
392      REINDEX(2,IPTR+1)=NWRITE
393      WRITE(LU'NDXPTR+1) NINDEX1
394      IPTR=IPTR+1
395 470 J=J+1
396      IF(IPTR .LE. NDXMAX .AND. J .GT. 2*NDXMAX) GO TO 480
397      IF(IPTR .LE. NDXMAX) GO TO 460
398      DO 475 K=1,NDXMAX
399      NINDEX1(1,K)=NINDEX2(1,K)
400      NINDEX1(2,K)=NINDEX2(2,K)
401 475 CONTINUE
402      NDXWRT = NDXWRT - 1
403      IPTR = IPTR - NDXMAX
404 480 NDXPTR=NDXPTR-1
405      J = J - NDXMAX
406      READ(LU'NDXPTR) NINDEX2
407      GO TO 460
408 C
409 C... RESTORE DATA BASE, OPTION =10
410 C
411 210 WRITE(OTUNIT,942)
412 942 FORMAT(' ENTER LOGICAL UNIT NO. TO SUPPLY INPUT')
413 610 READ( 5,971) SIR
414      IO=0
415      CALL ASCTOF(STR,10,X,IFLAG)
416      IF(IFLAG .GE. 1 .AND. IFLAG .LE. 2) GO TO 600
417      WRITE(OTUNIT,901)
418      GO TO 610
419 600 LIN=IFIX(X)
420      DO 620 K=1,MAXREC
421      READ(LIN,K) HEDREC
422      WRITE(LU,K) HEDREC
423 620 CONTINUE
424      GO TO 120
425 C
426 C... ZAP DATA BASE, OPTION = 11
427 C
428 211 NDXPTR=MAXREC
429      WRITE(OTUNIT,997)
430 997 FORMAT(' ONLY RWH CAN ZAP DATA BASE')
431      INLINE
432      REX, #11+#80
433      FINI

```

```

434      READ(LU'NDXPTR) INDEX
435      INDEX(1,1)=0
436      WRITE(LU'NDXPTR) INDEX
437      GO TO 120
438
439 C...  ONLOAD AND OFFLOAD, OPTIONS 13 AND 14
440 C...
441      214 IF(LU.EQ.21.OR.LU.EQ.22) GO TO 703
442      WRITE(OTUNIT,956) LU
443      956 FORMAT(1H,'CANNOT ON-LOAD OR OFF-LOAD FROM LU = ',I3)
444      GO TO 120
445      703 IF(OPTION.EQ.13) WRITE(OTUNIT,952)
446      952 FORMAT(1H,'ENTER SEQUENCE NUMBER OF FIRST AND LAST'/
447      X1H,'DATA SETS TO BE ON-LOADED')
448      IF(OPTION.EQ.14) WRITE(OTUNIT,953)
449      953 FORMAT(1H,'ENTER SEQUENCE NUMBER OF FIRST AND LAST'/
450      X1H,'DATA SETS TO BE OFF-LOADED')
451      READ(UNIT,971) STR
452      IO=0
453      CALL ASCIOF(STR,IO,X,IFLAG)
454      IF(IFLAG.GE.1.AND.IFLAG.LE.2) GO TO 700
455      WRITE(OTUNIT,901)
456      GO TO 213
457      700 ISTART=IFIX(X)
458      CALL ASCIOF(STR,IO,X,IFLAG)
459      IF(IFLAG.NE.0) GO TO 701
460      ISTOP=ISTART
461      GO TO 702
462      701 IF(IFLAG.GE.1.AND.IFLAG.LE.2) GO TO 702
463      WRITE(OTUNIT,901)
464      GO TO 213
465      702 ISTOP=IFIX(X)
466      CALL ASCIOF(STR,IO,X,IFLAG)
467      IF(IFLAG.NE.3) GO TO 706
468      ALL=.TRUE.
469      GO TO 708
470      ALL=.FALSE.
471      708 CONTINUE
472      WRITE(OTUNIT,954)
473      954 FORMAT(1H,'MOUNT DATA DISC'/
474      X1H,'AND ASSIGN LU'/
475      X1H,'THEN TYPE "/B/R" TO RESUME')
476      CALL HOLD
477      DO 704 J=ISTART,ISTOP
478      IF(OPTION.EQ.13) LLU=LU+10
479      IF(OPTION.EQ.14) LLU=LU
480      NAME = J
481      CALL FETCH(NAME,NERR)
482      IF(NERR.EQ.0) GO TO 705
483      IF(NERR.EQ.2) GO TO 709
484      WRITE(OTUNIT,957) J
485      957 FORMAT(1H,'ERROR IN FETCH FOR SEQ. NO. = ',I6)
486      GO TO 704
487      709 WRITE(OTUNIT,955) J
488      955 FORMAT(' ','SEQ = ',I6)
489      GO TO 704
490      705 IF(ALL.AND.MASTER) GO TO 707
491      IF(IACAN(ID).EQ.HEDREC(3)) GO TO 707
492      WRITE(OTUNIT,959) J
493      959 FORMAT(' ','DATA SET',I6,' NOT YOURS')
494      GO TO 704
495      707 IF(OPTION.EQ.13) LLU=LU
496      IF(OPTION.EQ.14) LLU=LU+10
497      IADD = 0
498      LENGTH = HEDREC(2)*100
499      CALL SAVE(NAME,LENGTH,IADD,NERR)
500      IF(NERR.EQ.0) GO TO 704
501      704 CONTINUE
502      WRITE(OTUNIT,958)
503      958 FORMAT(1H,'OK!'/
504      X1H,'DISMOUNT DATA DISC'/
505      X1H,'THEN TYPE "/B/R" TO RESUME')
506      CALL HOLD
507      GO TO 120
508
509 C...  STOP, OPTION = 12
510 C...
511      212 STOP
512      END

```

```

513      SUBROUTINE ASCTOP(ISTR,IO,X,IFLAG)
514      C *****
515      C *
516      C * ISTR - STRING TO BE SEARCHED
517      C * IO - STARTING POSITION FOR THE SEARCH
518      C * X - VALUE TO BE RETURNED
519      C * IFLAG - FLAG RETURNED:
520      C *          0 INDICATES OR NULL FIELD OR END OF LINE
521      C *          1 INDICATES NO DECIMAL POINT FOUND, ONE IS
522      C *          2 INDICATES OK
523      C *          3 INDICATES ILLEGAL CHARACTER FOUND
524      C *****
525      C
526      C DIMENSION K(8)
527      C INTEGER ISTR(10)
528      C LOGICAL TESTB
529      C INTEGER NULL,BLNK
530      C DATA NULL/4Z0000/,BLNK/' '//
531      C DO 10 I=1,10
532      C   J=11-I
533      C   IF(ISTR(J).NE.BLNK) GO TO 15
534      C   ISTR(J)=NULL
535      C 10 CONTINUE
536      C 15 M=0
537      C   L=0
538      C   CALL COLLEC(ISTR(1),IO,K,M,L)
539      C   IF(M.NE.2) GO TO 90
540      C
541      C ... NO PARAMTER FOUND
542      C
543      C   IFLAG = 0
544      C   RETURN
545      C
546      C 90 IF( TESTB(L,5)) GO TO 100
547      C ...
548      C ... NON-NUMERIC PARAMETER FOUND
549      C ...
550      C   IFLAG=3
551      C   RETURN
552      C ...
553      C ... NUMERIC PARAMETER
554      C ...
555      C 100 IPARM=IABIN(K,L)
556      C   X=FLOAT(IPARM)
557      C   IFLAG=2
558      C   RETURN
559      C END

```

EXIT
\$AVR CI 4
\$END DO

APPENDIX B

SOURCE LISTING OF SUBROUTINE *FETCH*

```

1  SUBROUTINE FETCH(NAME,NERR)
2  C... VERSION A.01 --- AUGUST 21,1979.
3  C... NXTNO - INDEX OF THE POSITION IN THE INDEX RECORD
4  C... WHERE INCOMIN DATA SET WILL BE ENTERED
5  C... NDXMAX - NUMBER OF DATA SET ENTRIES POSSIBLE IN EACH
6  C... INDEX RECORD (MXWRDS/2)
7  C... IDEM - NO OF WORDS IN DATA ARRAY IDATA
8  DIMENSION IDATA(1),INDEX(2,50),NDATA(100)
9  INTEGER HEDREC(100)
10 LOGICAL FIRST
11 COMMON/DATA/IDATA
12 COMMON/HEADER/HEDREC
13 COMMON/MANAGE/MAXREC,MXWRDS,LLU,IDEM
14 DATA FIRST/.TRUE./
15 LLU=LLU
16 NDXPTR=MAXREC
17 NDXMAX=MXWRDS/2
18 IF(NAME.NE.-8888.AND.NAME.NE.-9999) GO TO 80
19 WRITE(6,906)
20 906 FORMAT(' RESERVED WORD (',15,') USED AS DATA SET NAME')
21 NERR=6
22 RETURN
23 80 IF(FIRST) GO TO 90
24 IF(NERR.NE.5) GO TO 90
25 NSTRT=NSTRT+(IDEM/MXWRDS)
26 GO TO 160
27 90 FIRST=.FALSE.
28 C...
29 C... DETERMINE IF DATA BASE HAS BEEN INITIALLIZED
30 C...
31 IOERR=1
32 READ(LU'NDXPTR',ERR=200) INDEX
33 IF(INDEX(1,1).EQ.-9999) GO TO 100
34 WRITE(6,900)
35 900 FORMAT(' DATA BASE HAS NOT BEEN INITIALLIZED')
36 NERR=3
37 RETURN
38 C...
39 C... SEARCH FOR NAME
40 C...
41 100 DO 110 I=1,NDXMAX
42 IF(INDEX(1,I).NE.-8888) GO TO 120
43 WRITE(6,910)
44 910 FORMAT(' DATA SET NOT FOUND')
45 NERR=2
46 RETURN
47 120 IF(INDEX(1,I).NE.NAME) GO TO 110
48 NPTR=INDEX(2,I)
49 GO TO 130
50 110 CONTINUE
51 C...
52 C... IF CURRENT INDEX RECORD IS EXHAUSTED BEFORE NAME OR TERMINATOR IS
53 C... FOUND, DECREMENT NDXPTR AND READ NEXT INDEX RECORD
54 C...
55 NDXPTR=NDXPTR-1
56 IOERR=2
57 READ(LU'NDXPTR',ERR=200) INDEX
58 GO TO 100

```



```

59 C... BEGIN READING DATA SET
60 C...
61 C...
62 130 IOERR=3
63 READ(LU,NPTR,ERR=200) HEDREC
64 NSTRT=NPTR+1
65 NSTOP=NPTR+HEDREC(2)
66 160 JSTRT=0
67 DO 140 I=NSTRT,NSTOP
68 IREC=I
69 IOERR=4
70 READ(LU,IREF,ERR=200) NDATA
71 DO 150 J=1,MXWRDS
72 IDATA(JSTRT+J)=NDATA(J)
73 150 CONTINUE
74 IF(JSTRT.LT.IDEM-MXWRDS) GO TO 170
75 IF(JSTRT.EQ.IDEM-MXWRDS) GO TO 333
76 NERR=5
77 WRITE(6,905)
78 905 FORMAT(' DATA SET READ IS TOO LARGE FOR ARRAY')
79 333 RETURN
80 170 JSTRT=JSTRT+MXWRDS
81 140 CONTINUE
82 NERR=0
83 RETURN
84 200 WRITE(6,9000) IOERR
85 9000 FORMAT(1H,'DIRECT ACCESS I/O ERROR IN FETCH, IOERR =',I5)
86 RETURN
87 END
EXIT
$AUR CI 4
$END DO
$LIST MANAGER
$NOP
$EXE SED

```

APPENDIX C

SOURCE LISTING OF SUBROUTINE SAVE

```

1  SUBROUTINE SAVE(NAME,LENGTH,IADD,NERR)
2  C *** VERSION A.03 ----- AUGUST 21, 1979.
3  C... NDXPTR - ADDRESS OF INDEX RECORD CURRENTLY IN CORE
4  C... MAXREC - MAXIMUM NO OF RECORDS IN DIRECT ACCESS FILE
5  C... NPTR - ADDRESS OF THE RECORD CURRENTLY BEING WRITTEN
6  C... NRCRDS - NO OF RECORDS WRITTEN IN DATA SET
7  C... MXWRDS - NO OF WORDS IN A RECORD
8  C... NXTNO - INDEX OF THE POSITION IN THE INDEX RECORD
9  C... WHERE INCOMING DATA SET WILL BE ENTERED
10 C... NDXMAX - NUMBER OF DATA SET ENTRIES POSSIBLE IN EACH
11 C... INDEX RECORD (MXWRDS/2)
12 C... IDEM - NO OF WORDS IN DATA ARRAY IDATA
13 DIMENSION IDATA(1),INDEX(2,50)
14 INTEGER HEDREC(100)
15 LOGICAL EMPTY
16 COMMON/DATA/IDATA
17 COMMON/HEADER/HEDREC
18 COMMON/MANAGE/MAXREC,MXWRDS,LLU,IDEM
19 LLU=LLU
20 NDXPTR=MAXREC
21 NDXMAX=MXWRDS/2
22 IF(NAME.NE.-9999.AND.NAME.NE.-8888
23 X.AND.NAME.NE.-7777) GO TO 90
24 WRITE(6,906) NAME
25 906 FORMAT(' RESERVED WORD ('',15,'') USED AS DATA SET NAME')
26 NERR=6
27 RETURN
28 C...
29 C... DETERMINE IF DATA BASE HAS BEEN INITIALIZED
30 C...
31 IOERR=1
32 90 READ(LLU,NDXPTR,ERR=200) INDEX
33 IF(INDEX(1,1).EQ.-9999) GO TO 100
34 EMPTY=.TRUE.
35 INDEX(1,1)=-9999
36 INDEX(2,1)=1
37 INDEX(2,2)=1
38 NXTNO=2
39 GO TO 220
40 100 EMPTY=.FALSE.
41 C...
42 C... IF DATA SETS ARE TO BE CONCATENATED, DO THE FOLLOWING
43 C...
44 110 IF(IADD.GE.0) GO TO 130
45 IF(.NOT.EMPTY) GO TO 120
46 WRITE(6,901)
47 901 FORMAT(' NO DATA SETS IN FILE')
48 NERR=3
49 RETURN
50 120 DO 140 I=1,NDXMAX
51 IF(INDEX(1,I).EQ.-8888) GO TO 150
52 IF(INDEX(1,I).NE.NAME) GO TO 140
53 IF(INDEX(1,I+1).EQ.-8888) GO TO 190
54 WRITE(6,905)
55 905 FORMAT(' DATA SET TO BE CONCATENATED IS NOT LAST')
56 NERR=7
57 RETURN
58 190 NPTR=INDEX(2,I)

```

```

59      IOERR=0
60      READ(LU,NPTR,ERR=200) HEDREC
61      NRCRDS=HEDREC(2)
62      NPTR=NPTR+NRCRDS
63      GO TO 160
64 140 CONTINUE
65      NDXPTR = NDXPTR - 1
66      IOERR=3
67      READ(LU,NDXPTR,ERR=200) INDEX
68      GO TO 120
69 150 WRITE(6,902)
70 902 FORMAT(' DATA SET NOT FOUND')
71      NERR=2
72      RETURN
73 C...
74 C... SEARCH FOR TERMINATOR
75 C...
76 130 DO 210 I=1,NDXMAX
77      IF(INDEX(1,I) .NE. NAME) GO TO 170
78      WRITE(6,903)
79 903 FORMAT(' DUPLICATE DATA SET NAME')
80      NERR=4
81      RETURN
82 170 IF(INDEX(1,I) .NE. -8888) GO TO 210
83      NXTNO=I
84      GO TO 220
85 210 CONTINUE
86 C... CURRENT INDEX RECORD EXHAUSTED BEFORE NAME OR
87 C... TERMINATOR WAS FOUND. DECREMENT NDXPTR AND READ
88 C... NEXT INDEX RECORD
89      NDXPTR=NDXPTR-1
90      IOERR=4
91      READ(LU,NDXPTR,ERR=200) INDEX
92      GO TO 130
93 C...
94 C... WRITE DATA SET
95 C...
96 220 CONTINUE
97      HEDREC(1)=NAME
98      NPTR=INDEX(2,NXTNO)
99      NRCRDS=0
100 160 JSTRT=1
101      JSTOP=MXWRDS
102 230 NPTR=NPTR+1
103      IF(NPTR .GE. NDXPTR-1) GO TO 499
104      NRCRDS=NRCRDS+1
105      IOERR=5
106      WRITE(LU,NPTR,ERR=200)(IDATA(J),J=JSTRT,JSTOP)
107      JSTRT=JSTRT+MXWRDS
108      JSTOP=JSTOP+MXWRDS
109      IF(JSTRT .LT. LENGTH) GO TO 230
110 C...
111 C... COMPLETE AND WRITE HEADER RECORD
112 C...
113      HEDREC(2)=NRCRDS
114      IOERR=6
115      WRITE(LU,INDEX(2,NXTNO),ERR=200) HEDREC
116 C...

```

```

117 C... UPDATE AND REWRITE INDEX RECORD
118 C...
119 INDEX(1,NXTNO)=NAME
120 IF(NXTNO.LT.NDXM/X) GO TO 240
121 IOERR=7
122 WRITE(LU'NDXPTR,ERR=200) INDEX
123 IOERR=8
124 READ(LU'MAXREC,ERR=200) INDEX
125 INDEX(2,1)=INDEX(2,1)+1
126 IOERR=9
127 WRITE(LU'MAXREC,ERR=200) INDEX
128 NDXPTR=NDXPTR-1
129 NXTNO=0
130 240 INDEX(1,NXTNO+1)=8888
131 INDEX(2,NXTNO+1)=NPTR+1
132 IOERR=10
133 WRITE(LU'NDXPTR,ERR=200) INDEX
134 GO TO 500
135 499 WRITE(6,900)
136 900 FORMAT(' DATA BASE FULL: GO AWAY')
137 NERR=1
138 RETURN
139 500 NERR=0
140 RETURN
141 200 WRITE(6,9000) IOERR
142 9000 FORMAT(1H,'DIRECT ACCESS I/O ERROR IN SAVE. IOERR=',15)
143 RETURN
144 END
EXIT
$AVR CI 4
$END DO
$LIST FEICH
$NOP
$EXE SED

```